# LVGL UI Detector

This machine learning project is a user interface widget detector for the Light and Versatile Graphics Library (LVGL). The base is the YOLOv8 model, trained on synthesized datasets created from the programs/scripts in this repository and corresponding submodules.

The purpose of the model is to detect and locate widgets from a provided screenshot to aid in automated testing procedures requiring this information.

It was trained specifically for the LVGL framework, with the following widgets implemented in the dataset generator:

- Arc
- Bar
- Button
- Buttonmatrix
- Calendar
- Checkbox
- Dropdown
- Label
- Roller
- Scale
- Slider
- Spinbox
- Switch
- Table
- Textarea
- ...(more to come in the future)

This repository serves as the main entry point for the project and contains the main documentation as well as the project structure.
It also features a docker image for development purposes to work inside Jupyter notebook.
It additionally contains a few scripts for the container and also for inspecting or uploading datasets manually.

The image and scripts are **for my personal use** and will be deprecated in the future.

## Project structure

The project is divided into several submodules, each serving a specific purpose:

- UI Generator (Version 1): A modified version of the LVGL simulator for PC that generates a single random UI and captures a screenshot with annotations, using C and LVGL. This generator was originally used for UI screenshot generation, but was **deprecated** in favor of **version 2**, which uses micropython as a base.

- UI Generator (Version 2): An updated version of the UI generator that uses micropython and corresponding LVGL bindings as a base. This version is more flexible, easier to use and more maintenable than the original version.
- UI Randomizer: A python script module which uses the UI generator to create multiple UI screenshots with annotations in a single CLI interface. The output is organized into a dataset in YOLO format for training the model. **This module is generally deprecated** in favor of the UI Detector submodule, which combines the functionality of the UI Randomizer with YOLO training and optimization pipelines into a single repository.
- UI Detector: A repository containing multiple scripts/pipelines for generating datasets, training the YOLOv8 Model and hyperparameter optimization. **This repository is the main focus of the project.** It relies heavily on the usage of ClearML for experiment tracking, model optimization and dataset versioning.
- Paper: The repository containing the exposé and final paper for my bachelor thesis, describing the theoretical background, the project structure and the results of the project. Results include the evaluation of the base model and the project as a whole. Since this project and its submodules are subject to change in the future, a list of commits is provided in the paper repository, which showcase the state of the project at the time of finishing the paper. The final source state of the project from the perspective of the paper can also be found in the release marked `paper-final` of this repository.

# Dependency management

All projects use Poetry for python package creation and dependency management. This repository as well as each submodule contains a `pyproject.toml` file with the necessary dependencies to work with the project. **Please refer to the README of each submodule for more information on how to set up the environment.**

The main project has a `pyproject.toml` due to historic reasons, since it used to contain the code of the UI Detector submodule. It now serves more as a placeholder and starting point if new submodules would be added. In general, **the main project's `pyproject.toml` should not be used for setting up an environment**, but rather serve as a documentation and starting point for the project. In the future *(after the paper is released)*, contents of the main project will get more cleaned up.

# ClearML

The project uses ClearML for experiment tracking, model optimization and dataset versioning. ClearML is a powerful tool that allows for easy tracking of experiments, datasets and models. It also provides a powerful hyperparameter optimization tool that can be used to optimize the model for the best possible performance.

Since the UI Detector submodule relies so heavily on ClearML, it is required to have an account and API key setup to use it. This is also the reason why the UI Randomizer submodule exists, since it can be used to run the generator repeatedly without the hard dependency on ClearML. For training purposes, the YOLOv8 python package as well as CLI tools can be used to train a model on generated datasets without requiring ClearML.

ClearML also provides the possibility of setting up a standalone server, which can be used to host locally. If you do not want to setup an account with ClearML directly, you are required to setup a local server to use the UI Detector submodule. You can find out more information on how to set up a local server in the ClearML server documentation.

## ClearML configuration

For simply using ClearML, the API keys can be provided directly in the CLI, in code or as environment variables. To connect to the ClearML SDK it is as simple as running `clearml-init` in your terminal and following the instructions. Checkout the Getting Started Tutorial for more details.

However, to effectively orchestrate tasks/pipelines you need to use the ClearML agent, which requires a configuration file.
An example ClearML configuration can be found in the ClearML agent repository, which contains a basic `clearml.conf` file. This file can be modified and stored in the target machine, mainly to configure the ClearML agent.

More information about the configuration file can be found in the ClearML config documentation.

Configuration options that need to be modified on the template are:

- `api.api_server`: The URL of the ClearML server you are using. If you are using the ClearML cloud service, this should be `https://api.clear.ml`, otherwise it should point to the URL of your local server.
- `api.web_server`: The URL of the ClearML web server you are using. If you are using the ClearML cloud service, this should be `https://app.clear.ml`, otherwise it should point to the URL of your local server.
- `api.files_server`: The URL of the ClearML files server you are using. If you are using the ClearML cloud service, this should be `https://files.clear.ml`, otherwise it should point to the URL of your local server.
- `api.credentials`: An object containing the `access_key` and `secret_key` for your account on the used ClearML server. They can be found/created in your ClearML account settings on the web interface. You may also configure these by setting the `CLEARML_API_ACCESS_KEY` and `CLEARML_API_SECRET_KEY` environment variables. More information on setting up these keys can be found in the ClearML WebApp documentation.

Beyond that, it is recommend to checkout the `agent` section of the configuration file to configure the agent according to your needs. The agent is used to run experiments on remote machines and can also be used to run the UI Detector submodule on a remote machine.

*(For the generator it requires a setup display server, such as the X window system)*

# LVGL UI Generator (Version 1)

Version 1 of the LVGL UI Generator is a modified version of the LVGL simulator for PC that generates a single random UI and captures a screenshot with annotations. It is based on the lv_port_pc_vscode repository, since that is the IDE I use and also the only one I dared to modify.

When launched with the proper arguments, it will open a window with a single container widget of the provided size placed inside. After rendering the UI, it will generate a screenshot of the container and save it to the provided path. It will also generate a text file with the bounding boxes of the placed widgets.

Further information can be found in the UI Generator v1 README.

# LVGL UI Generator (Version 2)

Version 2 of the LVGL UI Generator is an updated version of the UI generator that uses micropython and corresponding LVGL bindings as a base. This version is more flexible, easier to use and more maintenable than the original version.

It has two modes of operation:

- **Random mode**: Generates a random UI with a specified number of widgets placed on a white background. It requires a provided list of widget types to randomly choose from.
- **Design mode**: Generates a UI based on a provided JSON design file. The design file describes the whole window, including styles, widgets and certain properties. There is a special `random` widget, which can be used to randomize widget creation in certain areas of the design. This mode is useful in creating more realistic looking user interfaces, as the random mode does not accomodate for styles regarding the containers.

Further information about usage and setup can be found in the UI Generator v2 README.

# UI Randomizer

The UI Randomizer is a python script module which uses the UI generator to create multiple UI screenshots with annotations in a single CLI interface. The output is organized into a dataset in YOLO format for training the model. Additionally, the script can handle uploading of these manually created datasets to ClearML, if the specific `clearml` options are provided in the CLI.

Further information can be found in the UI Randomizer README.

# UI Detector

The UI Detector is a repository containing multiple scripts for generating datasets, training the YOLOv8 Model and hyperparameter optimization. **This repository is the main focus of the project.** It relies heavily on the usage of ClearML for experiment tracking, model optimization and dataset versioning.

Further information can be found in the UI Detector README.

# Docker image

A docker image is provided for development purposes. It is derived from the official Ultralytics docker image and contains all necessary dev-dependencies for the project *(i.e. JupyterLab)*.

The image can be built using the provided `Dockerfile` in the root directory of the repository. Additionally, there's a run_container.sh script provided for convenience in the scripts folder.

The script and image currently only serve development purposes for my own personal use and are not taking any other needs into account.

# VSCode workspace

The project was developed using Visual Studio Code.

The main repository contains the `ui-detector.code-workspace`, which organizes all submodules into seperate workspaces. All deprecated submodules are still included in the workspace, but are commented out to clean up the workspace view.

The workspace file is intended for development purposes and suggests the usage of a few extensions. These included suggestions are generally required to develop the submodules properly.

▶ Required extensions for the workspace

- Python
- Jupyter
- C/C++ Extension Pack

I personally use a few more extensions than that and to credit their creators/maintainers, a purely optional list is provided here as well.

▶ Optional extensions for this workspace *(in no particular order)*

- Project Manager
- Todo Tree
- Zotero LaTeX
- Citation Picker for Zotero
- LaTeX Workshop
- LaTeX Utilities
- Icons for Visual Studio Code
- GitHub Copilot
- Docker
- Git Graph
- Git History
- Local History
- MemoryView

---

# Author's Note

This project was created for my bachelor thesis at the University of Applied Sciences Technikum Vienna.

...and I am tired.

# Milestones

- [x] Create UI generator
- [x] Create UI randomizer
- [x] Create model
- [x] Make model predict the pre-defined widget list
- [x] Finish Exposé
- [x] Finish Exposé presentation
- [x] Complete README documentation in all submodules
- [ ] Finish bachelor thesis
- [x] Finish thesis presentation
- [ ] Did not catch burn-out syndrome

# Feature roadmap

- [x] Add more widgets to the generator and model
- [x] Improve dataset size (min. 50 per class) and training process (1-click-pipeline)
- [x] Add widget customization in generator (color, size, etc.)
- [x] Add design file system for randomizer & generator

# License

This project is licensed under the MIT License - see the LICENSE file for details.

# LVGL UI Detector

Source code for LVGL UI detector model with ClearML tasks for generating, training and optimization.

# Pre-requisites & Installation

This project uses Poetry for managing dependencies.

## Setting up the virtual environment

1. Install `poetry` package manager. See corresponding documentation for more information.
2. Run `poetry install` to install the dependencies and prepare the virtual environment.

# Usage

```
poetry run python src/<task>.py <task-arguments>
```

# Available tasks

The source contains the following tasks:

- `generate.py` - Task for generating a dataset using LVGL UI Generator v2 in either `random` or `design` mode.
- `train.py` - Task for training the YOLOv8 model on a provided dataset ID from ClearML.
- `optimize.py` - Task for optimizing the YOLOv8 model on a provided task ID from ClearML using Optuna.
- `tune.py` - Task for tuning on a provided dataset ID from ClearML using the tuning functionality of YOLO *(Their built-in hyperparameter optimization)*.

## Generate

The task will generate random UIs using the LVGL UI Generator v2.
It does so by repeatedly calling the generator using the chosen mode and provided parameters for it.

Errors during generation, as well as statistics about the generated dataset, are tracked and reported in the ClearML task.

After finishing the generation, the task will create a ClearML dataset from the output folder.

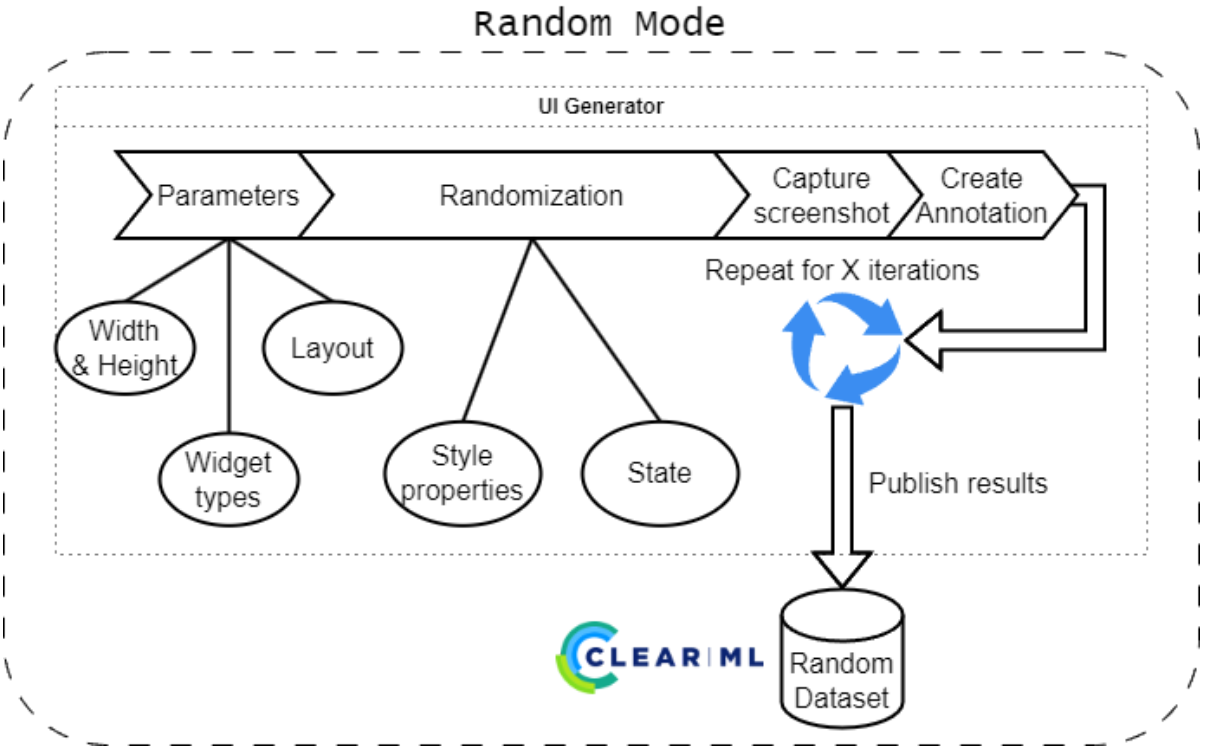
The details about this are described in the Dataset section, as it is the same for all modes.

