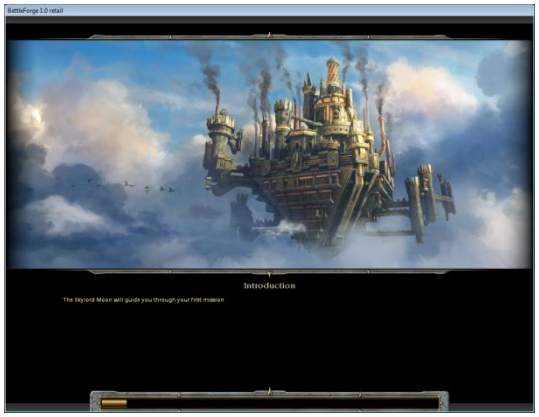
# BattleForge map customizations

**Please note:** All information given here is valid as of 15. October 2009 and may change without further notice.

It is assumed that you have already created a map and saved it in a path where BattleForge is able to load it. This document assumes that the map is called “FirstTest.map” and is stored in “My Documents\BattleForge\map”. If you save a map in the BattleForge Map Editor it automatically creates a sub-directory with the name of the map, in this example “My Documents\BattleForge\map\FirstTest”. This map specific directory is of high importance in this document. All of the information given is relative to this map specific directory. Keep this in mind.

## How to customize the map loading screen

Battleforge shows a standard loading screen for every custom created map (*left image*). You can customize the loading screen to look like a loading screen for a scenario map (*right image*) by just providing two files.


For this to work you must create a sub-directory called “ui” in the map specific directory. In this directory you must now put in an image and name it “artwork.jpg”. The image must be in JPEG format and should be in the resolution 1920x832. You may also provide a different resolution, but it is highly advised to keep the aspect ratio intact or the image will be displayed malformed. The fading that is visible on the left and right of the image is provided by the loading screen.

To change the map name and the map description displayed please provide the information in the file “description.xml” in the same directory. The structure of the file looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<mapdescription author="Tutorial">
	<description language="en">
		<mapname>FirstTest</mapname>
		<description>This is my first test map.</description>
	</description>
	<description language="de">
		<mapname>Erster Test</mapname>
		<description>Dies ist meine erste Map.</description>
	</description>
</mapdescription>
```

You can provide localized versions of both the map name and its description. Supported languages are English (“en”), German (“de”), French (“fr”) and Russian (“ru”). The loading screen will show the translated text for the language the user plays BattleForge in. If you provide no translation in the user’s language it will fall back to English. Furthermore, you can use the Author-Attribute of the mapdescription-Tag so everyone can see who made the map.
The file must be UTF-8 encoded.

## How to add outcries to your maps

Outcries can be added to your map by scripting. Please see the BattleForge map scripting documentation for a detailed explanation on generic scripting.
To show you how to announce an outcry the barely minimum of map scripting is used below. Feel free to enhance your more complex map scripts with the same functionality.


Create a directory called “script1” in the map specific directory. In there create a text file called “_Main.lua”. The content should look like this:

```
State
{
	StateName = "INIT",
	OnOneTimeEvent
	{
		Conditions =
		{
		},
		Actions =
		{
			MissionOutcry { Player = "ALL",
					Tag = "x", TextTag = "",
					Text = "Outcry text",
					DurationSeconds = 10,
					PortraitFileName = "default" },
		},
	};
};
```

The provided script will show an outcry with the content “Ourcry text” for 10 seconds to everyone who plays the map. If you want to have the outcry only for a specific player, provide the name in the ***Player*** parameter.
Setting ***PortraitFileName*** to “default” will show no portrait at all. You may show both character portraits provided by BattleForge as well as your own. Provided portraits are:

unit_blight_outcry
unit_brannoc_mutated_outcry
unit_brannoc_outcry
unit_enforcer_outcry
unit_engineer_outcry
unit_insect_outcry
unit_jorne_outcry
unit_mad_god_outcry
unit_moon_outcry
unit_mo_outcry
unit_mutating_frenzy_outcry
unit_queekqueek_outcry
unit_RavenCaptain_outcry
unit_RavenSailor_outcry
unit_rogan_kayle_outcry
unit_soldier_outcry
unit_striker_outcry
unit_umbabwe_outcry
unit_viridiya_outcry
unit_viridiya_spirit_outcry

If you want to show your own provide an image in TARGA format with the resolution 169x238, and place it into the folder “ui\outcry_portraits” in the map specific directory. Use the image’s filename without extension as parameter. Example: Put “my_own.tga” in “FirstTest\ui\outcry_portraits” and use parameter “my_own”.

Please note that you should ignore the ***Tag*** and ***TextTag*** parameters.

If you want to provide localized outcries you should create variables and check which game language is currently running through LUA. The definition and LUA code must be outside of a state description.

