DynamicWallpaper

Jul 02, 2020

Contents

1 User documentation **3**
1.1 Supported Platforms . . . 3
1.2 Installation . . . 3
1.3 Launching the application . . . 4
1.4 User Interface . . . 4
1.5 Color Modes . . . 7
1.6 Using videos from YouTube . . . 11

2 Developer Documentation **13**
2.1 How it works: Step by Step . . . 13
2.2 Problems . . . 20

DynamicWallpaper is a small Java program for playing a video as a wallpaper and settings up dynamic colors (e.g. based on your open processes or windows).

Fig. 1: Example of a color changing wave as wallpaper

CHAPTER 1

User documentation

1.1 Supported Platforms

OS	32-bit	64-bit
Windows 10	not tried	✓ yes
Windows 8	not tried	✓ mostly
Linux (KDE Plasma)	not tried	*not really*
Linux (GNOME)	not tried	*no*

Note: It's not very likely that it will work on 32-bit systems if it doesn't work on 64-bit systems.

1.2 Installation

1.2.1 Building from source

To build the application from source make sure you have a JDK (at least 8) installed and properly set up. It's all just about Gradle working well, so don't worry.

Now, download the source from GitHub using `git clone` (alternatively you could also download and extract a zipped version of the repository):

```
git clone https://github.com/JnCrMx/DynamicWallpaper
```

Then build the application using Gradle (you don't need to have it installed for this):

Windows (in CMD)

```
gradlew.bat build
```

Unix (in bash)

```
chmod +x gradlew
./gradlew build
```

- For next steps see *Launching the application*.
- For setting the program up for being able to use YouTube as a video source see *Using videos from YouTube*.

1.3 Launching the application

There are different ways to launch the application *Gradle* has built in the previous step.

1.3.1 Use a shadowJar

You can find the shadowJar in `build/libs/DynamicWallpaper-<version>-all.jar`. It already contains all required libraries. Simply copy it to any location you like and double-click it to launch.

1.3.2 Use a distribution

Besides the *shadowJar* there are also distribution archives. Those also contain all libraries, but must be extracted before using the application.

You can find them in `build/distributions/DynamicWallpaper-1.0.zip` or `build/distributions/DynamicWallpaper-1.0.tar`. Just extract them to any directory you want.

After extracting them, just open the `bin/`-directory and then execute `DynamicWallpaper.bat` (for Windows) or `DynamicWallpaper` (for Unix).

1.3.3 Use Gradle

You can also ask Gradle to run the application:

Windows (in CMD)

```
gradlew.bat run
```

Unix (in bash)

```
chmod +x gradlew
./gradlew run
```

If you use this way, you don't need to rebuild the program if the source changes. Building the program is included in this command.

- For information about controlling and configuring your wallpaper see *User Interface*

1.4 User Interface

You can easily control and configure the wallpaper with a simple user interface. To open (or close) this GUI click on the tray icon that appeared in your task bar after launching the program.

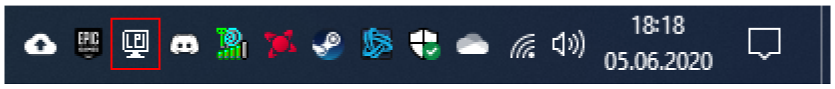
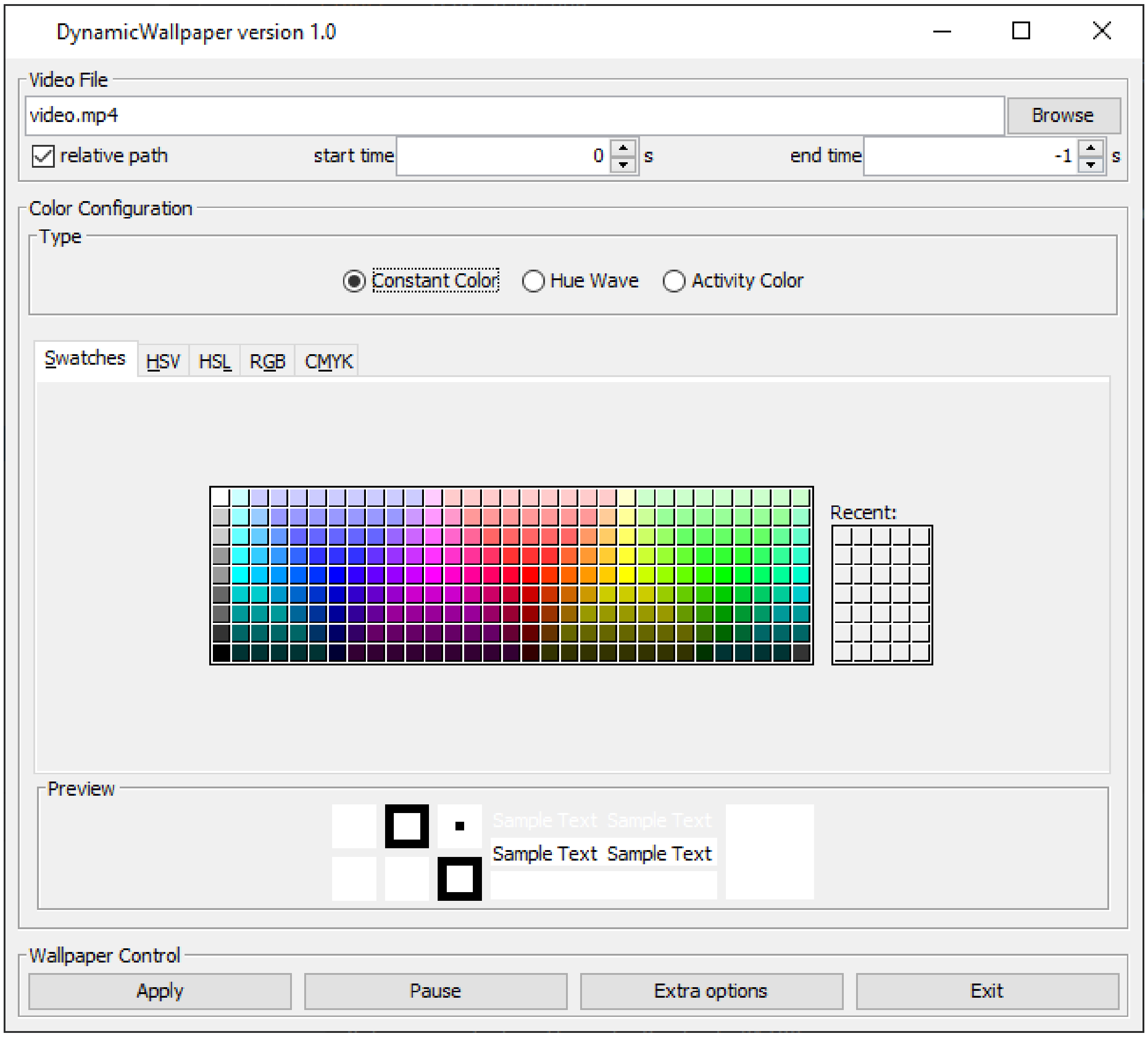