▶ Help

```
usage: generate.py [-h] [-o OUTPUT_FOLDER] [--mpy-path MPY_PATH] [--mpy-main MPY_MAIN] [-d DATASET] [-s SPLIT_RATIO SPLIT_RATIO SPLIT_RATIO] [--no-dataset]
                   {random,design} ...

Generate UI Detector dataset

positional arguments:
  {random,design}       Type of LVGL UI generator to use
    random              Generate random UI
    design              Generate UI from design files

options:
  -h, --help            show this help message and exit
  -o OUTPUT_FOLDER, --output-folder OUTPUT_FOLDER
                        Output folder (default: tmp/output)
  --mpy-path MPY_PATH   Path to MicroPython binary (loads from environment MICROPYTHON_BIN if not provided) (default: None)
  --mpy-main MPY_MAIN   Path to main.py of micropython script (loads from environment MICROPYTHON_MAIN if not provided) (default: None)
  -d DATASET, --dataset DATASET
                        Custom name of the dataset written in the task comment (default: None)
  -s SPLIT_RATIO SPLIT_RATIO SPLIT_RATIO, --split-ratio SPLIT_RATIO SPLIT_RATIO SPLIT_RATIO
                        Split ratio for train, val, test (default: None)
```

```
  --no-dataset          Do not create a ClearML dataset (artifacts are added to Task) (default: False)
Subparser 'random'
usage: generate.py random [-h] [-W WIDTH] [-H HEIGHT] [-c COUNT] [-l LAYOUT] [-x AMOUNT]

options:
  -h, --help            show this help message and exit
  -W WIDTH, --width WIDTH
                        Width of the UI screenshot
  -H HEIGHT, --height HEIGHT
                        Height of the UI screenshot
  -c COUNT, --count COUNT
                        Number of widgets to create per output
  -l LAYOUT, --layout LAYOUT
                        The main container layout of the random UI ["grid", "flex", "none"]
  -x AMOUNT, --amount AMOUNT
                        Amount of outputs per widget class to create

Subparser 'design'
usage: generate.py design [-h] {local,remote,gpt} ...

positional arguments:
  {local,remote,gpt}  Variant of design generator to use
    local             Generate UIs from local design files
    remote            Generate UIs from remote design files
    gpt               Generate UIs using ChatGPT API

options:
  -h, --help          show this help message and exit
```

## Random mode



The random mode will create a UI window in the provided dimensions and fill it with randomly chosen widgets. This mode will always start from the full list of implemented types.

The provided `count` determines the amount of widgets per screenshot, while the `amount` determines the target amount of widgets per class to be included in the dataset.
Since the generator chooses widget types randomly, the amount of created widget types is tracked.
The generator will remove the widget type from the list of choices once the target amount for that type is reached or exceeded.

This helps ensuring that there is an appropriate and balanced amount of widget types in the dataset.

You can view statistics about the generated dataset in the scalars and reports of the ClearML task.

▶ Statistics created in random mode

## Design mode

The design mode will create UI windows using design files.

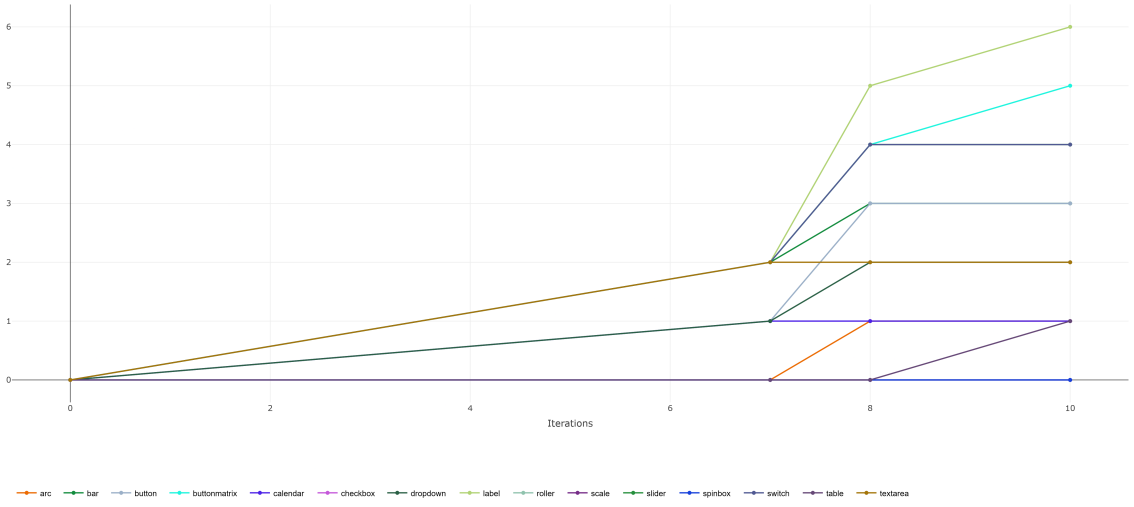
It supports additional modes of operation:

- `local` - Generate UIs from a folder containing local design files
- `remote` - Generate UIs from a remote location containing design files **(not implemented yet)**
- `gpt` - Generate UIs from design files generated by the ChatGPT API

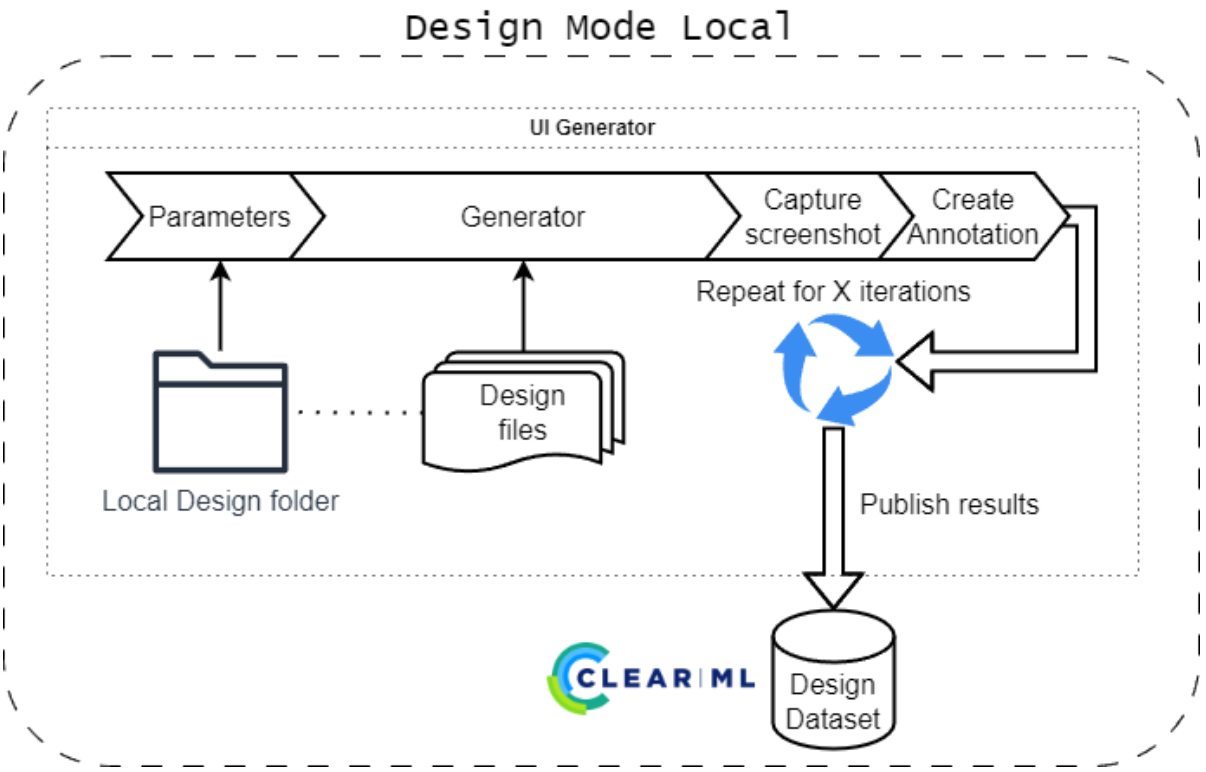
The design files are expected to be in a specific format, which is described in the Generator README.

You can also view the JSON schema for the design files in the design_schema.json file.

▶ Statistics in design mode



### Local mode




The local mode expects a folder containing JSON design files.

Design files are not validated against the schema, since the generator will report any missing properties or incorrect values as it encounters them.

The task will go through the folder, calling the generator for each design file, which determines the total iterations of the task.

It will attempt to retry generation up to 4 times, since the generator might fail due to memory issues, which can be resolved by retrying.
Overall this does not affect generation time significantly, since the generator is fast enough when exiting.

Any errors encountered during generation will be reported in the ClearML task and also tracked per iteration.

The created image and annotation file is then renamed and moved into the output folder.
The task will check each annotation file to create statistics about the amount of widgets per class in the dataset. When doing so, it will also verify the bounding box per widget to ensure it is within the image bounds. Any invalid bounding boxes will be reported in the ClearML task and also removed from the annotation file.

The task will show the following scalar statistics:

- Total amount of widgets created
- Total amount of generation errors
- Amount of widgets per class

▶ Help

```
usage: generate.py design local [-h] [-f DESIGN_FOLDER]

options:
  -h, --help            show this help message and exit
  -f DESIGN_FOLDER, --design-folder DESIGN_FOLDER
                        Source folder for design generator
```

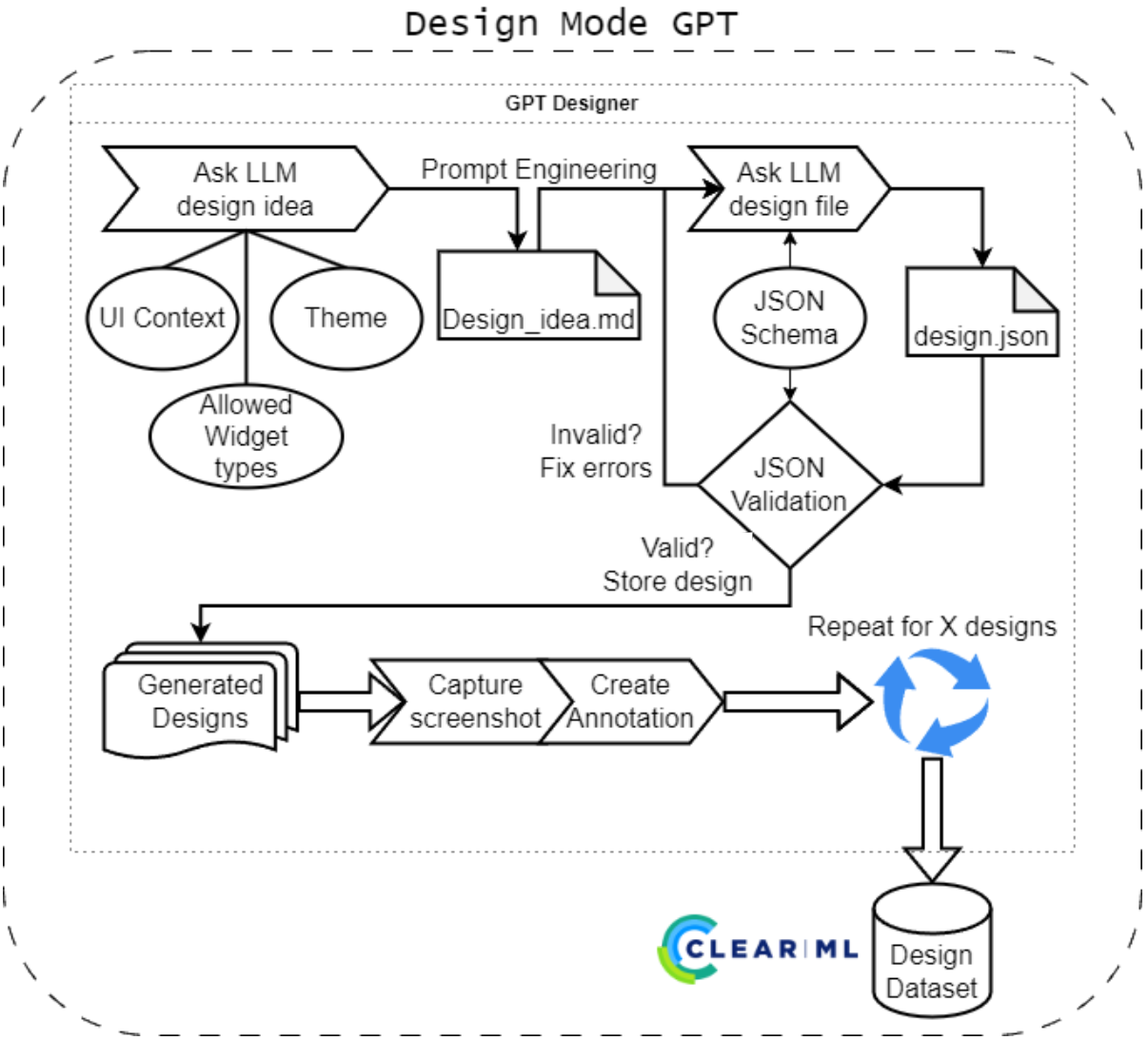
## Remote mode

The remote mode is not implemented yet.

