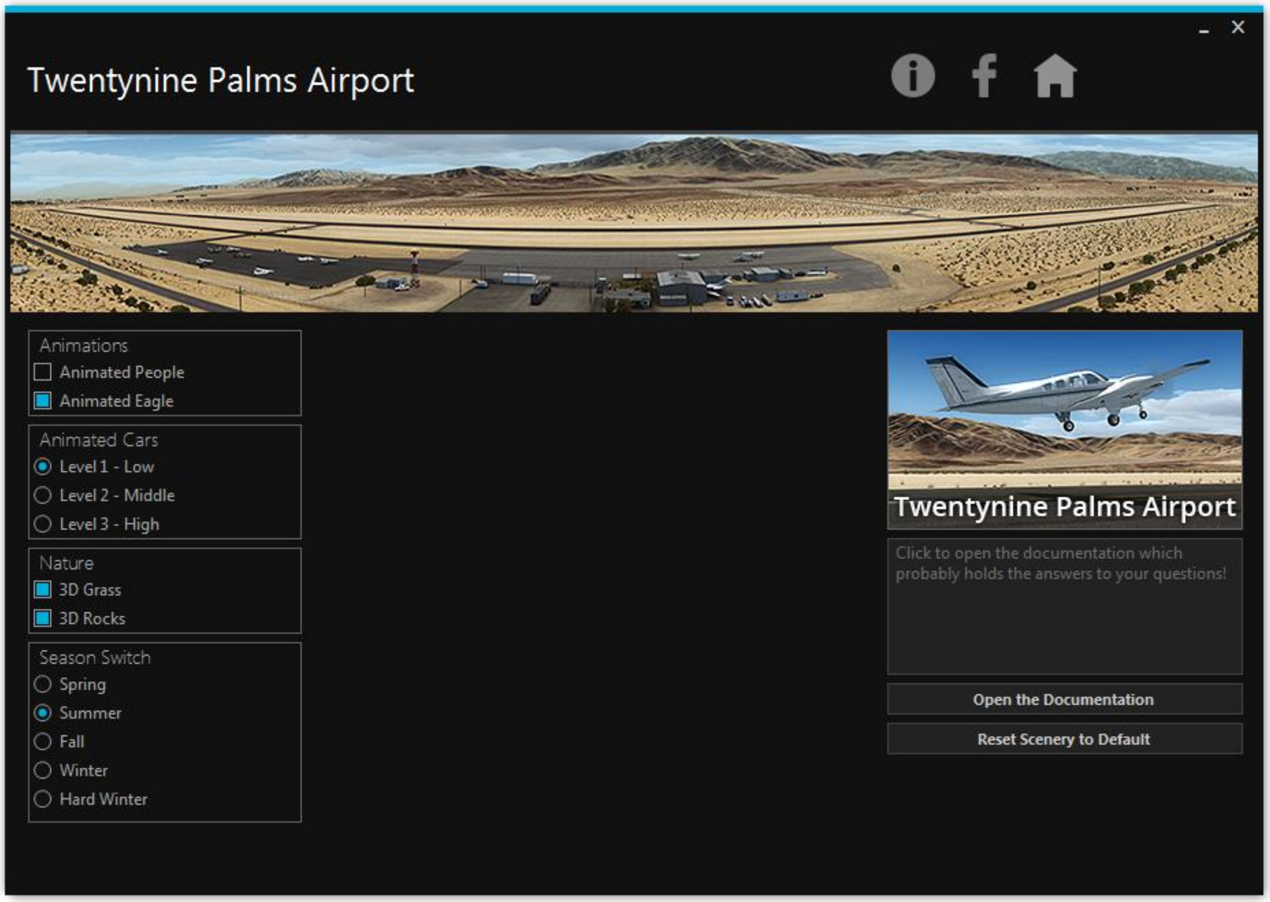
# Scenery Configurator
Pro v2.0.0.4


Twentynine Palms Airport

Animations
- [ ] Animated People
- [✓] Animated Eagle

Animated Cars
- [✓ Level 1 - Low]
- Level 2 - Middle
- Level 3 - High

Nature
- [✓] 3D Grass
- [✓] 3D Rocks

Season Switch
- Spring
- [✓ Summer]
- Fall
- Winter
- Hard Winter

Twentynine Palms Airport
Click to open the documentation which probably holds the answers to your questions!
Open the Documentation
Reset Scenery to Default