Fig. 1: Tray icon in the task bar / system tray (marked with red rectangle)

Note: You can use all mouse buttons to open/close the GUI. I'd like to implement a popup-menu, but for some reason it just doesn't work.

The GUI should look like this:



DynamicWallpaper version 1.0

Video File

video.mp4 Browse

[✓] relative path start time 0 s end time -1 s

Color Configuration

Type

[● Constant Color] ○ Hue Wave ○ Activity Color

Swatches HSV HSL RGB CMYK

Recent:

Preview

Sample Text Sample Text

Sample Text Sample Text

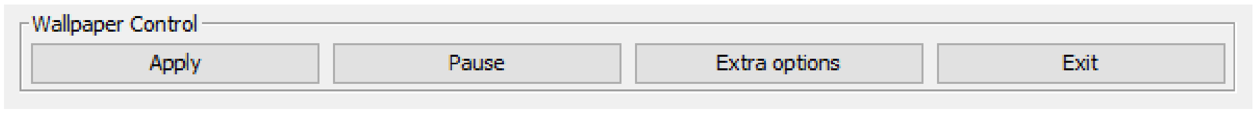
Wallpaper Control

Apply Pause Extra options Exit

You can do the following things in the GUI:

1.4.1 Controlling the wallpaper/application

Controlling the wallpaper can be done using three of the four buttons in the bottom of the GUI:



Apply

The “Apply” button is used to apply the changes you made in the configuration (see *Configuring the wallpaper*) to the wallpaper.

It also saves these changes.

Pause / Play

Using the “Pause” button you can pause the video playback and all active overlays (see Overlays). Dynamic colors are currently not paused, but I’m going to fix that soon.

Once you paused the application, the buttons text changes to “Play”. Clicking it then resumes video playback and overlays.

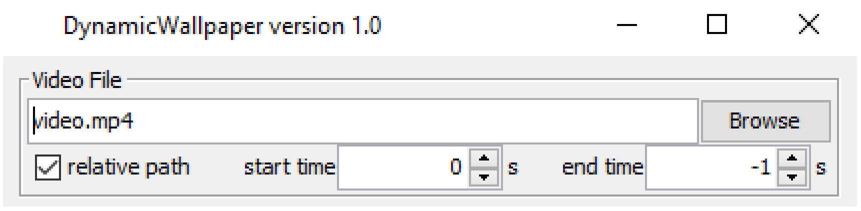
Note: Pausing and resuming overlays doesn’t work very well. This should not inflict functionality, but might cause some skipped frames/render issues.

Exit

Clicking “Exit” will prompt you with a dialog to either save or discard changes you made. After choosing your side (and optionally saving the configuration) the application terminates gracefully.

1.4.2 Configuring the wallpaper

The GUI also offers several configuration options for the wallpaper.



video.mp4
☑ relative path start time 0 s end time -1 s

Fig. 2: General configuration

The main options in this general configuration are about the video to use as the dynamic wallpaper.

Video file path

The video file path can be set in the big text field (which in the screenshot shows “video.mp4”). This path can be either relative or absolute. See *Relative path* for more details.

You can click the “Browse” button to open a file chooser dialog to help you find the video file you want to use.

Furthermore, it is also possible to use URLs (also YouTube is the right tools are installed) in this field. Refer to *Using videos from YouTube* for more information about YouTube and other video portals.

Relative path

Using this checkbox you can control how the program stores the path.

If it is checked it will try to store it as a relative path to the working directory. This might be useful if you want to be able to move the program together with the configuration and video file to another location or device.

Warning: It might not be possible for the program to store it as a relative path, e.g. due to different roots / drives. This might happen if for example the program runs somewhere on `C:\` and the video is located on `D:\`. In this case the path is stored as an absolute path.

If the video path is an URL this option has no effect and might be automatically set or unset after restarting the program.

Start and stop time

With the fields "start time" and "stop time" you can specify the timestamp at which the video should start and the timestamp at which it should restart from the beginning (aka the timestamp you specified as the start time).

Both timestamps are in seconds relative to the beginning of the video.

To use default values (aka start and end of the actual video), set "start time" to `0` (which of course **is** the actual start of the video) and "stop time" to `-1` (which tells the program to ignore this setting and use the regular video end).

Note: If the video is playing from the internet and finished caching, changing the stop time will cause the cache to be deleted and therefore the caching to restart.

This is required, because the caching will finish if the stop time is reached, regardless of whether the end of the actual video is reached or not. Without this re-downloading the video will replay at the old stop time, because the end of the cached is reached.

For information about color modes (which you can also configure in the UI), refer to *Color Modes*.

For information about color modes (which you can also configure in the UI), refer to *Color Modes*.

1.5 Color Modes

`Color Modes` are a way of dynamically coloring the video. An example for that is specifying different colors for different activities (programs, active windows, etc.).

Each color mode comes with its own configuration which is documented individually.