▶ Help

```
usage: generate.py design remote [-h]

options:
  -h, --help  show this help message and exit
```

## GPT mode




Since creating design files manually can be time-consuming, the GPT mode allows generating design files using a LLM, namely the ChatGPT API.

This is done by first asking the LLM for a design idea in a randomly chosen context and theme. This list contains **10000 options** *(100 contexts combined with 100 themes)* and is hardcoded currently, see further below.

The design idea is then passed to the LLM again to generate the design file, using the available schema as guidance. The returned JSON by the LLM is validated against the schema and any errors are reported in the ClearML task. The errors are feeded back into the LLM for correction. This is re-attempted 3 times before giving up and moving to the next randomly chosen context and theme.

If the returned design file is valid, it is added to the list to be generated.

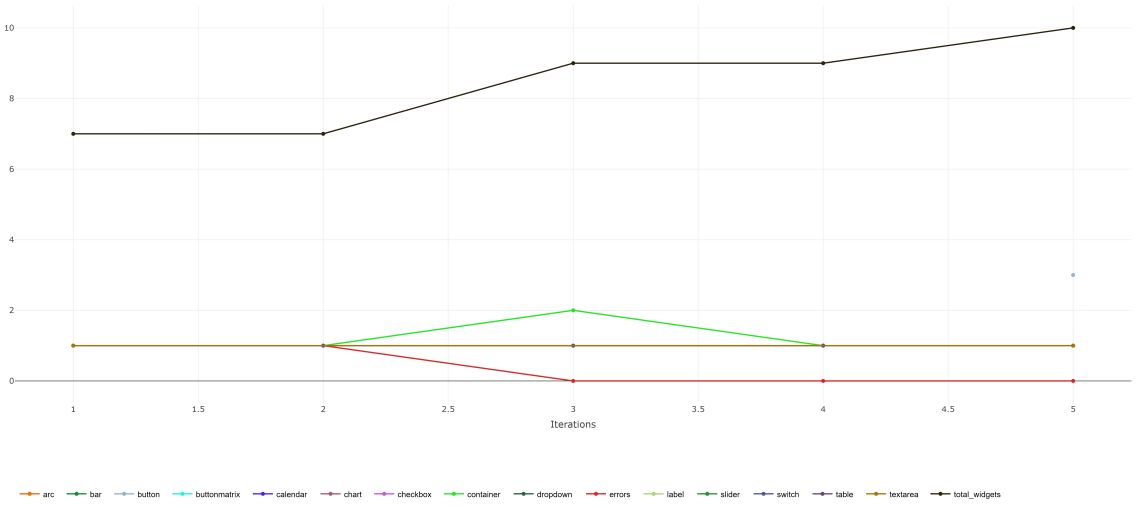
Once the desired amount of design files is reached, the task will go through the list and generate the UIs using the generator. This process is then the same as in the local mode.

▶ Help

```
usage: generate.py design gpt [-h] [--api-key API_KEY] --model MODEL [--max-
tokens MAX_TOKENS] [--designs DESIGNS] [--temperature TEMPERATURE | --top-p
TOP_P]

options:
  -h, --help            show this help message and exit
  --api-key API_KEY     ChatGPT API key
  --model MODEL         ChatGPT model name
  --max-tokens MAX_TOKENS
                        ChatGPT maximum tokens
  --designs DESIGNS     Number of designs to generate
  --temperature TEMPERATURE
                        ChatGPT sampling temperature
  --top-p TOP_P         ChatGPT top-p sampling
```

▶ Additional statistics in design mode GPT

The GPT designer reports on the amount of widgets used in a generated design *(per widget class)* for each iteration.




▶ 100 contexts and 100 themes used

```
# Prompted with: Create a list of 100 topics that an embedded user interface
could be about.
```

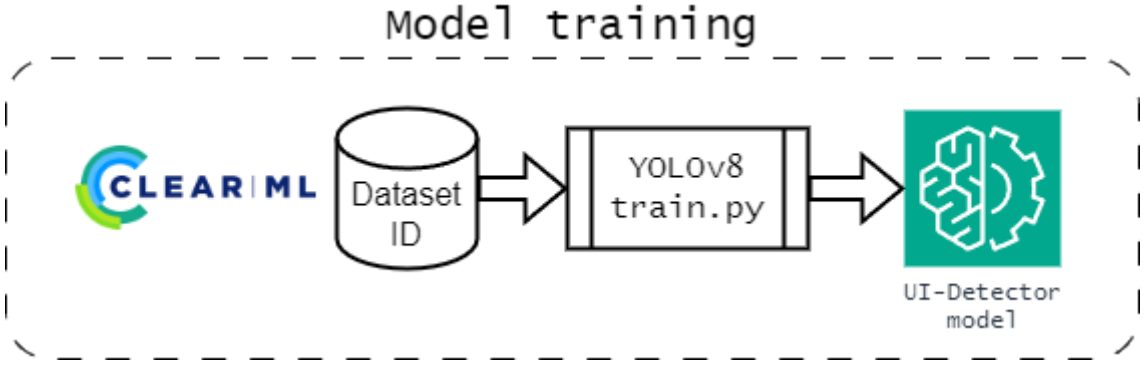
```
topics = [
    "Smart Home Control Systems",
    "Wearable Fitness Trackers",
    "Automotive Dashboard Displays",
    "Industrial Automation Interfaces",
    "Agricultural Monitoring Systems",
    "Medical Device Interfaces",
    "Drone Control Panels",
    "Retail Point of Sale Systems",
    "Smart Watches Interfaces",
    "Security System Controls",
    "Marine Navigation Systems",
    "Building Climate Control Systems",
    "Home Appliance Controls (e.g., Smart Refrigerators)",
    "Energy Management Displays",
    "Portable Music Players",
    "Electronic Thermostats",
    "Educational Tablets for Kids",
    "Emergency Alert Systems",
    "Water Purification System Controls",
    "Lighting Control Systems",
    "Portable Gaming Devices",
    "Smart Mirror Technologies",
    "Elevator Control Panels",
    "Vending Machine Interfaces",
    "Fitness Equipment Consoles",
    "Industrial Robot Controllers",
    "Smart Bed Controls",
    "Smart Glasses Interfaces",
    "Pet Tracking Devices",
    "Baby Monitoring Systems",
    "Digital Signage",
    "Ticketing Kiosks",
    "Virtual Reality Headset Interfaces",
    "Library Management Kiosks",
    "Smart Lock Interfaces",
    "Laboratory Equipment Interfaces",
    "Smart Pens",
    "Art Installation Controls",
    "HVAC Control Systems",
    "Railroad Monitoring Systems",
    "Handheld GPS Devices",
    "Digital Cameras",
    "Smart Toothbrushes",
    "Aircraft Cockpit Displays",
    "Electric Vehicle Charging Stations",
    "Soil Moisture Sensors",
    "Smart Jewelry",
    "Pipeline Monitoring Systems",
    "Waste Management Systems",
    "Personal Medical Devices (e.g., Insulin Pumps)",
    "Public Transportation Displays",
    "On-board Ship Computers",
    "Smart Plant Pots",
    "Industrial Pressure Sensors",
    "Interactive Museum Exhibits",
```

```
    "Smart Bicycle Systems",
    "Conference Room Booking Displays",
    "Augmented Reality Interfaces",
    "Remote Wilderness Cameras",
    "Interactive Retail Displays",
    "Spacecraft Control Interfaces",
    "Wireless Router Management",
    "Smart City Infrastructure Interfaces",
    "Factory Assembly Line Displays",
    "Car Rental Kiosks",
    "Airport Check-in Kiosks",
    "Digital Billboards",
    "Hospital Room Information Panels",
    "Power Grid Monitoring Systems",
    "Oil Rig Monitoring Interfaces",
    "Smart Suitcases",
    "Fishing Gear Electronics",
    "Underwater Exploration Devices",
    "Digital Menu Boards in Restaurants",
    "Emergency Vehicle Dashboards",
    "Voice-Controlled Home Assistants",
    "Smart Coasters (beverage temperature)",
    "Bicycle Sharing System Terminals",
    "Smart Shower Panels",
    "Mining Equipment Interfaces",
    "Forest Fire Detection Systems",
    "Smart Windows",
    "Interactive Dance Floors",
    "Smart Ring Interfaces",
    "Professional Camera Systems",
    "Home Brewing Systems",
    "Smart Mailboxes",
    "Autonomous Farm Equipment",
    "Wind Turbine Controls",
    "Smart Blinds and Curtains",
    "Logistics Tracking Systems",
    "Parking Garage Equipment",
    "Smart Helmet Displays",
    "Boat Instrumentation Panels",
    "Interactive Park Equipment",
    "Livestock Tracking Systems",
    "Remote Surgery Consoles",
    "Weather Monitoring Stations",
    "Smart Gloves",
    "Electronic Voting Machines"
]

# Prompted with: Create a list of 100 themes that could be applied to these
user interfaces.
themes = [
    " Minimalist",
    "Futuristic",
    "Retro",
    "High Contrast",
    "Dark Mode",
    "Light Mode",
```

```
    "Nature-inspired",
    "Nautical",
    "Neon Glow",
    "Earthy Tones",
    "Pastel Colors",
    "High Tech",
    "Art Deco",
    "Steampunk",
    "Material Design",
    "Flat Design",
    "3D Depth",
    "Monochrome",
    "Kids-Friendly",
    "Elderly-Friendly",
    "Luxury",
    "Industrial",
    "Sports",
    "Educational",
    "Seasonal (e.g., Winter, Summer)",
    "Holiday Themes (e.g., Christmas, Halloween)",
    "Cartoon",
    "Abstract",
    "Photorealistic",
    "Geometric",
    "Military",
    "Space Exploration",
    "Underwater",
    "Urban",
    "Rural",
    "Health Focused",
    "Accessibility Enhanced",
    "Cultural (e.g., Japanese, Mexican)",
    "Cyberpunk",
    "Virtual Reality",
    "Augmented Reality",
    "Transparent Interfaces",
    "Glass Effect",
    "Vintage Film",
    "Comic Book",
    "Parchment and Ink",
    "Origami",
    "Glow in the Dark",
    "Neon Signs",
    "Hand-drawn",
    "Watercolor",
    "Grunge",
    "Metallic",
    "Zen and Tranquility",
    "Casino",
    "Outer Space",
    "Sci-Fi",
    "Historical Periods (e.g., Victorian, Medieval)",
    "Typography-Based",
    "Animal Print",
    "Floral",
    "Ocean Waves",
```

```
    "Desert Sands",
    "Mountainous Terrain",
    "Tropical Paradise",
    "Arctic Freeze",
    "Jungle Theme",
    "Auto Racing",
    "Aviation",
    "Sailing",
    "Rock and Roll",
    "Hip Hop",
    "Classical Music",
    "Opera",
    "Ballet",
    "Theatre",
    "Film Noir",
    "Silent Film",
    "Neon Jungle",
    "Crystal Clear",
    "Witchcraft and Wizardry",
    "Steampunk Mechanisms",
    "Pop Art",
    "Renaissance Art",
    "Graffiti",
    "Pixel Art",
    "ASCII Art",
    "Mosaic",
    "Lego Style",
    "Board Game",
    "Video Game",
    "Dystopian",
    "Utopian",
    "Western",
    "Eastern",
    "Minimalist Text",
    "Bold Color Blocks",
    "Line Art",
    "Optical Illusions",
    "Neon Abstract"
]
combinations = [(t, c) for t in themes for c in topics]
```

# Train




The training task will train the YOLOv8 model on the provided dataset ID from ClearML.

The task will download the dataset and prepare it for training. It will then start the training process, which will be tracked in the ClearML task.
To allow for later editing and re-using of a task, the dataset path will be overrided with valid local path of the current environment, which can later be viewed in ClearML under parameter `General/data`.