## Contents

1. Introduction 2
2. Limitations & Requirements 2
3. Getting started 3
    a. The testbed 3
    b. The scenery configuration 3
4. Updating your scenery 5
5. Updating the Configurator (existing customers) 6
6. License 6

## 1. Introduction

This program allows flight simulation developers to create a customizable scenery configurator based on their needs.
The basic concept is a program that activates and deactivates files by changing the file extension. This makes it possible for the developer to provide **different levels of detail** and complexity which can be chosen by the end-user.
This concept makes it even possible to change the appearance of the scenery based on the season the user has selected. **Season switching** in the common flight simulators such as Microsoft's Flight Simulator X or Lockheed Martin's Prepar3D is a complex topic since the BGL file format does not allow automatic season switching by default. This tool makes it possible for the user to switch seasons and much more, it is all up to **your** configuration!

## 2. Limitations & Requirements

With the purchase of the Scenery Configurator PRO you purchased a license for use with **one commercial** scenery. If you want to use the Configurator in additional projects you should consider buying more licenses.

Unlike the lite version of the Scenery Configurator Pro supports an **unlimited** number of option groups and options per group.
**Option group** means that you can group options, e.g. "Animations" or "Season Switch".
**Option** means a checkbox or a radiobutton which controls one or more files.

Note that the program requires **.Net Framework 4.5** to be installed on your computer and the end-user's computer.

It is highly recommended that you use a proper XML editor to modify the xml file to your needs. A great choice would be **Sublime Text 3** which can be downloaded here.
You can change the Color Scheme of Sublime Text 3 to "Sunburst" by selecting Preferences -> Color Scheme -> Sunburst. That will ensure that the code has the same layout on your computer.
Also ensure that "Word Wrap" is activated in View -> Word Wrap.

# 3. Getting started

## a. The testbed

In the following, requirements, such as folder names, xml tags and attributes will be marked **red**.
This means when you implement the program in your project these folders, attributes and tags have to be present, otherwise the program will NOT work correctly or fail to start.
Texts, tags, attributes etc. marked **green** are up to your choice. You can include those attribute and tags or leave them out.

Start by getting to know the file structure of the program.
You can find a testbed in this package. A testbed exemplarily represents the inclusion of this software in a scenery add-on.
The testbed includes the AddOn Scenery folder which is usually the installation directory of scenery add-ons. Navigate to that folder and enter the subfolder "Developer – Scenery" which is generic for a possible folder naming convention.
Start the program "Scenery Configurator Pro.exe".
The program reads the scenery configuration file, an XML file, which can be found in the subfolder called "scripts". In this example the scenery configuration file is named "config_YourScenery.xml". How you name this file is up to you. But it should be clear that this file represents a scenery configuration file.
Let's open the scenery configuration file in "Testbed/AddOn Scenery/Developer-Scenery/scripts/config_YourScenery.xml". While having the corresponding program opened, compare the structure of the XML to the layout of the program. You should quickly understand how the program reads the XML and builds its controls on start-up.
But let us discuss this step by step.

## b. The scenery configuration

The first line <?xml version="1.0" encoding="utf-8" ?> defines the file format and is required.
It is followed by the opening tag <configuration>. Note that every opening tag has a closing tag. The first "level" configuration is closed at the very end of the document by this: </configuration>.

One level down comes the <title> tag. Its value, in this case "Twentynine Palms Airport" can be modified by you. It will become the program's heading the next time you start the program.

The <logo> tag is a self-closing tag. Self-closing tags look like this: <logo/>.
The <logo> tag has an attribute which defines the path to the logo which will be displayed in the program. In this example it looks like this: <logo path="images\logo.png"/>. You can freely chose the location of the logo and its name but that file has to be present. If you don't want to include a logo in your program, remove the tag entirely from the configuration file.
The recommended width of the logo (or panorama image you might call it) is **920px**.