You can select the color mode you want to use with these radio buttons in the configuration UI:

1.5.1 Constant Color

The constant color mode provides - as you might have already guessed due to its name - a constant color. It can also be used to play a video without any color changes by simply setting the color to white.

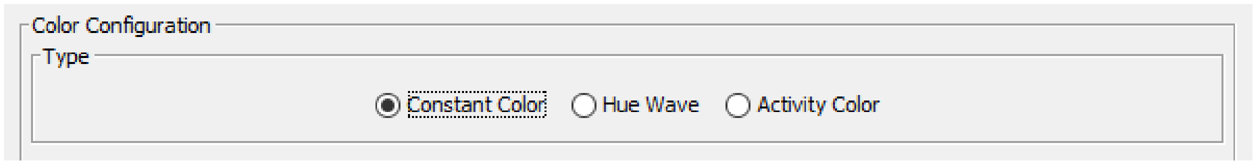


Fig. 3: Color mode selection UI.

The color to use can be selecting using the color picker in the UI:

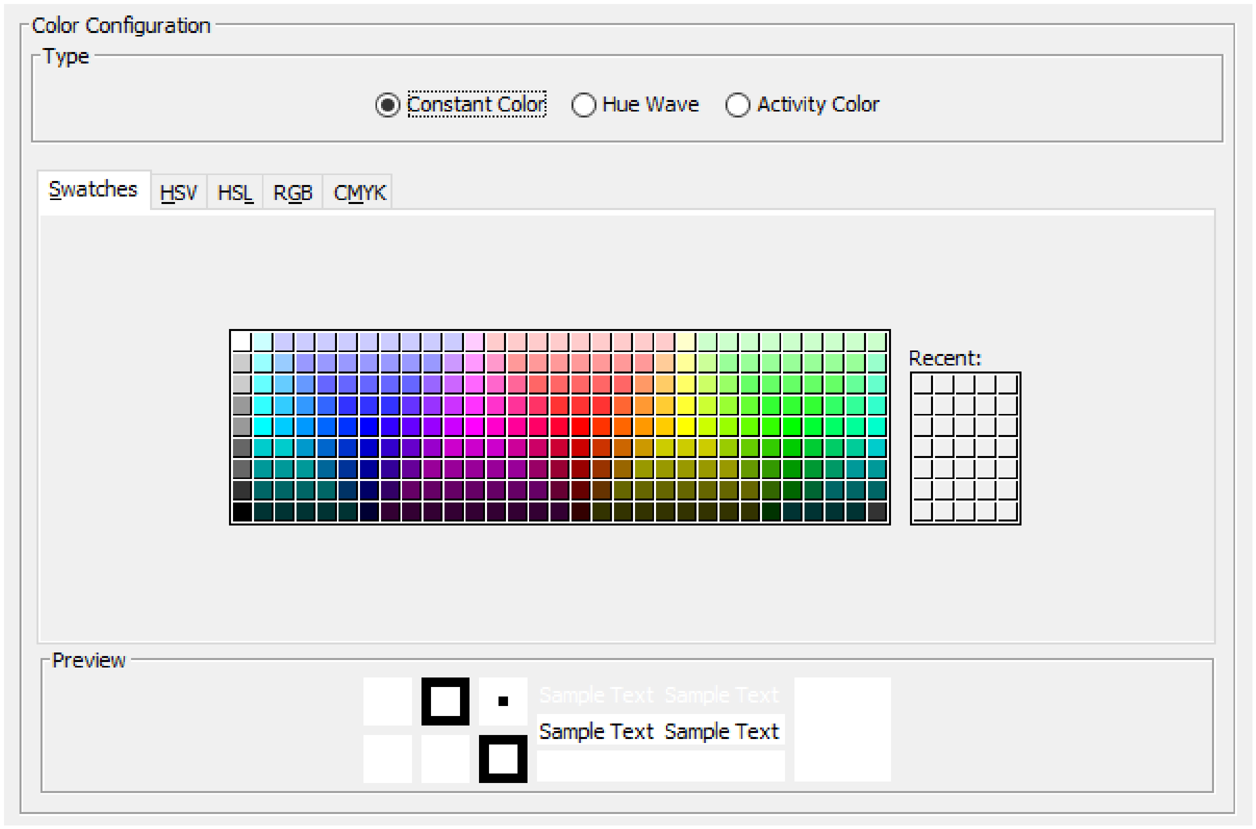


Fig. 4: Configuration panel of the "Constant Color" mode.

1.5.2 Hue Wave

The "Hue Wave" mode provides a color with permanently changing hue value.

Fig. 5: Hue color bar displaying all possible values/colors.

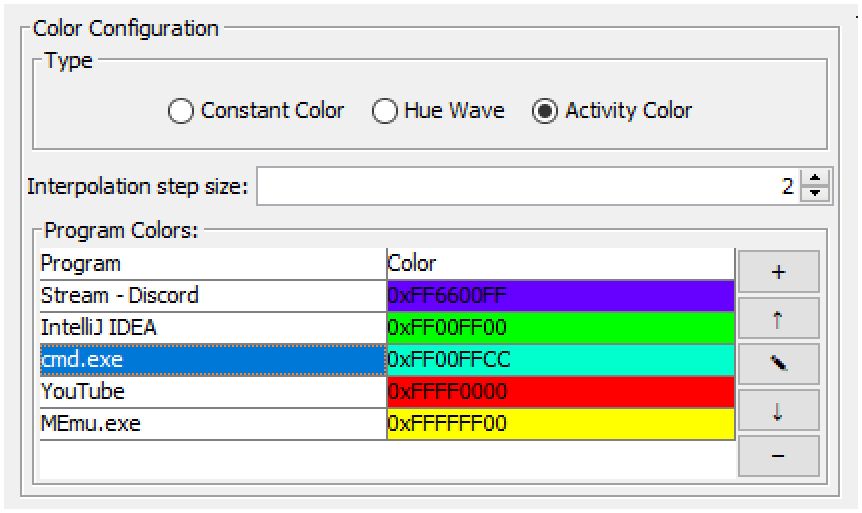
The only possible configuration option for this color mode is "Slowness". And as the name says: the higher this value the slower the color changes.