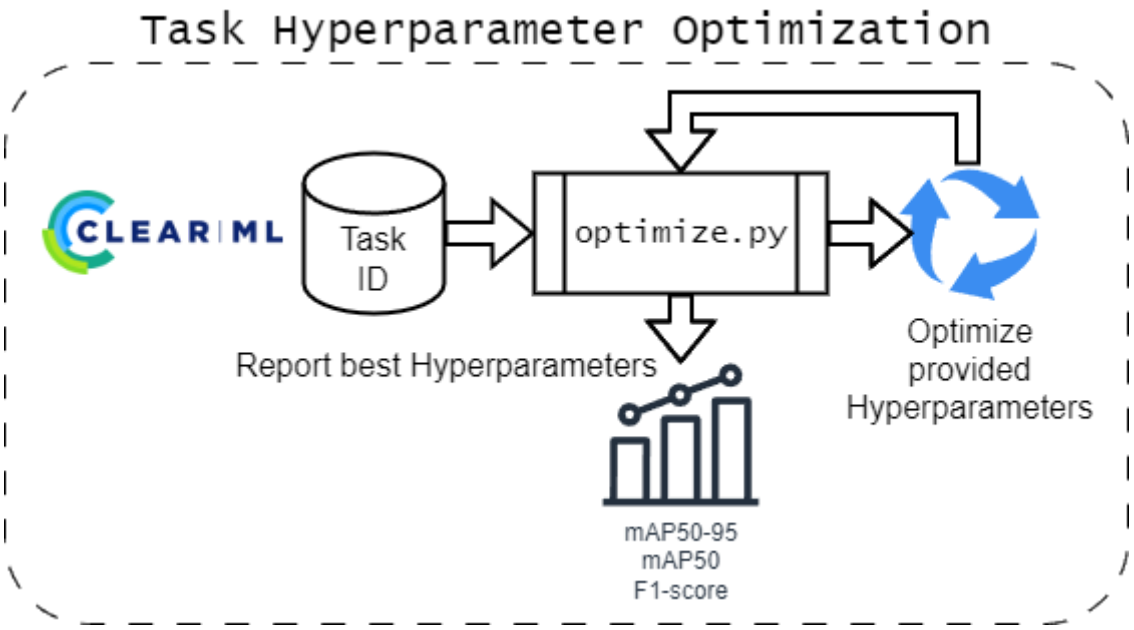
The CLI does not have any options for overriding hyperparameters, as ClearML provides a way to edit them in the task configuration.

The easiest way to work with the task is a training locally for 1 epoch and then clone the task in ClearML to edit the hyperparameters.

▶ Help

```
usage: train.py [-h] [--dataset DATASET] [--model MODEL] [--epochs EPOCHS] [--imgsz IMGSZ]

Train a YOLOv8 model on a dataset

options:
  -h, --help          show this help message and exit
  --dataset DATASET   Dataset ID to use for training (default: None)
  --model MODEL       Model variant to use for training (default: None)
  --epochs EPOCHS     Number of epochs to train for (default: 10)
  --imgsz IMGSZ       Image size for training (default: 640)
```

# Optimize




The optimization task will perform hyperparameter optimization on the paramaters of a provided task ID from ClearML using Optuna.

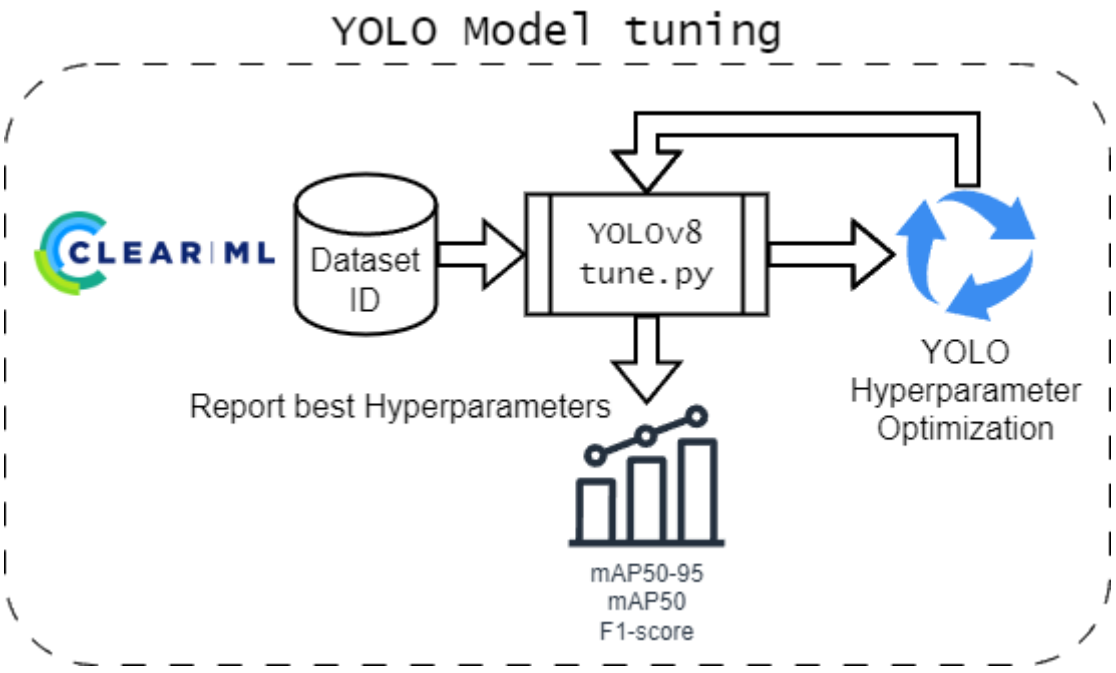
It will currently always optimize towards the `mAP50-95` metric, which is the mean average precision of the model on the validation dataset.

In the future, other target metrics will be added to the CLI options.

▶ Help

```
usage: optimize.py [-h] [--id ID] [--local] [--pool-period POOL_PERIOD] [--
max-jobs MAX_JOBS] [--max-concurrent MAX_CONCURRENT]
                   [--max-iterations MAX_ITERATIONS] [--time-limit
TIME_LIMIT] [--top-k TOP_K] [--execution-queue EXECUTION_QUEUE]

Optimize hyperparameters for a ClearML training task

options:
  -h, --help            show this help message and exit
  --id ID               Task ID to optimize (default: None)
  --local               Run the optimization locally (default: False)
  --pool-period POOL_PERIOD
                        Pool period in minutes (default: 5)
  --max-jobs MAX_JOBS   Maximum number of jobs to run (default: 25)
  --max-concurrent MAX_CONCURRENT
                        Maximum number of concurrent tasks (default: 2)
  --max-iterations MAX_ITERATIONS
                        Maximum number of iterations per job (default: 100)
  --time-limit TIME_LIMIT
                        Time limit for optimization in minutes (default: 120)
  --top-k TOP_K         Number of top experiments to print (default: 5)
  --execution-queue EXECUTION_QUEUE
                        Execution queue for optimization (default: training)
```

## Tune




The tune task will perform hyperparameter optimization on the provided dataset ID from ClearML using the `tune` function of the ultralytics engine.

It will then report the best hyperparameters found in the ClearML task.

Due to the nature of automatic reporting of ClearML, the main tuning operation will not be a visible experiment in ClearML, but each individual training run will be.

To get a more detailed visualization and also more control over the optimization process, it is recommended to use the `optimize` task.

▶ Help

```
usage: tune.py [-h] [--model MODEL] [--dataset DATASET] [--epochs EPOCHS] [--iterations ITERATIONS] [--imgsz IMGSZ] [--optimizer OPTIMIZER]

Tune dataset hyperparameters for a YOLO model

options:
  -h, --help            show this help message and exit
  --model MODEL         Model variant to use for tuning (default: None)
  --dataset DATASET     Dataset ID to use for tuning (default: None)
  --epochs EPOCHS       Number of epochs to train for (default: 30)
  --iterations ITERATIONS
                        Number of iterations to tune for (default: 100)
  --imgsz IMGSZ         Image size for tuning (default: 640)
  --optimizer OPTIMIZER
                        YOLO Optimizer (from: Adam, AdamW, NAdam, RAdam, RMSProp, SGD, auto) (default: AdamW)
```

# Known issues

- Sometimes the 'Widget metrics' iteration reporting has issues in ordering and as such, the reported scalars might appear incorrect. This occurs, since sometimes a reported value might have been placed in the wrong iteration *(possibly race condition? upload delay?)* by ClearML and therefor the scalar plot will show a value dropping. This is generally incorrect, since widget count can only increase with each iteration. The issue originates in the reporting mechanism of ClearML and can't be fixed in this source. The issue is reported here: [ClearML issue 1265]

# License

This project is licensed under the MIT License - see the LICENSE file for details.

# UI Randomizer

The UI randomizer is a python script for repeatedly calling the UI generator to create a desired amount of user interfaces with annotations.
The script will process the created annotation files and organize them in a folder structure suitable for a YOLO dataset.

# Pre-requisites & Installation

This project uses Poetry for managing dependencies.

The `ui_randomizer.py` script requires the LVGL UI Generator v1 source in order to work. Further information about that generator can be found in its README.

The `ui_randomizer_v2.py` script requires the LVGL UI Generator v2 source in order to work. Further information about that generator can be found in its README.

Since there are two versions of the generator, there are also two versions of the randomizer script. The first version is `ui_randomizer.py` and the second version is `ui_randomizer_v2.py`.

## Setting up the virtual environment

1. Install `poetry` package manager. See corresponding documentation for more information.
2. Run `poetry install` to install the dependencies and prepare the virtual environment.

# Usage

**Randomizer v1:**

```
poetry run python src/ui_randomizer.py <arguments>
```

▶ Randomizer v1 Help

```
usage: ui_randomizer.py [-h] -p APP_PATH [-i ITERATIONS] -t WIDGET_TYPES [WIDGET_TYPES ...] [--width WIDTH] [--height HEIGHT] -o OUTPUT_FOLDER
                        [-d DELAY_COUNT] [--split_widgets] [-l LAYOUT] [-r SPLIT_RATIO] (-s | -m MULTI)

Capture UI and create image and annotation with correct folders.

options:
  -h, --help            show this help message and exit
  -p APP_PATH, --app_path APP_PATH
                        Path to the random UI generator binary
  -i ITERATIONS, --iterations ITERATIONS
                        Number of UIs to generate
  -t WIDGET_TYPES [WIDGET_TYPES ...], --widget_types WIDGET_TYPES [WIDGET_TYPES ...]
                        List of widgets to be used in the UI
  --width WIDTH         Width of the UI screenshot
  --height HEIGHT       Height of the UI screenshot
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Folder to save the output images
  -d DELAY_COUNT, --delay_count DELAY_COUNT
                        Amount of times the timer handler shall be called with a fixed delay before capturing the UI
  --split_widgets       Split widgets into subfolders (only creates one widget type per iteration)
  -l LAYOUT, --layout LAYOUT
                        Path to the layout file to be used
  -r SPLIT_RATIO, --split_ratio SPLIT_RATIO
                        Split ratio for train, val, test
  -s, --single          Create only a single widget per iteration
  -m MULTI, --multi MULTI
                        Create multiple widgets per iteration
```

**Randomizer v2:**

```
poetry run python src/ui_randomizer_v2.py <arguments>
```

▶ Randomizer v2 Help

```
usage: ui_randomizer_v2.py [-h] [-mpy MICROPYTHON] [-m MAIN] -o OUTPUT_FOLDER [-r SPLIT_RATIO] [--datalist DATALIST] [-cwd WORKING_DIR] [--clean] [-d DELAY]
                           [--dataset DATASET] [--continue_on_error] [--capture_output] [--normalize] [-v] [--clearml_project CLEARML_PROJECT]
```

```
                          [--clearml_task CLEARML_TASK] [--
clearml_run_as_task] [--clearml_upload] [--normalize_bbox] [--
replace_class_names]
                          {random,design} ...

Invoke the generator and structure the captured UI images into a dataset

positional arguments:
  {random,design}       Generator options
    random              Random UI generator options
    design              Design file generator options

options:
  -h, --help            show this help message and exit
  -mpy MICROPYTHON, --micropython MICROPYTHON
                        Path to the micropython binary
  -m MAIN, --main MAIN  Path to the main script
  -o OUTPUT_FOLDER, --output_folder OUTPUT_FOLDER
                        Folder to save the output images

options:
  Additional options

  -r SPLIT_RATIO, --split_ratio SPLIT_RATIO
                        Split ratio for train, val, test
  --datalist DATALIST   Create a textfile with provided name to write all
images and labels
  -cwd WORKING_DIR, --working_dir WORKING_DIR
                        Working directory for the generator
  --clean               Clean the output folder before generating new data
  -d DELAY, --delay DELAY
                        Fixed delay between each generator call in
milliseconds
  --dataset DATASET     Name of the dataset
  --continue_on_error   Continue running the generator even if an error
occurs
  --capture_output      Capture the output of the generator
  --normalize           Activate normalize functionality of the generator
  -v, --verbose         Enable verbose output

clearml:
  Options for working with ClearML

  --clearml_project CLEARML_PROJECT
                        ClearML Dataset project name
  --clearml_task CLEARML_TASK
                        ClearML Dataset task name prefix
  --clearml_run_as_task
                        Run the randomizer as a ClearML task
  --clearml_upload      Upload the created dataset to ClearML

fixes:
  Annotation fixes

  --normalize_bbox      Post-process annotation files to normalize bounding
boxes
```

```
53  --replace_class_names
54                        Replace class names with their index in annotations
```

# Randomizer v1

The v1 generator is written in C, so prior compilation of the binary is required. More information about this setup be found in the LVGL UI Generator v1 README

The randomizer script will call the generator binary with the provided arguments and save the generated UIs in the specified output folder.