The <doc /> tag is also a self-closing tag.
The <doc> tag has an attribute which defines the path to the manual or a documentation which can be launched from within the program. In this example it looks like this:

<doc path="doc\Manual.pdf"/>. You can freely chose the location of the manual and its name but that file has to be present. If you don't want to include a manual in your scenery, remove the tag from the configuration program entirely.

The <thumbnail /> tag is a self-closing tag as well.
The <thumbnail> tag has an attribute which defines the path to a thumbnail which can represent your scenery. In this example it looks like this: <thumbnail path="images\thumb.png"/>. You can freely chose the location of the thumbnail and its name but that file has to be present. If you don't want to include a thumbnail in your program, remove the tag entirely from the configuration file. Image files should not exceed the width of **260px**! The recommended size is **260px x 146px**.

If included, the <facebook /> tag displays a facebook icon on the Scenery Configurator menu bar which can link to your facebook page.
The "href" attribute should hold the hyperlink to your facebook page. An example for the tag is: <facebook href="https://www.facebook.com/29PalmsSceneryDesign"/>
You can remove the tag from the configuration file if you don't want to link your facebook page.

The <website /> tag works the same way and allows you to link to your website or shop.

If included, the <theme /> tag styles the Scenery Configurator with a Light or Dark Theme. The default Theme is Dark. You can change the value of the "style" attribute to "Light" and should see a light version of the Configurator when you start it the next time.

Then come the <Optiongroup> tags which hold the different options. Note that I marked the tag green since theoretically you can define no <Optiongroup> at all, the program will start up correctly. Each <Optiongroup> tag has two attributes: "text" and "type".
The text attribute gives the option group in the program a header. You can leave this attribute empty if you don't want the option group to have a header, like this:
<Optiongroup text="" type="checkboxes">

The type attribute defines the type of the options within the group. This attribute HAS to be set to either "checkboxes" **or** "radiobuttons".
Let's take a look at how checkboxes and radiobuttons work:

- Checkboxes: An option group can have options defined as checkboxes. Each checkbox controls one or more files. The checkboxes within the option group work **independently** of one another.
- Radiobuttons: An option group can have Options defined as radiobuttons. Each radiobutton controls one or more files BUT within the option group only ONE radiobutton can be activated at a time. The other radiobuttons and the files they control are automatically deactivated. Radiobuttons **depend** on one another.

Depending on how you set up your scenery, either checkboxes or radiobuttons is the right choice.

Each option group holds <Option> tags. Each <Option> tag has four attributes: "text", "default", "description" and "thumbnail". The text attribute gives the option in the program a name which is being displayed on the right side of the control.

The default attribute indicates whether this option is active by default or not. The value of this attribute can be either "on" or "off" and the initial state of your files should correspond to this setting. This attribute is **mandatory** meaning it **has** to be either "on" or "off".
Groups of radiobuttons can only have **one** option active at a time meaning only **one** default value within that group can be set to "on". Read below for further information.

The text you have defined in the description attribute appears in the description textbox when you hover over the option in the program.

Once again, if you don't want the option to have a text and / or a description you can leave the attributes empty, like this:
<Option text="" default="**on**" description="" thumbnail="images\thumb_cars.png"> or
<Option text="" default="**off**" description="">

Note that the first Option in this example has an optional **thumbnail** attached to it which will appear in the thumbnail box if you hover over the Option.

Each option holds one or more <file> tags.
Each <file> tag has two attributes: "onpath" and "offpath". These attributes are required for each <file> tag.
The onpath attribute holds the filepath to the activated file. The offpath holds the filepath to the deactivated file. Generally I would recommend that the offpath has the same value as the onpath but has an ".off" extension added to it.

Now, what if you want to have an option deactivated by default? Very simple! Just go into the scenery folder and rename the file from "example.bgl" to "example.bgl.off". The program will see that the file is deactivated and uncheck the checkbox on start-up.
If files should be controlled by radiobuttons that means only **ONE** set of files can be activated at a time.
Look into the Testbed for an exemplary set-up of radiobuttons and their associated files.