```
sOutcryText = "English (and default) outcry";

if (CScriptMain.GetLanguage() == "de") then
      sOutcryText = "Deutscher Outcry";
end

State
{
      StateName = "INIT",
      OnOneTimeEvent
      {
            Conditions =
            {
            },
            Actions =
            {
                  MissionOutcry { Player = "ALL",
                                  Tag = "x", TextTag = "",
```

```
                                    Text = sOutcryText,
                                    DurationSeconds = 10,
                                    PortraitFileName = "default" },
            },
      };
};
```

Supported languages are English (“en”), German (“de”), French (“fr”) and Russian (“ru”).

Also note that the user must have “Show subtitles” enabled in the game’s options or the text provided in the outcries will not be shown.
Unfortunately it is not possible to add custom voice-overs to your BattleForge map.

## How to add goals to your map

A goal is like the name suggests a task for a player to achieve or prevent. A goal can either be a simple task (“Do this”), a counter (“Get X of something”) or a timer (“Do until”).
To define goals for your map you have to provide their definition in a special XML file and trigger them through scripting.


The definition file must be called “goals.xml” and located in the map specific directory. Its structure looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<mapgoals>
      <goal
            tag="Goal_Test1" sortorder="5" timeleft_colorchange="0"
            musictype="off" positivetimer="1" goalletter="A"
            targettag="Goal_Test1_Target" hint="0">
                  <text language="en">English Goal</text>
                  <text language="de">Deutsches Ziel</text>
      </goal>
</mapgoals>
```

***Tag*** is the unique goal tag to identify this goal. Provide this name if you want to trigger changes on a goal through scripting.
***SortOrder*** gives you the possibility to ensure in which order goals are listed. With higher numbers goals get moved down the goal list. Please note that the sorting works as follows: first your own goals, then other goals. Each goals type is sorted by letter. If the letter collides (or none is set) the sort order value is used to determine the correct sorting.

***TimeLeft_ColorChange*** is only valid for timer goals and can be ignored otherwise. Gives the amount of time in seconds on when the timer changes its color to red to mark that it runs out soon.
***MusicType*** is also a timer only property and defines if none (“off”), “short” or “long” timer music should be played.
***PositiveTimer*** is the last in line to be timer goal specific. Set it to 1 if the timer is positive (a good goal, e.g. “defend until”) or 0 if the timer is negative (a bad goal, e.g. “time runs out”).
***GoalLetter*** specifies the letter to display both in the goal list as well as on the minimap (if target tag is valid).
***TargetTag*** is the unique name of an entity and connects as such the goal with that entity. The goal’s location is then displayed on the minimap and directional arrows are drawn on screen if the mouse hovers the goal icon in the goal list.
***Hint*** is a special parameter allowing for additional information to be defined which is shown as a tooltip if you hover the goal list. Set to 1 if the entry is meant as a hint.

You may leave out parameters you do not need for a goal. Default settings will be used then instead.

The text element allows for localized goal description. Supported languages are English (“en”), German (“de”), French (“fr”) and Russian (“ru”). If you do not provide a translation for the player’s language it will fall back to English.
The file must be UTF-8 encoded.

Each goal you define through this description file can now be triggered (started, stopped, changed) through special scripting commands. The command used to create the goal defines its type. Each goal can be of any type. Please note that the mentioned ***TaskTag***, ***CounterTag*** or ***TimerTag*** all correspond to the ***Tag*** given the goal in the definition file.

For generic goals (or “tasks”) you can use the following script commands:

```
MissionTaskSetActive { Player, TaskTag, TargetTag, Summary, Description }
MissionTaskSetSolved { Player, TaskTag, TargetTag, Summary, Description }
MissionTaskSetFailed { Player, TaskTag, TargetTag, Summary, Description }
MissionTaskRemove { Player, TaskTag }
```

Put them into a script state and you are ready to go:

```
State
{
      StateName = "INIT",
      OnOneTimeEvent
      {
            Conditions =
            {
            },
            Actions =
            {
                  MissionTaskSetActive { Player = "ALL",
                                         TaskTag = "Goal_Test1",
                                         TargetTag = "x",
                                         Summary = "Description for me",
                                         Description = "default"},
            },
      };
};
```

`MissionTaskSetActive`, `MissionTaskSetSolved` and `MissionTaskSetFailed` all have the same set of parameters, while `MissionTaskRemove` only supports ***Player*** and ***TaskTag***.

***Player*** is either ALL for every possible player or a specific player name.
***TaskTag*** is the unique goal identifier specified in the XML description file.
***TargetTag*** can be ignored and should always be "x".
***Summary*** is a reminder for you which goal this is. Same for ***Description***.

If you want to show a goal as a counter use the following script commands:

```
MissionCounterShow { Player, CounterTag, LocaTag, MaxValue }
MissionCounterSet { Player, CounterTag, Value }
MissionCounterHide { Player, CounterTag }
```

Each of the above commands support the ***Player*** and ***CounterTag*** parameters, specifying the player the goal is meant for and which goal to trigger. For `MissionCounterShow` also provide ***LocaTag*** (please ignore, set to "x") and ***MaxValue***. `MissionCounterSet` supports ***Value*** to update a counter value.

To show a goal as a timer please use these script commands:

```
MissionTimerStart { Player, TimerTag, LocaTag, Seconds, Minutes }
MissionTimerChange { Player, TimerTag, LocaTag, TimerTagOld }
MissionTimerStop { Player, TimerTag }
```

Each commands support ***Player*** and ***TimerTag*** to identify the goal target player and the goal. `MissionTimerStart` also needs ***LocaTag*** (please ignore, set to "x") as well as ***Seconds*** and ***Minutes*** to specify the amount of time to count down. With `MissionTimerChange` you are able to change a timer goal from one to another without loosing its current count down value. For this please provide ***TimerTagOld*** to identify the goal timer to change.

## How to display a custom background for the minimap

Customizing the background of the minimap is simple. Please provide a JPEG-image and name it "minimap.jpg". It must be copied to the "ui" sub-folder of the map specific directory. The recommended resolution is 586x586. If you provide a non-quadratic image, the image will be displayed distorted and / or cropped.