The size of the dataset is set by the `--iterations` argument. The script will call the generator the specified amount of times.

By supplying the `--delay_count` argument, you can set the amount of times the timer handler shall be called with a fixed delay before capturing the UI. This is useful for fixing issues with user interfaces not being fully rendered before capturing.

The script will rename and move the generated images and annotations to a folder structure suitable for a YOLO dataset.

By supplying the `--split_widgets` flag, the script will only generate screenshots containing a single widget type per iteration.

You can also specifiy the layout in which widgets will be structured by providing the `--layout` argument. The argument can be either `grid`, `flex`, or `none`.

If you only want to have a single widget per iteration, you can provide the `--single` flag. If you want to have multiple widgets per iteration, you can provide the `--multi` flag followed by the number of widgets you want to have.

The types of widgets used in the generator is specified by the `--widget_types` argument. The argument should be a list of widget types separated by spaces.

# Randomizer v2

The v2 generator is written in micropython, so prior compilation of the micropython binary is required. More information about this setup be found in the LVGL UI Generator v2 README

The randomizer has optional annotation fixes that it can apply to the created annotation files. The fixes include normalizing bounding boxes (`--normalize_bbox`) and replacing class names (`--replace_class_names`) with their index in the widget list.

These fixes are used by providing their respective flags in the command line arguments.

Furthermore, you can adjust the split ratio for the dataset by providing the `--split_ratio` argument. The argument should be a string of three numbers separated by commas. The numbers represent the ratio for the train, validation, and test sets respectively.

Since calling the generator can produce a lot of output, it is disabled by default. You can enable printing to the console by providing the `--capture_output` flag.

The script and generator may also produce errors during the generation process and will abort by default. You can choose to continue running if such events occur by providing the `--continue_on_error` flag.

The output of the script produces a dataset, but you can activate an additional datalist output by providing the `--datalist` argument. The argument should be the name of the text file that will be created in the output folder. The file will contain the paths to the images and annotations in the dataset in the form of `image_path annotation_path`.

By default, the script will error if the provided output folder is not empty. You can choose to clean the folder before generating new data by providing the `--clean` flag. Be aware that this will delete all files in the output folder.

To slow down the generation process, you can provide a fixed delay between each generator call in milliseconds by providing the `--delay` argument. This can be useful for fixing issues with user interfaces not being fully rendered before capturing, or to simply watch the generation process.

## Modes

The randomizer v2 script supports both random and design modes of the generator.

For details about these modes, see the LVGL UI Generator v2 README.

▶ Help for random mode

```
usage: ui_randomizer_v2.py random [-h] -t WIDGET_TYPES [WIDGET_TYPES ...] [-W WIDTH] [-H HEIGHT] [--split_widgets] [-c COUNT] [-l LAYOUT] [-i ITERATIONS]

options:
  -h, --help            show this help message and exit
  -t WIDGET_TYPES [WIDGET_TYPES ...], --widget_types WIDGET_TYPES [WIDGET_TYPES ...]
                        List of widgets to be used in the UI
  -W WIDTH, --width WIDTH
                        Width of the UI screenshot
  -H HEIGHT, --height HEIGHT
                        Height of the UI screenshot
```

```
  --split_widgets       Split widgets into subfolders (only creates one
widget type per iteration)
  -c COUNT, --count COUNT
                        Number of widgets to create per iteration
  -l LAYOUT, --layout LAYOUT
                        The main container layout of the random UI ["grid",
"flex", "none"]
  -i ITERATIONS, --iterations ITERATIONS
                        Number of UIs to generate
```

▶ Help for design mode

```
usage: ui_randomizer_v2.py design [-h] [-f DESIGN_FOLDER]

options:
  -h, --help            show this help message and exit
  -f DESIGN_FOLDER, --design_folder DESIGN_FOLDER
                        Folder containing the design files
```

# ClearML Integration

The randomizer v2 script has integration with ClearML to directly upload the created dataset to a ClearML project.

To use this feature, you need to provide the `--clearml_upload` flag. You can also specify the ClearML project name with the `--clearml_project` argument. Additionally, you can run the generation process as a ClearML task by providing the `--clearml_run_as_task` flag. This will run the script as a ClearML task, which is useful for tracking & storing the progress of the generation process.
You may also specify the task name prefix with the `--clearml_task` argument.

# Known issues

# License

This project is licensed under the MIT License - see the LICENSE file for details.

# LVGL UI Generator

This forked project has been archived in favor of the newer and much simpler generator based on lv_micropython.

# Simulator project for LVGL embedded GUI Library

The LVGL is written mainly for microcontrollers and embedded systems however you can run the library **on your PC** as well without any embedded hardware. The code written on PC can be simply copied when your are using an embedded system.

Using a PC simulator instead of an embedded hardware has several advantages:

- **Costs $0** because you don't have to buy or design PCB
- **Fast** because you don't have to design and manufacture PCB
- **Collaborative** because any number of developers can work in the same environment
- **Developer friendly** because much easier and faster to debug on PC

## Requirements

This project is configured for VSCode and only tested on Linux, although this may work on OSx or WSL. It requires a working version of GCC, GDB and make in your path.

To allow debugging inside VSCode you will also require a GDB extension or other suitable debugger. All the requirements have been pre-configured in the .workspace file (simply open the project by doubleclick on this file).

The project can use **SDL** or **X11** as LVGL display driver for lowlevel graphics/mouse/keyboard support. This can be defined in the Makefile.
Please make sure the selected library is installed in the system (check Install graphics driver).

## Usage

### Get the PC project

Clone the PC project and the related sub modules:

```
git clone --recursive https://github.com/lvgl/lv_port_pc_vscode
```

### Install graphics driver

The project can use **SDL** or **X11** as LVGL display driver. This can be selected in the Makefile.
Please make sure the used library is installed in the system:

#### Install SDL

You can download SDL from https://www.libsdl.org/

On on Linux you can install it via terminal:

```
sudo apt-get update && sudo apt-get install -y build-essential libsdl2-dev
```

### Install X11

On on Linux you can install it via terminal:

```
sudo apt-get update && sudo apt-get install -y libx11-dev
```

## Optional library

There are also FreeType and FFmpeg support. You can install FreeType support with:

```
# FreeType support
wget https://kumisystems.dl.sourceforge.net/project/freetype/freetype2/2.13.2/freetype-2.13.2.tar.xz
tar -xf freetype-2.13.2.tar.xz
cd freetype-2.13.2
make
make install
```

The FFmpeg support can be installed with:

```
# FFmpeg support
git clone https://git.ffmpeg.org/ffmpeg.git ffmpeg
cd ffmpeg
git checkout release/6.0
./configure --disable-all --disable-autodetect --disable-podpages --disable-asm --enable-avcodec --enable-avformat --enable-decoders --enable-encoders --enable-demuxers --enable-parsers --enable-protocol='file' --enable-swscale --enable-zlib
make
sudo make install
```

And then remove all the comments in the `Makefile` on `INC` and `LDLIBS` lines. \
They should be for **SDL**:

```
INC    := -I./ui/simulator/inc/ -I./ -I./lvgl/ -I/usr/include/freetype2 -L/usr/local/lib
LDLIBS := -lSDL2 -lm -lfreetype -lavformat -lavcodec -lavutil -lswscale -lm -lz -lpthread
```

They should be for **X11**:

```
INC    := -I./ui/simulator/inc/ -I./ -I./lvgl/ -I/usr/include/freetype2 -L/usr/local/lib
LDLIBS := -lX11 -lm -lfreetype -lavformat -lavcodec -lavutil -lswscale -lm -lz -lpthread
```

## Setup

To allow custom UI code an `lv_conf.h` file placed at `ui/simulator/inc` will automatically override this projects lv_conf.h file. By default code under `ui` is ignored so you can reuse this repository for multiple projects. You will need to place a call from `main.c` to your UI's entry function.

To build and debug, press F5. You should now have your UI displayed in a new window and can access all the debug features of VSCode through GDB.

To allow temporary modification between simulator and device code, a SIMULATOR=1 define is added globally.

# License

This project is licensed under the MIT License - see the LICENSE file for details.

# LVGL UI Generator v2

A project to generate a user interface using LVGL and capturing a screenshot alongside an annotation file with widget metadata (bounding box).

LVGL UI Generator version 2 is an updated version of the UI generator that uses micropython and corresponding LVGL bindings as a base. This version is more flexible, easier to use and more maintenable than the original version.

It has two modes of operation:

- **Random mode**: Generates a random UI with a specified number of widgets placed on a white background. It requires a provided list of widget types to randomly choose from.
- **Design mode**: Generates a UI based on a provided JSON design file. The design file describes the whole window, including styles, widgets and certain properties. There is a special `random` widget, which can be used to randomize widget creation in certain areas of the design. This mode is useful in creating more realistic looking user interfaces, as the random mode does not accomodate for styles regarding the containers.

# Prerequisites & Installation

In order to run the UI generator, you need to compile the micropython binary with the LVGL bindings. To make this process easier, the project has a `tasks.py` file, which already contains necessary routines via the usage of `invoke`.

To use the `invoke` package, you will need to setup a virtual environment and install the dependencies using the provided poetry project file (`pyproject.toml`).

Since `lv_micropython` is included as a submodule, you will need to initialize the submodules **before** running the build task.

⚠️ **Be aware, that initializing this submodule can take quite a while to complete, due to all the additional source dependencies being downloaded.** *(including unnecessary sources for various MCUs, ports and architectures)*

As of yet, there is no way to speed this up, but it is generally a one-time operation.

## Initializing the micropython submodule

Run the following command to initialize the submodule:

```
git submodule update --init --recursive
```

Make sure to grab a cup of coffee or tea ☕, as this operation can take quite a while to complete.

## Setting up the virtual environment

1. Install `poetry` package manager. See corresponding documentation for more information.
2. Run `poetry install` to install the dependencies.

## Compiling the micropython binary

Run `poetry run invoke build` to compile the micropython binary with the LVGL bindings, using the provided `lv_conf.h` file.