**New in version 2.0.0.4**

The <Seasons> tag allows you to define seasons for your scenery. The basic concept works as follows: Instead of switching seasons by creating different versions of a model BGL which references different textures based on the selected season, it makes more sense to copy textures corresponding to the selected season.
So you would create subfolders in the texture folder of your scenery. The default naming convention is "texture.SU" for the summer textures, "texture.FA" for your fall textures and so on. How you name the folders is up to you since you tell the program in your configuration which folder to use for a specific season.
The <Seasons> tag has one required attribute: current="Summer". So it looks like this:
<Seasons current="Summer">.
Before releasing the scenery, you should change the attribute to the default season of your scenery (most likely Summer).

Please note that each time you toggle the seasons in the Configurator, it will change the attribute to the currently selected season when you close it. So it "knows" which season is currently active the next time you start it.

Now let's get to the definition of seasons. If you take a look at the config.xml file in the Testbed, the scenery has five seasons referencing five subfolders of the main texture folder:
Spring, Summer, Fall, Winter and HardWinter. You could also reference only two seasons, for example Summer and Winter.
A tag looks like this:

```
<SeasonName>
        <folder source="soureFolder" destination="destinationFolder" />
</ SeasonName >
```

The SeasonName must be one of the following:

Spring, Summer, Fall, Winter, HardWinter (no spaces!), January, February, March, April, May, June, July, September, October, November, December.

You see that you can even specify seasons for the different months.
The source and destination attributes **must** be valid paths to directories that exist! The directories are relative to path of the Scenery Configurator.
The destination folder should always be the main texture folder of your scenery. So just leave it at destination="texture". Here's an example configuration for seasonal textures that the user can activate in January:

```
<January>
        <folder source="texture\texture.January" destination="texture" />
</ January >
```

So what should the different folders contain?
The **texture** folder should contain the **default** seasonal textures of your scenery (most likely summer).
The subfolders should contain the same seasonal textures in the corresponding seasonal variation. When the user switches the season to (say) Fall, the textures from the texture.FA folder will be copied to the main texture folder and the seasonal textures that were there before (say the summer textures) will be overwritten.
Does that make sense to you? If not, feel free to drop me a message: contact@29palms.de

I'd recommend that you try and "mess" around with the testbed till you feel comfortable enough to transfer the program and the general concept to your projects.
Nothing can go wrong in the testbed! Even if the program crashes, it won't damage your computer nor will it have a negative effect on any file outside of that testbed.
If you feel like you've messed the configuration or the testbed up, delete it and start with a fresh one.
Happy coding!

## 4. Updating your Scenery

What if you want to update your scenery in the future? The main difference to "static" sceneries which don't use Configurators is that we're basically renaming files in order to control scenery settings. So if you want to update your scenery you have to keep in mind that the scenery files might have been renamed by the Configurator.
For example:
You have animated birds in your scenery which are activated by default and the filename is "birds.bgl". But your customer has deactivated the birds using the Configurator. Now the filename is "birds.bgl.off". You want to update the birds.bgl file but you don't know in which "state" the file is.
There are multiple ways to solve the issue and you have to pick the option which works best for you:

1. Let your installer search for and then delete "birds.bgl" as well as "birds.bgl.off" prior installing the new file.
2. Prompt your customer prior installation of the update the he / she should reset the scenery to default (option in the Scenery Configurator) prior installing the update.
3. Only provide full installers instead of updates and uninstall the old scenery before installing the newer version.

You should try to avoid installing updates over a scenery which uses the Configurator without considering "what could go wrong". The worst case is that the user ends up with both files "birds.bgl" and "birds.bgl.off" at the same time. This won't let the Configurator crash though, it will delete the "birds.bgl" file the next time the option is toggled. But isn't that newer version of the file? In order to avoid this, please consider the three case studies above and decide which one works best for you!