1.5.3 Activity Color

The Activity Color mode provides color accents based on the currently open programs/windows.

Note: This mode is the original reason for this programs very existance. Huge thanks to Cynder who really motivated and inspired me to create this.

The configuration of this mode is a bit more complicated, since it can handle activities and corresponding colors.



Color Configuration

Type: Constant Color   Hue Wave   [✓ Activity Color]

Interpolation step size: 2

Program Colors:

Program	Color
Stream - Discord	0xFF6600FF
IntelliJ IDEA	0xFF00FF00
cmd.exe	0xFF00FFCC
YouTube	0xFFFF0000
MEmu.exe	0xFFFFFF00

Fig. 6: Configuration panel of this mode.

Interpolation step size

This mode makes the color change slowly when the activity is changed. The speed of the change is controlled by this option.

Higher values (maximum of 256 means immediate) increase the speed of the change and lower values decrease it.

Warning: A value of `0` will completely prevent the color from changing and hence the mode from working.

Program colors

The activities and their corresponding colors are defined in the "Program colors" table. They order of the entries is important. Entries with higher position in the table override entries with lower position.

They can be selected by clicking them.

The buttons on the right side have the following effects:

Button	Effect	Availability
+	Opens a *dialog* to add a new entry.	always
↑	Move the selected entry one position up.	if an entry is selected and it is not the first one
	Opens a *dialog* the edit the selected entry.	if an entry is selected
↓	Move the selected entry one position down.	if an entry is selected and it is not the last one
–	Removes the selected entry.	if an entry is selected

Add/Modify entry dialog

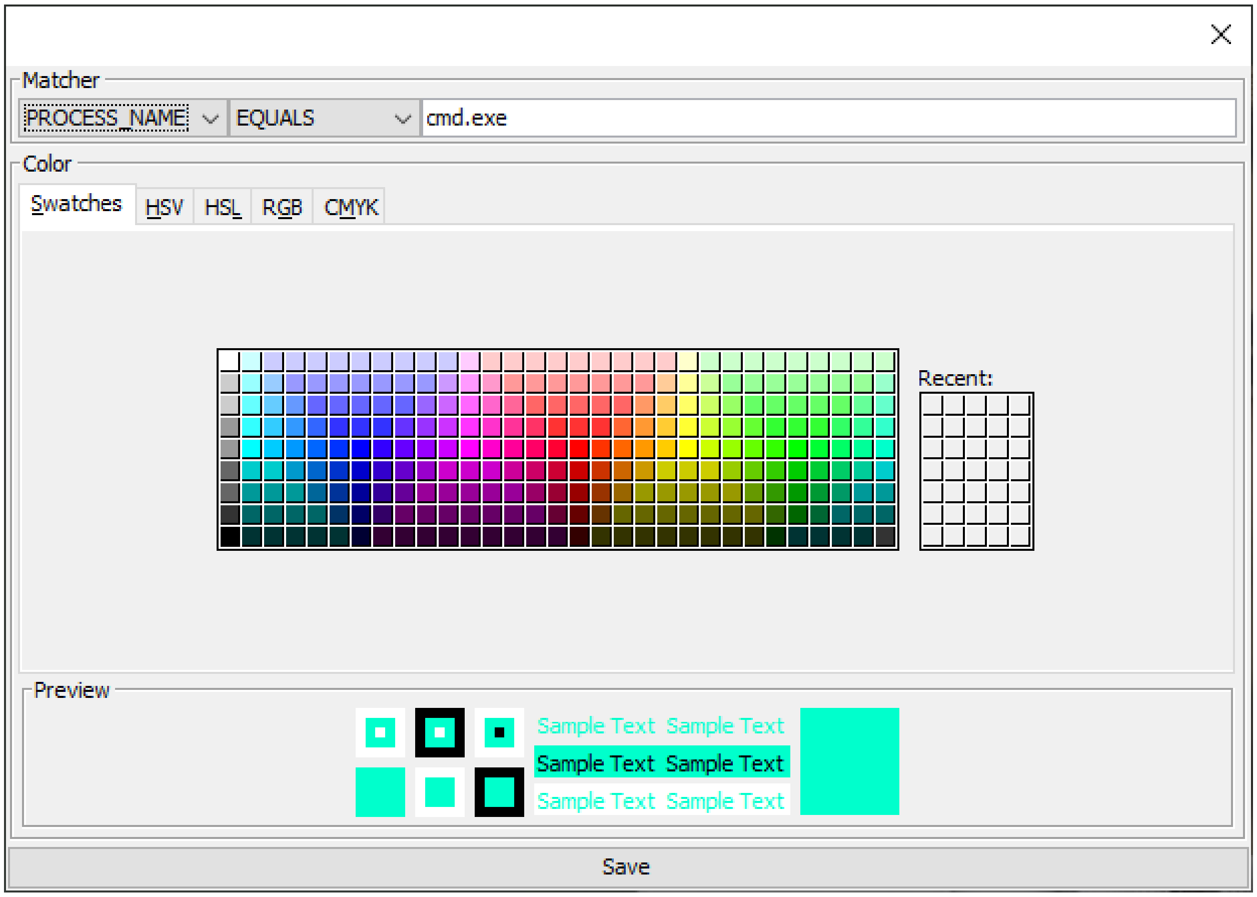

Matcher: PROCESS_NAME, EQUALS, cmd.exe

Fig. 7: Dialog for adding/modifying entries.

In this dialog you need to configure two things:

Matcher

Using the “Matcher” options in the top of the dialog you can select the program or window to match:

1. Select either `PROCESS_NAME` (for matching processes/programs) or `WINDOW_TITLE` (for matching individual windows).
2. Specify how to match the previously selected *thing*:
    - `EQUALS` is used for *exact* matching.

  - `STARTS_WITH` and `ENDS_WITH` match things starting or ending with the given string.
  - `CONTAINS` matches things that contain the given string.
3. Provide a string to search for in the text field (which says `cmd.exe` in the example screenshot).