▶ Example build output

```
$ poetry run inv build
make: Entering directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/mpy-cross'
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
GEN build/genhdr/mpversion.h
CC ../py/modsys.c
CC main.c
LINK build/mpy-cross
   text    data     bss     dec     hex filename
 305806   13856     856  320518   4e406 build/mpy-cross
make: Leaving directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/mpy-cross'
make: Entering directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/ports/unix'
```

```
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
Updating submodules: lib/mbedtls lib/berkeley-db-1.xx lib/micropython-lib
Synchronizing submodule url for '../../lib/berkeley-db-1.xx'
Synchronizing submodule url for '../../lib/mbedtls'
Synchronizing submodule url for '../../lib/micropython-lib'
make: Leaving directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/ports/unix'
make: Entering directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/ports/unix'
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
LVGL-GEN build-standard/lvgl/lv_mpy.c
GEN build-standard/genhdr/mpversion.h
GEN build-standard/genhdr/qstr.i.last
GEN build-standard/genhdr/qstr.split
GEN build-standard/genhdr/moduledefs.split
GEN build-standard/genhdr/root_pointers.split
GEN build-standard/genhdr/compressed.split
GEN build-standard/genhdr/root_pointers.collected
GEN build-standard/genhdr/qstrdefs.collected.h
GEN build-standard/genhdr/moduledefs.collected
Root pointer registrations not updated
GEN build-standard/genhdr/compressed.collected
Module registrations not updated
QSTR not updated
Compressed data not updated
CC ../../py/modsys.c
CC ../../extmod/moduplatform.c
CC build-standard/lvgl/lv_mpy.c
CC ../../lib/lv_bindings/lvgl/src/drivers/evdev/lv_evdev.c
CC ../../lib/lv_bindings/lvgl/src/drivers/windows/lv_windows_input.c
CC ../../lib/lv_bindings/lvgl/src/drivers/windows/lv_windows_display.c
CC ../../lib/lv_bindings/lvgl/src/drivers/windows/lv_windows_context.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/st7735/lv_st7735.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/fb/lv_linux_fbdev.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/ili9341/lv_ili9341.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/drm/lv_linux_drm.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/st7796/lv_st7796.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/st7789/lv_st7789.c
CC ../../lib/lv_bindings/lvgl/src/drivers/display/lcd/lv_lcd_generic_mipi.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_lcd.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_libuv.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_fbdev.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_entry.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_profiler.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_touchscreen.c
CC ../../lib/lv_bindings/lvgl/src/drivers/nuttx/lv_nuttx_cache.c
CC ../../lib/lv_bindings/lvgl/src/drivers/x11/lv_x11_display.c
CC ../../lib/lv_bindings/lvgl/src/drivers/x11/lv_x11_input.c
CC ../../lib/lv_bindings/lvgl/src/drivers/sdl/lv_sdl_window.c
CC ../../lib/lv_bindings/lvgl/src/drivers/sdl/lv_sdl_mouse.c
CC ../../lib/lv_bindings/lvgl/src/drivers/sdl/lv_sdl_keyboard.c
CC ../../lib/lv_bindings/lvgl/src/drivers/sdl/lv_sdl_mousewheel.c
CC ../../lib/lv_bindings/lvgl/src/themes/default/lv_theme_default.c
CC ../../lib/lv_bindings/lvgl/src/themes/lv_theme.c
```

```
CC ../../lib/lv_bindings/lvgl/src/themes/simple/lv_theme_simple.c
CC ../../lib/lv_bindings/lvgl/src/themes/mono/lv_theme_mono.c
CC ../../lib/lv_bindings/lvgl/src/tick/lv_tick.c
CC ../../lib/lv_bindings/lvgl/src/lv_init.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_pthread.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_cmsis_rtos2.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_windows.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_os_none.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_rtthread.c
CC ../../lib/lv_bindings/lvgl/src/osal/lv_freertos.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_class.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_id_builtin.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_scroll.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_style.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_event.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_refr.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_group.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_pos.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_style_gen.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_tree.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_property.c
CC ../../lib/lv_bindings/lvgl/src/core/lv_obj_draw.c
CC ../../lib/lv_bindings/lvgl/src/others/sysmon/lv_sysmon.c
CC ../../lib/lv_bindings/lvgl/src/others/imgfont/lv_imgfont.c
CC ../../lib/lv_bindings/lvgl/src/others/file_explorer/lv_file_explorer.c
CC ../../lib/lv_bindings/lvgl/src/others/observer/lv_observer.c
CC ../../lib/lv_bindings/lvgl/src/others/snapshot/lv_snapshot.c
CC ../../lib/lv_bindings/lvgl/src/others/monkey/lv_monkey.c
CC ../../lib/lv_bindings/lvgl/src/others/fragment/lv_fragment.c
CC ../../lib/lv_bindings/lvgl/src/others/fragment/lv_fragment_manager.c
CC ../../lib/lv_bindings/lvgl/src/others/gridnav/lv_gridnav.c
CC ../../lib/lv_bindings/lvgl/src/others/ime/lv_ime_pinyin.c
CC ../../lib/lv_bindings/lvgl/src/others/vg_lite_tvg/vg_lite_matrix.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/rtthread/lv_string_rtthread.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/rtthread/lv_sprintf_rtthread.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/rtthread/lv_mem_core_rtthread.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/lv_mem.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/clib/lv_string_clib.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/clib/lv_mem_core_clib.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/clib/lv_sprintf_clib.c
CC
../../lib/lv_bindings/lvgl/src/stdlib/micropython/lv_mem_core_micropython.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/builtin/lv_sprintf_builtin.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/builtin/lv_tlsf.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/builtin/lv_mem_core_builtin.c
CC ../../lib/lv_bindings/lvgl/src/stdlib/builtin/lv_string_builtin.c
CC ../../lib/lv_bindings/lvgl/src/misc/cache/lv_cache_entry.c
CC ../../lib/lv_bindings/lvgl/src/misc/cache/lv_image_cache.c
CC ../../lib/lv_bindings/lvgl/src/misc/cache/_lv_cache_lru_rb.c
CC ../../lib/lv_bindings/lvgl/src/misc/cache/lv_cache.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_profiler_builtin.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_color_op.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_color.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_text.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_bidi.c
```

```
CC ../../lib/lv_bindings/lvgl/src/misc/lv_style_gen.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_async.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_palette.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_style.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_text_ap.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_array.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_lru.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_anim.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_rb.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_math.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_fs.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_timer.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_log.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_event.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_ll.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_area.c
CC ../../lib/lv_bindings/lvgl/src/misc/lv_anim_timeline.c
CC ../../lib/lv_bindings/lvgl/src/layouts/flex/lv_flex.c
CC ../../lib/lv_bindings/lvgl/src/layouts/grid/lv_grid.c
CC ../../lib/lv_bindings/lvgl/src/layouts/lv_layout.c
CC ../../lib/lv_bindings/lvgl/src/libs/fsdrv/lv_fs_stdio.c
CC ../../lib/lv_bindings/lvgl/src/libs/fsdrv/lv_fs_memfs.c
CC ../../lib/lv_bindings/lvgl/src/libs/fsdrv/lv_fs_fatfs.c
CC ../../lib/lv_bindings/lvgl/src/libs/fsdrv/lv_fs_posix.c
CC ../../lib/lv_bindings/lvgl/src/libs/fsdrv/lv_fs_win32.c
CC ../../lib/lv_bindings/lvgl/src/libs/bin_decoder/lv_bin_decoder.c
CC ../../lib/lv_bindings/lvgl/src/libs/rlottie/lv_rlottie.c
CC ../../lib/lv_bindings/lvgl/src/libs/libpng/lv_libpng.c
CC ../../lib/lv_bindings/lvgl/src/libs/tiny_ttf/lv_tiny_ttf.c
CC ../../lib/lv_bindings/lvgl/src/libs/barcode/code128.c
CC ../../lib/lv_bindings/lvgl/src/libs/barcode/lv_barcode.c
CC ../../lib/lv_bindings/lvgl/src/libs/rle/lv_rle.c
CC ../../lib/lv_bindings/lvgl/src/libs/lz4/lz4.c
CC ../../lib/lv_bindings/lvgl/src/libs/bmp/lv_bmp.c
CC ../../lib/lv_bindings/lvgl/src/libs/lodepng/lv_lodepng.c
CC ../../lib/lv_bindings/lvgl/src/libs/lodepng/lodepng.c
CC ../../lib/lv_bindings/lvgl/src/libs/tjpgd/lv_tjpgd.c
CC ../../lib/lv_bindings/lvgl/src/libs/gif/gifdec.c
CC ../../lib/lv_bindings/lvgl/src/libs/gif/lv_gif.c
CC ../../lib/lv_bindings/lvgl/src/libs/qrcode/qrcodegen.c
CC ../../lib/lv_bindings/lvgl/src/libs/qrcode/lv_qrcode.c
CC ../../lib/lv_bindings/lvgl/src/libs/freetype/lv_freetype_glyph.c
CC ../../lib/lv_bindings/lvgl/src/libs/freetype/lv_freetype_image.c
CC ../../lib/lv_bindings/lvgl/src/libs/freetype/lv_ftsystem.c
CC ../../lib/lv_bindings/lvgl/src/libs/freetype/lv_freetype_outline.c
CC ../../lib/lv_bindings/lvgl/src/libs/freetype/lv_freetype.c
CC ../../lib/lv_bindings/lvgl/src/libs/libjpeg_turbo/lv_libjpeg_turbo.c
CC ../../lib/lv_bindings/lvgl/src/libs/ffmpeg/lv_ffmpeg.c
CC ../../lib/lv_bindings/lvgl/src/display/lv_display.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_8.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_20.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_30.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_44.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_18.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_unscii_8.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font.c
```

```
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_simsun_16_cjk.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_38.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_22.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_fmt_txt.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_32.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_dejavu_16_persian_hebrew.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_binfont_loader.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_28.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_42.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_unscii_16.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_28_compressed.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_36.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_40.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_26.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_34.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_16.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_24.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_48.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_46.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_12.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_14.c
CC ../../lib/lv_bindings/lvgl/src/font/lv_font_montserrat_10.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_image.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_triangle.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_line.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_label.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_mask_rect.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_box_shadow.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_gradient.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_mask.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_triangle.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_transform.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_letter.c
CC
../../lib/lv_bindings/lvgl/src/draw/sw/blend/lv_draw_sw_blend_to_argb8888.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/blend/lv_draw_sw_blend.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/blend/lv_draw_sw_blend_to_rgb888.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/blend/lv_draw_sw_blend_to_rgb565.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_arc.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_vector.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_border.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_fill.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_line.c
CC ../../lib/lv_bindings/lvgl/src/draw/sw/lv_draw_sw_img.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_image_decoder.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_vector.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_rect.c
CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_arc.c
CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_line.c
CC
../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_border.c
CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_arc.c
CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d.c
CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_label.c
CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_fill.c
```

```
229 CC
    ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_triangle.c
230 CC
    ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_mask_recta
    ngle.c
231 CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_utils.c
232 CC ../../lib/lv_bindings/lvgl/src/draw/renesas/dave2d/lv_draw_dave2d_image.c
233 CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_mask.c
234 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_draw_pxp_img.c
235 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_draw_pxp_layer.c
236 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_pxp_osa.c
237 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_pxp_cfg.c
238 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_draw_buf_pxp.c
239 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_draw_pxp.c
240 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_draw_pxp_fill.c
241 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/pxp/lv_pxp_utils.c
242 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_fill.c
243 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_vglite_path.c
244 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_border.c
245 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_buf_vglite.c
246 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_img.c
247 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_layer.c
248 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_line.c
249 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_arc.c
250 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_vglite_utils.c
251 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_vglite_matrix.c
252 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_vglite_buf.c
253 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_triangle.c
254 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite.c
255 CC ../../lib/lv_bindings/lvgl/src/draw/nxp/vglite/lv_draw_vglite_label.c
256 CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw_buf.c
257 CC ../../lib/lv_bindings/lvgl/src/draw/lv_image_buf.c
258 CC ../../lib/lv_bindings/lvgl/src/draw/sdl/lv_draw_sdl.c
259 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_vg_lite_utils.c
260 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_mask_rect.c
261 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_arc.c
262 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_layer.c
263 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_border.c
264 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_buf_vg_lite.c
265 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_img.c
266 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite.c
267 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_vg_lite_path.c
268 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_line.c
269 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_label.c
270 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_vg_lite_decoder.c
271 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_box_shadow.c
272 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_fill.c
273 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_vector.c
274 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_draw_vg_lite_triangle.c
275 CC ../../lib/lv_bindings/lvgl/src/draw/vg_lite/lv_vg_lite_math.c
276 CC ../../lib/lv_bindings/lvgl/src/draw/lv_draw.c
277 CC ../../lib/lv_bindings/lvgl/src/indev/lv_indev.c
278 CC ../../lib/lv_bindings/lvgl/src/indev/lv_indev_scroll.c
279 CC ../../lib/lv_bindings/lvgl/src/widgets/dropdown/lv_dropdown.c
280 CC ../../lib/lv_bindings/lvgl/src/widgets/arc/lv_arc.c
281 CC ../../lib/lv_bindings/lvgl/src/widgets/keyboard/lv_keyboard.c
```

```
CC ../../lib/lv_bindings/lvgl/src/widgets/line/lv_line.c
CC ../../lib/lv_bindings/lvgl/src/widgets/scale/lv_scale.c
CC ../../lib/lv_bindings/lvgl/src/widgets/switch/lv_switch.c
CC ../../lib/lv_bindings/lvgl/src/widgets/animimage/lv_animimage.c
CC ../../lib/lv_bindings/lvgl/src/widgets/slider/lv_slider.c
CC ../../lib/lv_bindings/lvgl/src/widgets/canvas/lv_canvas.c
CC ../../lib/lv_bindings/lvgl/src/widgets/button/lv_button.c
CC ../../lib/lv_bindings/lvgl/src/widgets/checkbox/lv_checkbox.c
CC ../../lib/lv_bindings/lvgl/src/widgets/span/lv_span.c
CC ../../lib/lv_bindings/lvgl/src/widgets/spinner/lv_spinner.c
CC ../../lib/lv_bindings/lvgl/src/widgets/imagebutton/lv_imagebutton.c
CC ../../lib/lv_bindings/lvgl/src/widgets/roller/lv_roller.c
CC ../../lib/lv_bindings/lvgl/src/widgets/tabview/lv_tabview.c
CC ../../lib/lv_bindings/lvgl/src/widgets/label/lv_label.c
CC ../../lib/lv_bindings/lvgl/src/widgets/menu/lv_menu.c
CC ../../lib/lv_bindings/lvgl/src/widgets/textarea/lv_textarea.c
CC ../../lib/lv_bindings/lvgl/src/widgets/tileview/lv_tileview.c
CC ../../lib/lv_bindings/lvgl/src/widgets/image/lv_image.c
CC ../../lib/lv_bindings/lvgl/src/widgets/bar/lv_bar.c
CC ../../lib/lv_bindings/lvgl/src/widgets/buttonmatrix/lv_buttonmatrix.c
CC ../../lib/lv_bindings/lvgl/src/widgets/chart/lv_chart.c
CC ../../lib/lv_bindings/lvgl/src/widgets/msgbox/lv_msgbox.c
CC ../../lib/lv_bindings/lvgl/src/widgets/list/lv_list.c
CC ../../lib/lv_bindings/lvgl/src/widgets/spinbox/lv_spinbox.c
CC ../../lib/lv_bindings/lvgl/src/widgets/win/lv_win.c
CC
../../lib/lv_bindings/lvgl/src/widgets/calendar/lv_calendar_header_arrow.c
CC
../../lib/lv_bindings/lvgl/src/widgets/calendar/lv_calendar_header_dropdown.
c
CC ../../lib/lv_bindings/lvgl/src/widgets/calendar/lv_calendar.c
CC ../../lib/lv_bindings/lvgl/src/widgets/led/lv_led.c
CC ../../lib/lv_bindings/lvgl/src/widgets/table/lv_table.c
CC ../../lib/lv_bindings/lvgl/examples/anim/lv_example_anim_2.c
CC ../../lib/lv_bindings/lvgl/examples/anim/lv_example_anim_1.c
CC ../../lib/lv_bindings/lvgl/examples/anim/lv_example_anim_timeline_1.c
CC ../../lib/lv_bindings/lvgl/examples/anim/lv_example_anim_3.c
CC ../../lib/lv_bindings/lvgl/examples/others/imgfont/lv_example_imgfont_1.c
CC
../../lib/lv_bindings/lvgl/examples/others/file_explorer/lv_example_file_exp
lorer_3.c
CC
../../lib/lv_bindings/lvgl/examples/others/file_explorer/lv_example_file_exp
lorer_1.c
CC
../../lib/lv_bindings/lvgl/examples/others/file_explorer/lv_example_file_exp
lorer_2.c
CC
../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_2.c
CC
../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_5.c
CC
../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_3.c
CC
../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_4.c
```

```
CC ../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_6.c
CC ../../lib/lv_bindings/lvgl/examples/others/observer/lv_example_observer_1.c
CC ../../lib/lv_bindings/lvgl/examples/others/snapshot/lv_example_snapshot_1.c
CC ../../lib/lv_bindings/lvgl/examples/others/monkey/lv_example_monkey_1.c
CC ../../lib/lv_bindings/lvgl/examples/others/monkey/lv_example_monkey_2.c
CC ../../lib/lv_bindings/lvgl/examples/others/monkey/lv_example_monkey_3.c
CC ../../lib/lv_bindings/lvgl/examples/others/fragment/lv_example_fragment_1.c
CC ../../lib/lv_bindings/lvgl/examples/others/fragment/lv_example_fragment_2.c
CC ../../lib/lv_bindings/lvgl/examples/others/gridnav/lv_example_gridnav_1.c
CC ../../lib/lv_bindings/lvgl/examples/others/gridnav/lv_example_gridnav_4.c
CC ../../lib/lv_bindings/lvgl/examples/others/gridnav/lv_example_gridnav_3.c
CC ../../lib/lv_bindings/lvgl/examples/others/gridnav/lv_example_gridnav_2.c
CC ../../lib/lv_bindings/lvgl/examples/others/ime/lv_example_ime_pinyin_2.c
CC ../../lib/lv_bindings/lvgl/examples/others/ime/lv_example_ime_pinyin_1.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_8.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_11.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_9.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_1.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_13.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_5.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_3.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_6.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_12.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_10.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_14.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_15.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_7.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_2.c
CC ../../lib/lv_bindings/lvgl/examples/styles/lv_example_style_4.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_star.c
CC ../../lib/lv_bindings/lvgl/examples/assets/imgbtn_mid.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_hand.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_caret_down.c
CC ../../lib/lv_bindings/lvgl/examples/assets/animimg002.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_skew_strip.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_cogwheel_rgb.c
CC ../../lib/lv_bindings/lvgl/examples/assets/animimg001.c
CC ../../lib/lv_bindings/lvgl/examples/assets/imgbtn_right.c
CC ../../lib/lv_bindings/lvgl/examples/assets/animimg003.c
CC ../../lib/lv_bindings/lvgl/examples/assets/imgbtn_left.c
CC ../../lib/lv_bindings/lvgl/examples/assets/emoji/img_emoji_F617.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_cogwheel_indexed16.c
CC ../../lib/lv_bindings/lvgl/examples/assets/img_cogwheel_argb.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_2.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_4.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_3.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_6.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_1.c
CC ../../lib/lv_bindings/lvgl/examples/scroll/lv_example_scroll_5.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_3.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_1.c
```

```
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_4.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_2.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_6.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/flex/lv_example_flex_5.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_5.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_2.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_1.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_4.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_6.c
CC ../../lib/lv_bindings/lvgl/examples/layouts/grid/lv_example_grid_3.c
CC ../../lib/lv_bindings/lvgl/examples/libs/rlottie/lv_example_rlottie_2.c
CC ../../lib/lv_bindings/lvgl/examples/libs/rlottie/lv_example_rlottie_1.c
CC
../../lib/lv_bindings/lvgl/examples/libs/rlottie/lv_example_rlottie_approve.
c
CC ../../lib/lv_bindings/lvgl/examples/libs/libpng/lv_example_libpng_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/tiny_ttf/lv_example_tiny_ttf_3.c
CC ../../lib/lv_bindings/lvgl/examples/libs/tiny_ttf/lv_example_tiny_ttf_2.c
CC ../../lib/lv_bindings/lvgl/examples/libs/tiny_ttf/ubuntu_font.c
CC ../../lib/lv_bindings/lvgl/examples/libs/tiny_ttf/lv_example_tiny_ttf_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/barcode/lv_example_barcode_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/bmp/lv_example_bmp_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/lodepng/lv_example_lodepng_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/lodepng/img_wink_png.c
CC ../../lib/lv_bindings/lvgl/examples/libs/tjpgd/lv_example_tjpgd_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/gif/img_bulb_gif.c
CC ../../lib/lv_bindings/lvgl/examples/libs/gif/lv_example_gif_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/qrcode/lv_example_qrcode_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/freetype/lv_example_freetype_1.c
CC
../../lib/lv_bindings/lvgl/examples/libs/libjpeg_turbo/lv_example_libjpeg_tu
rbo_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/ffmpeg/lv_example_ffmpeg_1.c
CC ../../lib/lv_bindings/lvgl/examples/libs/ffmpeg/lv_example_ffmpeg_2.c
CC
../../lib/lv_bindings/lvgl/examples/get_started/lv_example_get_started_1.c
CC
../../lib/lv_bindings/lvgl/examples/get_started/lv_example_get_started_3.c
CC
../../lib/lv_bindings/lvgl/examples/get_started/lv_example_get_started_4.c
CC
../../lib/lv_bindings/lvgl/examples/get_started/lv_example_get_started_2.c
CC ../../lib/lv_bindings/lvgl/examples/event/lv_example_event_1.c
CC ../../lib/lv_bindings/lvgl/examples/event/lv_example_event_2.c
CC ../../lib/lv_bindings/lvgl/examples/event/lv_example_event_4.c
CC ../../lib/lv_bindings/lvgl/examples/event/lv_example_event_3.c
CC
../../lib/lv_bindings/lvgl/examples/widgets/dropdown/lv_example_dropdown_3.c
CC
../../lib/lv_bindings/lvgl/examples/widgets/dropdown/lv_example_dropdown_1.c
CC
../../lib/lv_bindings/lvgl/examples/widgets/dropdown/lv_example_dropdown_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/arc/lv_example_arc_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/arc/lv_example_arc_1.c
CC
../../lib/lv_bindings/lvgl/examples/widgets/keyboard/lv_example_keyboard_1.c
```

```
419 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/keyboard/lv_example_keyboard_2.c
420 CC ../../lib/lv_bindings/lvgl/examples/widgets/line/lv_example_line_1.c
421 CC ../../lib/lv_bindings/lvgl/examples/widgets/scale/lv_example_scale_5.c
422 CC ../../lib/lv_bindings/lvgl/examples/widgets/scale/lv_example_scale_2.c
423 CC ../../lib/lv_bindings/lvgl/examples/widgets/scale/lv_example_scale_1.c
424 CC ../../lib/lv_bindings/lvgl/examples/widgets/scale/lv_example_scale_4.c
425 CC ../../lib/lv_bindings/lvgl/examples/widgets/scale/lv_example_scale_3.c
426 CC ../../lib/lv_bindings/lvgl/examples/widgets/switch/lv_example_switch_1.c
427 CC ../../lib/lv_bindings/lvgl/examples/widgets/slider/lv_example_slider_4.c
428 CC ../../lib/lv_bindings/lvgl/examples/widgets/slider/lv_example_slider_1.c
429 CC ../../lib/lv_bindings/lvgl/examples/widgets/slider/lv_example_slider_2.c
430 CC ../../lib/lv_bindings/lvgl/examples/widgets/slider/lv_example_slider_3.c
431 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_5.c
432 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_1.c
433 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_3.c
434 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_2.c
435 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_7.c
436 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_4.c
437 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_8.c
438 CC ../../lib/lv_bindings/lvgl/examples/widgets/canvas/lv_example_canvas_6.c
439 CC ../../lib/lv_bindings/lvgl/examples/widgets/button/lv_example_button_2.c
440 CC ../../lib/lv_bindings/lvgl/examples/widgets/button/lv_example_button_3.c
441 CC ../../lib/lv_bindings/lvgl/examples/widgets/button/lv_example_button_1.c
442 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/checkbox/lv_example_checkbox_2.c
443 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/checkbox/lv_example_checkbox_1.c
444 CC ../../lib/lv_bindings/lvgl/examples/widgets/span/lv_example_span_1.c
445 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/spinner/lv_example_spinner_1.c
446 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/imagebutton/lv_example_imagebutt
    on_1.c
447 CC ../../lib/lv_bindings/lvgl/examples/widgets/roller/lv_example_roller_3.c
448 CC ../../lib/lv_bindings/lvgl/examples/widgets/roller/lv_example_roller_2.c
449 CC ../../lib/lv_bindings/lvgl/examples/widgets/roller/lv_example_roller_1.c
450 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/tabview/lv_example_tabview_2.c
451 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/tabview/lv_example_tabview_1.c
452 CC ../../lib/lv_bindings/lvgl/examples/widgets/label/lv_example_label_3.c
453 CC ../../lib/lv_bindings/lvgl/examples/widgets/label/lv_example_label_4.c
454 CC ../../lib/lv_bindings/lvgl/examples/widgets/label/lv_example_label_5.c
455 CC ../../lib/lv_bindings/lvgl/examples/widgets/label/lv_example_label_2.c
456 CC ../../lib/lv_bindings/lvgl/examples/widgets/label/lv_example_label_1.c
457 CC ../../lib/lv_bindings/lvgl/examples/widgets/menu/lv_example_menu_4.c
458 CC ../../lib/lv_bindings/lvgl/examples/widgets/menu/lv_example_menu_5.c
459 CC ../../lib/lv_bindings/lvgl/examples/widgets/menu/lv_example_menu_2.c
460 CC ../../lib/lv_bindings/lvgl/examples/widgets/menu/lv_example_menu_3.c
461 CC ../../lib/lv_bindings/lvgl/examples/widgets/menu/lv_example_menu_1.c
462 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/textarea/lv_example_textarea_2.c
463 CC
    ../../lib/lv_bindings/lvgl/examples/widgets/textarea/lv_example_textarea_3.c
```

```
CC ../../lib/lv_bindings/lvgl/examples/widgets/textarea/lv_example_textarea_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/animimg/lv_example_animimg_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/tileview/lv_example_tileview_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/image/lv_example_image_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/image/lv_example_image_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/image/lv_example_image_4.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/image/lv_example_image_3.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/obj/lv_example_obj_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/obj/lv_example_obj_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_4.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_3.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_6.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_5.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/bar/lv_example_bar_7.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/buttonmatrix/lv_example_buttonmatrix_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/buttonmatrix/lv_example_buttonmatrix_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/buttonmatrix/lv_example_buttonmatrix_3.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_6.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_3.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_5.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_4.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_7.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/chart/lv_example_chart_8.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/msgbox/lv_example_msgbox_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/list/lv_example_list_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/list/lv_example_list_2.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/spinbox/lv_example_spinbox_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/win/lv_example_win_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/calendar/lv_example_calendar_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/led/lv_example_led_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/table/lv_example_table_1.c
CC ../../lib/lv_bindings/lvgl/examples/widgets/table/lv_example_table_2.c
CC main.c
LINK build-standard/micropython
   text    data     bss     dec     hex filename
1741466  225840    7472 1974778  1e21fa build-standard/micropython
make: Leaving directory '/home/rini-debian/git-stash/lvgl-ui-detector/lvgl_ui_generator_v2/lv_micropython/ports/unix'
```