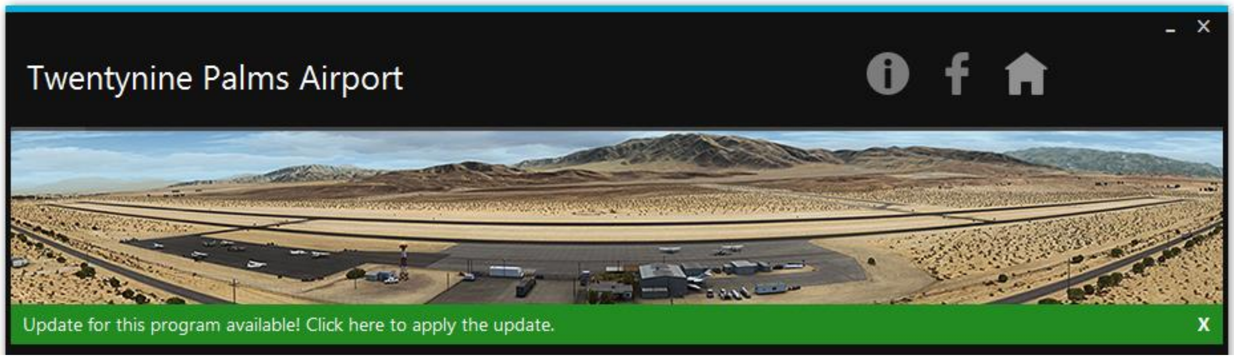
## 5. Updating the Configurator (existing customers)

Customers of the "Customizable Scenery Configurator v1" which purchased the program at 29palms-store.de prior December 2016 can update their existing scenery projects with this never version of the Configurator free of charge. The new Configurator can be included in your project without any adjustments in most cases since it "understands" the same XML syntax as the old version of the Configurator.
It is recommended though that you **adjust the sizes of your logo and thumbnail images** in order to match the new layout.
In any case, it is highly recommended that you **test** the integration of the new Configurator in your existing scenery project thoroughly!
If you want to update your existing scenery project with the new version of the Configurator, in most cases you only need to **overwrite the original Scenery Configurator .exe file and the thumbnail and logo images with the newer versions**.

**New in version 2.0.0.4**

This version includes an automatic Updater for the Scenery Configurator. On start-up, the Configurator will check for the latest version on our Servers. If it finds a newer version, it will display a message to the user. If the user clicks on the message, the automatic update process will start.

Therefore the Scenery Configurator creates a “modules” subfolder in its directory where it extracts the Updater application.


Twentynine Palms Airport
Update for this program available! Click here to apply the update.
X

Please note the “Additional Comments” section in the following chapter!

# 6. License

## End User License Agreement

This is a legal agreement between you and 29Palms Scenery Design covering your use of the Customizable Scenery Configurator Lite (the "Software").

You can redistribute the Software and/or modify it under the following terms:

1. **Copyright**. This Software is owned by 29Palms Scenery Design (Lars Pinkenburg) and is protected by copyright laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material.

2. **De-compilation**. You may not remove any proprietary notices, labels, trademarks on the Software or documentation. You may not de-compile, disassemble or reverse engineer the Software.

3. **Support**. Software is provided under an AS-IS basis and without any support, updates or maintenance. Nothing in this Agreement shall require 29Palms Scenery Design to provide you with support or fixes to any bug, failure, mis-performance or other defect in the Software.

4. **Warranty**. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

5. **Group Buying**. You may **NOT** aggregate funds to Purchase the Software with one or more other parties. An example of this prohibited use is a website membership where members pool their money to make a single Purchase that is shared by the members of the group. Each such member must Purchase individually.

6. **No Obscene or Unlawful Use**. You may **NOT** use the Software for any defamatory, harassing, obscene, or racist purpose, or to infringe any party's Intellectual Property rights.

7. **License**. You may **NOT** use the Software in more than **one commercial** project / scenery / Add-On. If you want to include the Software in additional projects you should consider buying more licenses.

## Additional Comments

We do NOT take any responsibility for errors resulting from wrong set-ups and possible bugs in the software. Only YOU are responsible for a correctly operating program which you deliver to the end-users.

## Copyright