Warning: The matcher is **case-sensitive**.

Color

To complete the configuration simply select a color using the standard color chooser.

Add or Save

The button in the bottom of the dialog says either "Add" or "Save" (depending on if you are adding of modifying an entry).

Click it to add the entry to the list or to save your changes to the existing entry you are modifying.

1.6 Using videos from YouTube

In order to use videos from platforms like YouTube as your wallpaper you just need to have youtube-dl installed.

Caution: Make also sure that the location you installed it to is in your `PATH` or it won't work. You can check this by trying to run `youtube-dl` from CMD (or Bash if you are on Linux). If that works, everything should be fine.

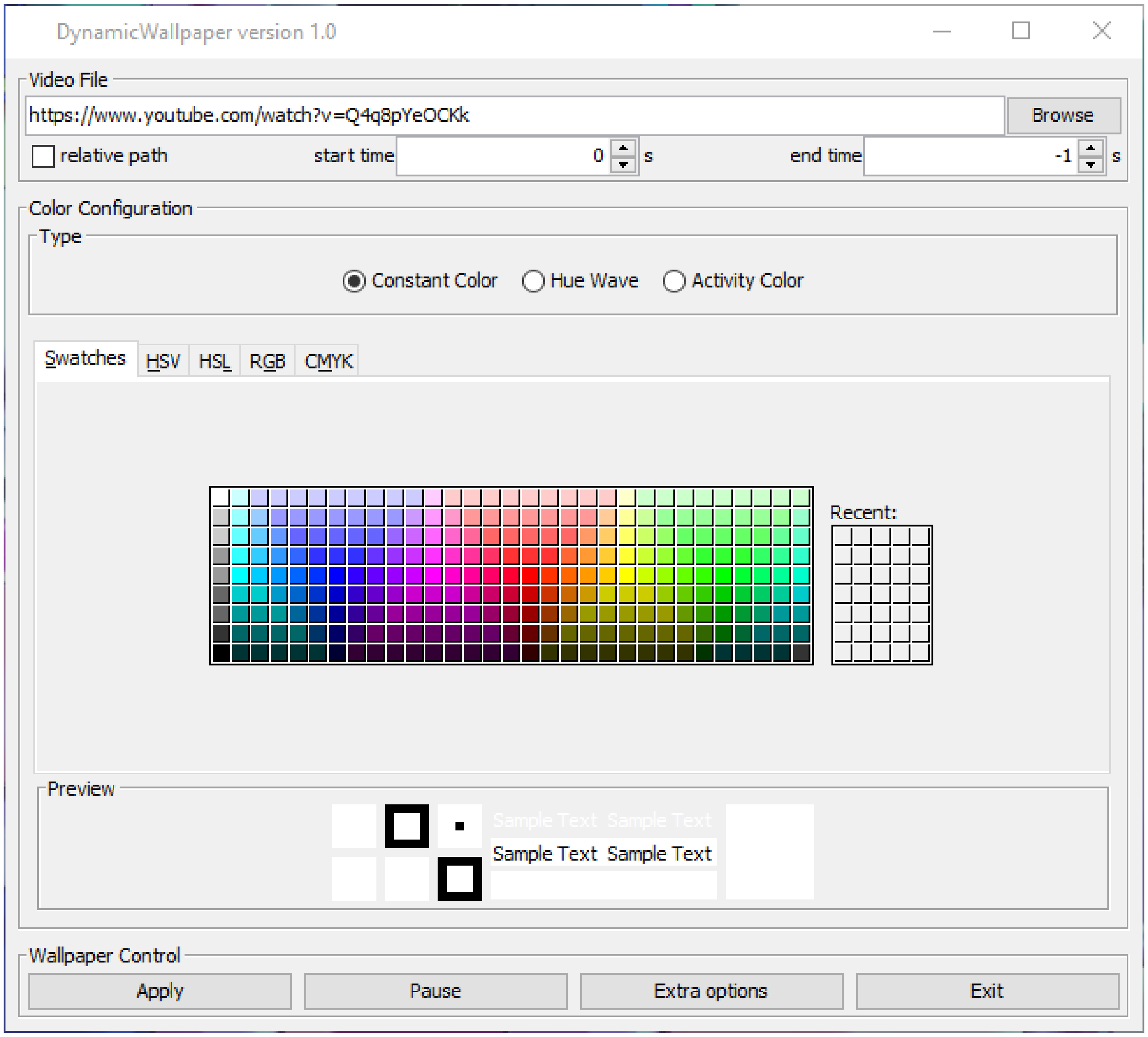
Now, you can simply enter an URL in the "Video File" text field and click apply. It might take some time to load, but then it should be playing the video as your wallpaper.

Warning: It does not work with every video, because the format this program needs is not available for every video. This issue seems to be most likely to happen with less popular videos.

Note: It might also work with other sites supported by youtube-dl, but I haven't tried, so you need to experiment a bit. ;)

While playing the video for the first time, it is cached into a file (`cache.mp4`). On the second playback is it played from this file which makes it faster and more stable.

Warning: Sometimes (e.g. if the program terminates abnormally and could not finish caching) the cached video might be empty or incomplete. In such cases you just need to manually delete the cache, so the video is downloaded again.


DynamicWallpaper version 1.0

Video File

https://www.youtube.com/watch?v=Q4q8pYeOCKk

relative path start time 0 s end time -1 s

Color Configuration

Type

[✓ Constant Color] Hue Wave Activity Color

Fig. 8: Configuration to use a YouTube video as wallpaper.

CHAPTER 2

Developer Documentation

2.1 How it works: Step by Step

2.1.1 Opening a window with an OpenGL context

The first step is to get hold on a window in which we can render with OpenGL. For that we simply use LWJGL. The code basically follows the structure of the example you can find on their website.

2.1.2 Setting a window as wallpaper

The theory behind it

The basic idea is to put a window between your “normal” desktop background and the desktop symbols.