# Usage

```
usage: src/main.py [-h] [-m, --mode mode] [-?, --usage] [-n, --normalize] [-o, --output_file output_file]

Process CLI arguments for the UI generator.

optional args:
  -h, --help                        show this message and exit
  -m, --mode mode                   the mode to run the program in
  -?, --usage                       Print usage information for that mode.
  -n, --normalize                   normalize the bounding boxes
  -o, --output_file output_file     The output file (screenshot)
```

# TL;DR

To quickly generate a user interface without prior knowledge of the CLI, use the following commands to copy & paste:

## Random mode

Run via `invoke`:

```
poetry run invoke generate-random
```

or via `poetry`:

```
poetry run micropython src/main.py -m random --normalize -o screenshot.jpg -W 640 -H 640 -c 4 -l none --random-state -t arc bar button buttonmatrix calendar checkbox dropdown label roller scale slider spinbox switch table textarea
```

or directly:

```
./lv_micropython/ports/unix/build-standard/micropython src/main.py -m random --normalize -o screenshot.jpg -W 640 -H 640 -c 4 -l none --random-state -t arc bar button buttonmatrix calendar checkbox dropdown label roller scale slider spinbox switch table textarea
```

## Design mode

Run via `invoke`:

```
poetry run invoke generate-design
```

or via `poetry`:

```
poetry run micropython src/main.py -m design --normalize -f ./designs/widgets_showcase.json -o screenshot.jpg
```

or directly:

```
./lv_micropython/ports/unix/build-standard/micropython src/main.py -m design --normalize -f ./designs/widgets_showcase.json -o screenshot.jpg
```

# Usage of random mode

```
usage: src/main.py [-h] [-m, --mode mode] [-?, --usage] [-n, --normalize] [-o, --output_file output_file] [-W, --width width] [-H, --height height] [-c, --widget_count widget_count] [-t, --widget_types widget_types+] [-l, --layout layout] [--random-state]

Process CLI arguments for the UI generator.

optional args:
  -h, --help                        show this message and exit
  -m, --mode                        mode the mode to run the program in
  -?, --usage                       Print usage information for that mode.
  -n, --normalize                   normalize the bounding boxes
  -o, --output_file output_file     The output file (screenshot)
  -W, --width width                 the width of the UI
  -H, --height height               the height of the UI
  -c, --widget_count widget_count   the count of widgets
  -t, --widget_types widget_types+  A list of widget types
  -l, --layout layout               the layout option
  --random-state                    Use a random state for each created widget (experimental)
```

## Widget types

Not all widget types of LittlevGL are implemented yet. You may use non-implemented widget types, but they probably will not be displayed properly or simply exist in their default state, if they have one.

The names of widget types are the lowercase names of the classes in the LittlevGL library, e.g. `lv_arc` is `arc`.

▶ Details

Implemented types

- Arc
- Bar
- Button
- Buttonmatrix
- Calendar
- Checkbox
- Dropdown
- Label
- Roller
- Scale
- Slider

- Spinbox
- Switch
- Table
- Textarea

## Layouts

The generator supports different layouts to structure the widgets inside the container. The following layouts are available:

- `none`: No layout, widgets are placed using absolute positioning. This is the default layout and recommended to use. To avoid overlapping widgets, the generator will try to find a free spot using a approximated spatial map of the UI.
- `flex`: A layout, which will align widgets in either row or column, fitting as needed. The flex mode used is hardcoded to ROW_WRAP, which means that the widgets will be placed in a row, and if the row is full, the next widget will be placed in the next row.
- `grid`: A layout, which will align widgets in a grid. The grid layout is not yet implemented, since it is very error-prone in the way widgets are randomly created and placed.

## Style randomization

The generator will always randomize the style of each widget upon creation.

It does so by randomly choosing multiple properties from a list of hardcoded properties and setting a random value for each of them. The hardcoded list can be found in the `randomize_style()` function of `src/random_ui.py`, but for convenience is also provided below.

The generator will randomize at least 3 properties, up to the length of the hardcoded property list.

The properties are applied to the widget by first creating a style object, then setting the properties on the style object and finally applying the style to the widget. This should avoid issues with properties not being available or applicable for certain widget types.

▶ Details

List of style properties used for randomization

- `set_bg_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`
- `set_bg_opa` -> `random.randint(0, 100)`
- `set_border_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`
- `set_border_opa` -> `random.randint(0, 100)`
- `set_border_width` -> `random.randint(0, 10)`
- `set_outline_width` -> `random.randint(0, 10)`
- `set_outline_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`
- `set_outline_opa` -> `random.randint(0, 100)`
- `set_shadow_width` -> `random.randint(0, 15)`
- `set_shadow_offset_x` -> `random.randint(0, 10)`
- `set_shadow_offset_y` -> `random.randint(0, 10)`
- `set_shadow_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`

- `set_shadow_opa` -> `random.randint(0, 100)`
- `set_line_width` -> `random.randint(0, 10)`
- `set_line_dash_width` -> `random.randint(0, 10)`
- `set_line_dash_gap` -> `random.randint(0, 10)`
- `set_line_rounded` -> `random.choice([True, False])`
- `set_line_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`
- `set_line_opa` -> `random.randint(0, 100)`
- `set_text_color` -> `lv.color_hex(random.randint(0, 0xFFFFFF))`
- `set_text_opa` -> `random.randint(0, 100)`
- `set_text_letter_space` -> `random.randint(0, 10)`
- `set_text_line_space` -> `random.randint(0, 10)`
- `set_opa` -> `random.randint(0, 100)`
- `set_align` -> `random.choice([lv.ALIGN.CENTER, lv.ALIGN.TOP_LEFT, lv.ALIGN.TOP_RIGHT, lv.ALIGN.TOP_MID, lv.ALIGN.BOTTOM_LEFT, lv.ALIGN.BOTTOM_RIGHT, lv.ALIGN.BOTTOM_MID, lv.ALIGN.LEFT_MID, lv.ALIGN.RIGHT_MID, lv.ALIGN.DEFAULT])`
- `set_pad_all` -> `random.randint(0, 10)`
- `set_pad_hor` -> `random.randint(0, 10)`
- `set_pad_ver` -> `random.randint(0, 10)`
- `set_pad_gap` -> `random.randint(0, 10)`
- `set_pad_top` -> `random.randint(0, 10)`
- `set_pad_bottom` -> `random.randint(0, 10)`
- `set_pad_left` -> `random.randint(0, 10)`
- `set_pad_right` -> `random.randint(0, 10)`
- `set_pad_row` -> `random.randint(0, 10)`
- `set_pad_column` -> `random.randint(0, 10)`
- `set_margin_top` -> `random.randint(0, 10)`
- `set_margin_bottom` -> `random.randint(0, 10)`
- `set_margin_left` -> `random.randint(0, 10)`
- `set_margin_right` -> `random.randint(0, 10`

## State randomization

The `--random-state` flag will randomize the state of each widget upon creation.

This is an experimental feature, as it is not always desired to be used. Additionally, randomizing the state of a widget may lead to a widget not being displayed, due to random choice of a state that is either not supported by the widget or the state hiding the widget in general.
It may also simply not affect the widget at all, which is another reason I have provided this as an optional flag.

▶ Details

List of widget states used for randomization

- `lv.STATE.CHECKED`
- `lv.STATE.DISABLED`
- `lv.STATE.FOCUSED`
- `lv.STATE.PRESSED`
- `lv.STATE.HOVERED`
- `lv.STATE.EDITED`

# Design mode

```
usage: src/main.py [-h] [-m, --mode mode] [-?, --usage] [-n, --normalize] [-o, --output_file output_file] [-f, --file file]

Process CLI arguments for the UI generator.

optional args:
  -h, --help                          show this message and exit
  -m, --mode mode                     the mode to run the program in
  -?, --usage                         Print usage information for that mode.
  -n, --normalize                     normalize the bounding boxes
  -o, --output_file output_file       The output file (screenshot)
  -f, --file                          file path to JSON design file
```

## Design file specification

Design files need to be valid according to the JSON schema ( `design_file.schema.json` ).

If design files are invalid, the design parser will throw a `ValueError` whenever it encounters required objects that are missing or have the wrong type.
For widget definition, not all properties are required and if some are missing, the generator will make up for it by randomly choosing an appropriate value.

For example, if you create the `label` widget and do not provide a `text` property, the generator will choose a random amount of symbols from the displayable ASCII table and set it as the text of the label.

The overall structure of the design file should look like this:

```
{
    "$schema": "./schema/design_file.schema.json",
    "ui": {
        "window": {
            "width": 640,
            "height": 640,
            "title": "Example design file"
        },
        "root": {
            "id": "main_container",
            "type": "container",
            "options": {
```

```
                "layout_type": "none"
            },
            "style": [
                "main_container_style"
            ],
            "children": [
                ...
            ]
        },
        "styles": {
            ...
        }
    }
}
```

Have a look at the designs folder for examples of design files. The widgets_showcase.json file is a good starting point to see usage of all implemented widget types.

## General design file rules & notes

Writing a design parser is a bit complicated, so there are some rules to follow when creating a design file:

1. It is mandatory that the first widget object in `root` is a container, as the root widget is always a container *(in any UI framework as far as I am aware)*. **Unexpected/error behavior will occur if this is not the case.**
2. The title of the window is not mandatory and also not used by the generator. It is only there for reference to the user possibly looking through dozens of design files.
3. The `styles` object is optional and can be omitted if no styles are defined.
4. Added styles are referenced by their name in the `style` array of each widget. If a style is not found, the generator will throw a `ValueError`.
5. A style defines a list of properties that are applied to widgets via the usage of a `lv.style_t` object. The possible properties are the same as documented in the LittlevGL API for styles. Properties are verified by checking if the specified name has a corresponding `setter` attribute in the `lv.style_t` object. This is done by appending `set_` to the property name, thus you are required to use the property setter function names without the `set_` prefix. For example, to set the background color of a widget, you would use the property `bg_color`. The generator will then look for the `set_bg_color` attribute in the `lv.style_t` object and apply the converted value to it.
6. If a provided `property` inside a `style` object does not actually correspond to an available attribute in `lv.style_t`, the generator will ignore it and continue.
7. Values supplied to style properties are converted according to the required type of the property. Some properties taking in special objects, like colors, require a specific string to be supplied (e.g. `#AABBCC` for any color property or `top-left` for the `align` property). You can checkout the details of the value conversion in the function `convert_value()` of `design_parser.py`.
8. If value conversion fails, the property is ignored and the generator will ignore it and continue.
9. The `id` property is mandatory for widgets of type `container`, as it is required to reference the container inside the `children` array, when the special widget type `random` is used.

10. The special widget type `random` may be used to supply a list of widget types for the generator to randomly choose from and then create a random widget in similar fashion to the random mode. This is useful for randomizing widgets in certain areas of the UI, while keeping the rest of the UI static.

## Validating design files

You can validate your design files against the available JSON schema in the repository by using the `jsonschema` package in python. Keep in mind, that `micropython` does not have this package and you will need to use the regular python interpreter to do this.

This is usually more descriptive than the error messages provided by the generator.

Here is a simple script to validate a design file:

▶ Details

validate_design.py

```python
def load_json_file(filepath: str):
    import json
    with open(filepath, 'r') as f:
        return json.load(f)

def verify_design_from_file(design_file: str, schema_file: str) -> tuple[bool, Exception]:
    from jsonschema import validate
    from jsonschema.exceptions import ValidationError
    design = load_json_file(design_file)
    schema = load_json_file(schema_file)
    try:
        validate(instance=design, schema=schema)
        print(f"Provided design file {design_file} is valid.")
        return True, None
    except ValidationError as e:
        print(f"Provided design file {design_file} is invalid:\n{e}")
        return False, e

if __name__ == '__main__':
    verify_design_from_file('path/to/design_file.json', 'path/to/design_file.schema.json')
```

# Development

Inside the `stubs` folder is the `lvgl.pyi` stubs file, which contains type hints for the LVGL micropython bindings. This is useful for development in an IDE that supports type hinting, like VS Code with the Python extension.

The `settings.json` file in the `.vscode` folder contains the necessary settings to enable type hinting for the `lvgl.pyi` file in Visual Studio Code.

The stubs file was generated by kdschlosser and supplied to me during a discussion on the LVGL forum.

The used stub generator for this file can be viewed in this PR and is generally not merged yet into LVGL, so it is not complete and may cause errors.

The created stubs file also may not cover all functions and classes of the LittlevGL bindings, but generally covers enough and linting errors are more of a nuisance than a real issue.

# Known issues

- Creating a screenshot using the `snapshot` API of LittlevGL certainly causes memory leakage due to the manually added JPEG encoding mechanism in `screenshot.py` and dereferencing of the data buffer. It is hard to deal with this without a proper JPG encoder library built into micropython binary. The memory leakage is not too severe and I attempted to mitigate it by attempting to always free the snapshot buffer using `lv.snapshot_free()` but it is not fool-proof.
- The generator may sometimes cause a memory allocation error when attempting to create the JPG buffer for the screenshot. This is due to the fact that the JPG buffer is created in heap and knowingly it is limited in size. The generator will attempt to free the buffer after the screenshot is taken, but it is not guaranteed that the buffer is freed properly. This is a known issue and there is no solution as of yet. You can try to run the generator again and it might work again after the OS has cleared up some memory.
- The JPG output of the screenshot may sometimes be corrupted or the image data is heavily distorted. This is due to race conditions between creating the snapshot buffer and LVGL re-rendering the UI. It is currently not possible to mitigate this issue without writing a custom C library for LVGL which will handle the snapshot creation and JPEG encoding in a more controlled manner. The LVGL bindings do not have exposed APIs to handle this inside micropython as far as I know.

# License

This project is licensed under the MIT License - see the LICENSE file for details.

# LVGL UI Detector Paper (Bachelor Thesis)

This repository contains the paper for the bachelor thesis "Precision at Pixel-Level: YOLO doing UI test automation" by Nikolaus Rieder (HackXIt).

# License

The Expose and Thesis are licensed under the MIT License - see the LICENSE file for details.

The template used in the LaTeX document is under the copyright of University of Applied Sciences Technikum Vienna (UAS Technikum Vienna) and is licensed at the discretion of the UAS Technikum Vienna.
The template includes any files in the folder BASE as well as the Pictures buchruecken.png and fhtw_cover.png.