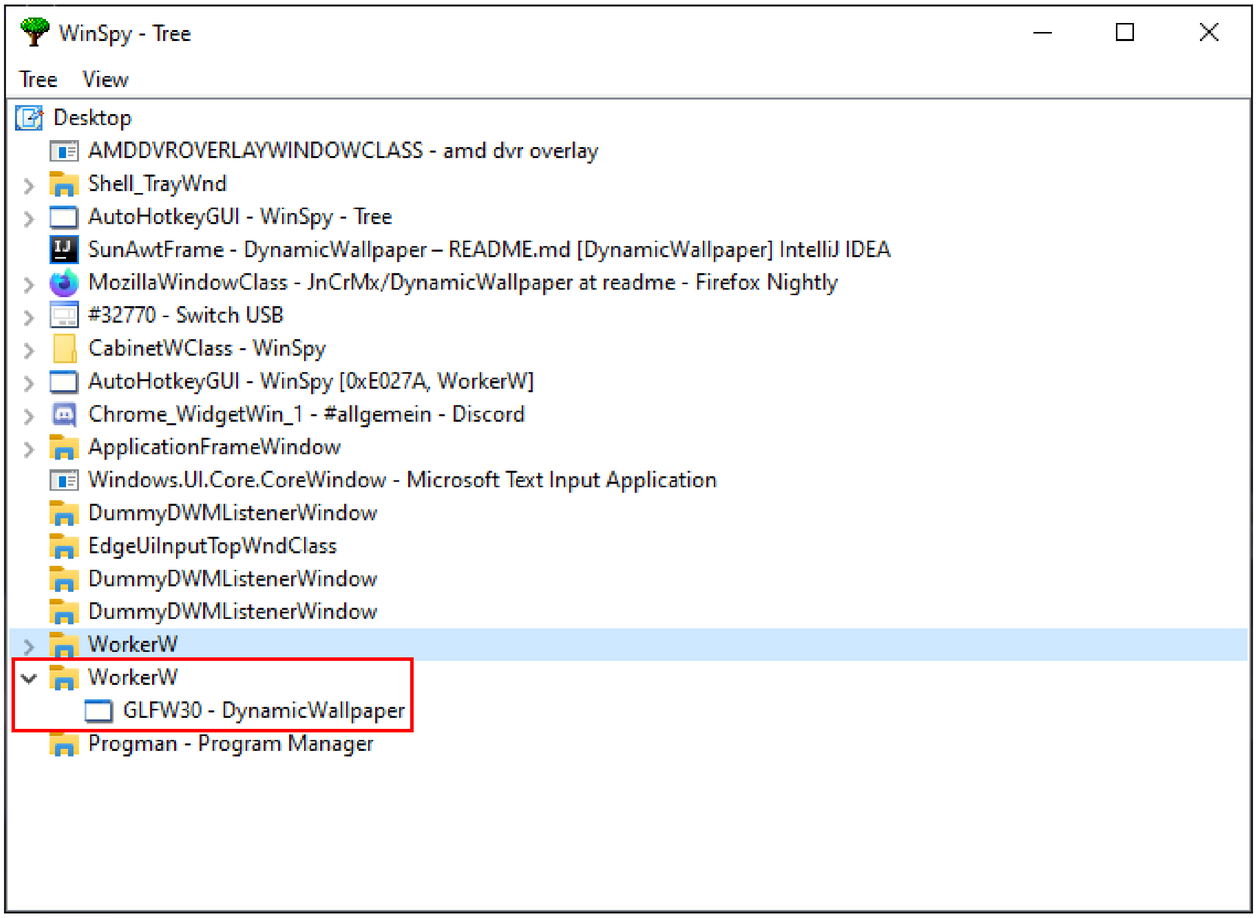
Note: I took most of the procedure from the project “weebp” by Francesco149.

I really recommend you having a look at it, if you are interested in more details about it. It also provides more features (setting any window as wallpaper).

There is a pretty nice and useful window we can use for that. It is called `WorkerW` and lies exactly where we want it to be.

To utilize this, we now need to ask `Progman` to spawn this window if it doesn’t already exists, and then make our own window a child of `WorkerW`.

Note: Here is another useful resource about this, which also explains how `WorkerW` is spawned.


WinSpy - Tree

Tree View

```
Desktop
    AMDDVROVERLAYWINDOWCLASS - amd dvr overlay
  > Shell_TrayWnd
  > AutoHotkeyGUI - WinSpy - Tree
    SunAwtFrame - DynamicWallpaper – README.md [DynamicWallpaper] IntelliJ IDEA
  > MozillaWindowClass - JnCrMx/DynamicWallpaper at readme - Firefox Nightly
  > #32770 - Switch USB
  > CabinetWClass - WinSpy
  > AutoHotkeyGUI - WinSpy [0xE027A, WorkerW]
  > Chrome_WidgetWin_1 - #allgemein - Discord
  > ApplicationFrameWindow
    Windows.UI.Core.CoreWindow - Microsoft Text Input Application
    DummyDWMListenerWindow
    EdgeUiInputTopWndClass
    DummyDWMListenerWindow
    DummyDWMListenerWindow
  > WorkerW
  v WorkerW
        GLFW30 - DynamicWallpaper
    Progman - Program Manager
```

Fig. 1: Screenshot from WinSpy showing `WorkerW` and our window.

Java implementation

After calling `glfwCreateWindow` and receiving a GLFW handle to the window, we can start setting it as our wallpaper. This is basically where the magic happens.

```
// Create the window
window = glfwCreateWindow(mode.width(), mode.height(), "DynamicWallpaper", NULL,␣
↪NULL);
if(window == NULL)
    throw new RuntimeException("Failed to create the GLFW window");
glfwSetWindowPos(window, 0, 0); // I will explain those two later
glfwSetWindowSize(window, mode.width(), mode.height());

Utils.makeWallpaper(window);
```

The beginning

So, let's have a look into the `de.jcm.dynamicwallpaper.Utils` class:

```
public static void makeWallpaper(long window)
{
    if(Platform.isWindows())
        windowsMakeWallpaper(window);
    else
        throw new UnsupportedOperationException("only available on Windows");
}
```

This first method is used to eventually support multiple platforms. Currently it only really supports Windows, so we just check if we are on Windows using JNA (see https://java-native-access.github.io/jna/4.2.1/com/sun/jna/Platform.html#isWindows– for details). If we aren't on Windows, we just throw a `java.lang.UnsupportedOperationException` and return.

If everything is right, we call `Utils.windowsMakeWallpaper` and continue our journey there:

Native window handles

```
private static void windowsMakeWallpaper(long window)
{
    long nativeWindow = GLFWNativeWin32.glfwGetWin32Window(window);

    // procedure from https://github.com/Francesco149/weebp
    WinDef.HWND thisWindow = new WinDef.HWND(new Pointer(nativeWindow));
```

Here we want to operate on the window as the OS (Windows) sees it and not as GLFW sees it. Therefore, we need to ask GLFW to give us the "native" handle of our window. In the rest of the method we only work with this one and **not** with GLFW's handle. You can safely assume that every window handle in this section mean the native handle. The next step is to wrap the returned handle into JNA's / Windows' structure for Window handles. This makes it easier for us to operate with it and pass it to other JNA functions.

```
WinDef.HWND workerW = getWorkerW();
```

Now we need to spawn and find the `WorkerW` which we do in a separate method.

Spawning `WorkerW`

To spawn `WorkerW` all we need to do is sending two (undocumented) messages to `Progman`.

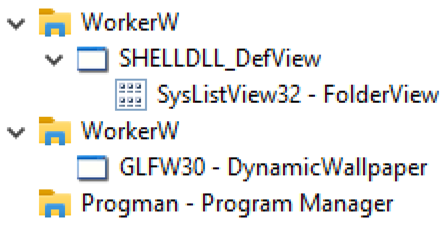
```
30 private static WinDef.HWND getWorkerW()
31 {
32     WinDef.HWND progman =  User32.INSTANCE.FindWindow("Progman", null);
33 
34     User32.INSTANCE.SendMessage(progman, 0x052C, new WinDef.WPARAM(0xD), new WinDef.
   ↪LPARAM(0));
35     User32.INSTANCE.SendMessage(progman, 0x052C, new WinDef.WPARAM(0xD), new WinDef.
   ↪LPARAM(1));
```

To do so, we first need to find `Program` (line 32) and then we can just send the messages (lines 34-35).

Note: We need to wrap some arguments in `WinDef.WPARAM` and `WinDef.LPARAM`, because JNA does not do that automatically and the Windows API requires it.

Finding `WorkerW`

The problem with finding `WorkerW` is that - as you can see in the figure below - there are two `WorkerW` windows: One of them contains a window with the class `SHELLDLL_DefView` and the other one will contain our frame.



```
WorkerW
    SHELLDLL_DefView
        SysListView32 - FolderView
WorkerW
    GLFW30 - DynamicWallpaper
Progman - Program Manager
```

Fig. 2: Screenshot from WinSpy showing both `WorkerW` windows.

Hence we need a way to “skip” the first one and then find the second one.

```
37 AtomicReference<WinDef.HWND> workerRef = new AtomicReference<>();
38 User32.INSTANCE.EnumWindows(new WinUser.WNDENUMPROC()
39 {
40     @Override
41     public boolean callback(WinDef.HWND hWnd, Pointer data)
42     {
43         if(User32.INSTANCE.FindWindowEx(hWnd, null, "SHELLDLL_DefView", null)==null)
44             return true;
45 
46         WinDef.HWND worker = User32.INSTANCE.FindWindowEx(null, hWnd, "WorkerW",␣
   ↪null);
47         if(worker != null)
48         {
49             workerRef.set(worker);
50         }
51 
52         return true;
53     }
```

(continues on next page)

(continued from previous page)

```java
}, null);
return workerRef.get();
```

At the very beginning of this part we need to create an `AtomicReference` to store the result. We need this, because we will be operating within an inner class and are therefore unable to directly set local variables outside this inner class.

Now, we iterate over all top-level windows on the screen using `EnumWindows`:

For each of those windows we then check if it contains the `SHELLDLL_DefView` window. So, we basically search for the first `WorkerW` (the one we don't want). If we found this `WorkerW` we proceed with our code otherwise we return `true` and continue our search with the next top-level window.

To find the `WorkerW` we actually need, we search for a `WorkerW` in the root window using `FindWindowEx`. To avoid finding the "wrong" one (which is the one we just found in the previous step), we tell the method to start searching after the first `WorkerW` which results in it returning the second one. We do this by simply passing a handle to the wrong one to `FindWindowEx` as the second parameter (`hWndChildAfter`).

If we successfully found such a window, we put it into our `AtomicReference`.

Note: At this point we could return `false` to stop iterating over the top-level windows. To be honest I'm not sure why I don't do this.

Finally, we return the handle to the `WorkerW` we (hopefully) found and stored in the `AtomicReference`.

Modifying our window styles

Returning to the `de.jcm.dynamicwallpaper.Utils.windowsMakeWallpaper` method, we now need to adjust some window styles to make it work as wallpaper.

```java
long style = User32.INSTANCE.GetWindowLong(thisWindow, User32.GWL_STYLE);
style &= ~(
        WS_CAPTION |
                WS_THICKFRAME |
                WS_SYSMENU |
                WS_MAXIMIZEBOX |
                WS_MINIMIZEBOX
);
style |= User32.WS_CHILD;
User32.INSTANCE.SetWindowLong(thisWindow, User32.GWL_STYLE, (int) style);

// not sure if we need those, but better keep them in
long exStyle = User32.INSTANCE.GetWindowLong(thisWindow, User32.GWL_EXSTYLE);
exStyle &= ~(
        WS_EX_DLGMODALFRAME |
                WS_EX_COMPOSITED |
                WS_EX_WINDOWEDGE |
                WS_EX_CLIENTEDGE |
                WS_EX_LAYERED |
                WS_EX_STATICEDGE |
                WS_EX_TOOLWINDOW |
                WS_EX_APPWINDOW
);
User32.INSTANCE.SetWindowLong(thisWindow, User32.GWL_EXSTYLE, (int) exStyle);
```

There are certain styles we apparently need to remove from our window. Doing that is really simple by just getting the current styles, removing the flags using a bitwise and with the bitwise compliment of the flags we want to remove, and finally setting the modified styles.

Note: The style blacklist is taken from https://github.com/Codeusa/Borderless-Gaming/blob/2fef4ccc121412f215cd7f185c4351fd634cab8b/BorderlessGaming.Logic/Windows/Manipulation.cs#L70

Making `WorkerW` adopt our window

Making the `WorkerW` we just found adopt our window (so it becomes our window's parent and we inherit its stacking position) is rather simple:

```
User32.INSTANCE.SetParent(thisWindow, workerW);
User32.INSTANCE.ShowWindow(thisWindow, User32.SW_SHOW);
```

We just call `SetParent` to set `WorkerW` as our parent and then make our window visible using `ShowWindow`.

Note: To be honest I'm not sure if the `ShowWindow` step is neccessary, because we will make the window visible using GLFW's `glfwShowWindow` later.

Buf it probably won't hurt since the documentation states the following:

> If the window is already visible or is in full screen mode, this function does nothing.

Done!

Adjusting position and size

Almost done.

We still need to take care about one small thing mainly related to decorated and undecorated windows in GLFW.

The problem is that our window needs to be decorated (or it won't work for some reason) and is therefore a bit smaller than the actual screen and won't cover the whole desktop background.

```
WinDef.RECT rect = new WinDef.RECT();
User32.INSTANCE.GetWindowRect(thisWindow, rect);
```

The fix this we first need to store our window's size before modifying its styles and making it the wallpaper.

```
// not sure wtf we do here, but it seems to work (not really well, but idk)
User32.INSTANCE.MoveWindow(thisWindow, 0, rect.top, rect.right,
                           rect.bottom+10, false);
rect.clear();
```

Then after setting our parent to `WorkerW`, we move our window to its previous position and add `10` to its height. I'm not sure why this works, but it does. Finally we release the `WinDef.RECT` we allocated.

Note: This is also the reason why we need to set the window position and size via GLFW after creating the window:

```
glfwSetWindowPos(window, 0, 0);
glfwSetWindowSize(window, mode.width(), mode.height());
```

And now we are *really* **done**!

2.1.3 Playing a video

Playing a video in our frame is basically done in three steps:

Decoding video frames

To decode the video frames we use FFmpeg with bindings provided by JavaCV. So, we can basically make a class called `FFmpegFrameGrabber` do all the work for us.

First, we need initialize a `FFmpegFrameGrabber` and then start it:

```
File file = ...;
FFmpegFrameGrabber grabber = new FFmpegFrameGrabber(file);
grabber.start();
```

Here we want to read and decode a file.

If we want to read from a stream we need to do some adjustments to prevent it from reading the stream to its end when calling `start`:

```
InputStream in = ...;
grabber = new FFmpegFrameGrabber(in, 0);
grabber.setFormat("mp4");
grabber.start(false);
```

We first need to pass this `0` as the second argument to the constructor to tell the FrameGrabber not to read the stream in chunks. This makes it impossible to seek the stream backwards, but we are fine with that, because we don't want to do that anyway.

Then we need to tell the FrameGrabber which format the stream is gonna be in. In our case that is `mp4`. I'm not sure which formats are possible, just read the documentation or try it out.

Finally we need to pass `false` to the `start` method's `findStreamInfo` argument. This tells the FrameGrabber not to read the stream info. This heavily reduces startup time, because for MP4 this information is located at the end of the file or stream. So, passing `true` (or nothing) would require the stream to be fully read before we can start decoding.

Warning: Using all these options might decrease stability of the FrameGrabber, but we don't really care about that. Losing a few frames should not be a problem after all.

Actually grabbing the frame is the same no matter if we read from a stream or a file:

```
Frame frame = grabber.grabImage();
ByteBuffer buf = (ByteBuffer) frame.image[0];
```

We can simply grab an image frame using `grabImage` and then retrieve the raw image from the `image` array in the `Frame`.

Luckily we don't need to de- or encode the image, because OpenGL can handle the raw image as it is decoded by the FrameGrabber.

Uploading frames to an OpenGL texture

Uploading the image to and OpenGL texture is even easier:

```
glTexImage2D(GL_TEXTURE_2D,          // target to upload the image to
             0,                      // level of detail
             GL_RGB8,                // internal format
             frame.imageWidth,       // frame width
             frame.imageHeight,      // frame height
             0,                      // border width
             GL_BGR,                 // texel format
             GL_UNSIGNED_BYTE,       // texel type
             buf);                   // image
```

Most of those argument are either not important or should be easy to understand with basic knowledge of OpenGL. Just look at the documentation or read/watch some OpenGL tutorials.

Caution: We pass `GL_BGR` and not `GL_RGB` as texel format. That is also the format the FrameGrabber decodes the video to. If you don't pay attention to this, your colors might get messed up.

Rendering a plane with OpenGL

2.2 Problems

2.2.1 Why doesn't it work well on Linux?

I've experimented a bit on Linux and searched for ways to put a window as a wallpaper, but at least with KDE Plasma none of them really worked.

This is because the desktop background and the desktop icons appear to be in the same windows. Hence, it seems to be impossible to put a window between them.

We change the property `_NET_WM_WINDOW_TYPE` of our window to `_NET_WM_WINDOW_TYPE_DESKTOP` which makes it appear as the desktop window. Sadly our window is still above the original desktop window and therefore covers all symbols and prevents you from interacting with your desktop.

Making sure our window is below the desktop seems hard and wouldn't really help, because then the desktop window (which included the desktop background) would cover our window instead.

I need to experiment a bit more; maybe setting a transparent wallpaper could fix this.

2.2.2 What's the problem on GNOME/Ubuntu?

The application uses an icon in the system tray which you can click to open or close the configuration window.

Ubuntu/GNOME does not support the system tray anymore.

There might be some plugins providing a system tray, but I haven't tested them yet.