ShoK GUI Editor Guide

Inhaltsverzeichnis

Arbeitsverzeichnis

Beim ersten Start des Editors muss das Arbeitsverzeichnis festgelegt werden. Dieses sollte ein leerer Ordner in einem Laufwerk mit mindestens 50 MB Speicher sein. Nach dem auswählen werden alle vom Spiel verwendeten Grafiken aus den .bba Dateien eures Spiels geladen und in diesem Ordner deponiert.

Zusätzlich werden die originalen GUI Definitionen des Spiels im Arbeitsverzeichnis unter *menu\projects* abgespeichert. Es handelt sich dabei um die *ingame.xml* und *mainmenu.xml*

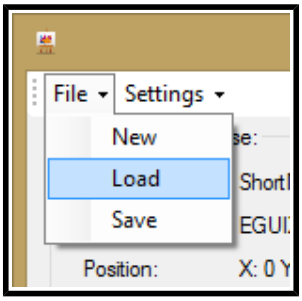
Das Arbeitsverzeichnis bildet die interne Baumstruktur des Dateisystems von Siedler 5 ab. Das heißt wenn später eigene Grafiken in die .s5x geladen werden müssen sie exakt in dasselbe Verzeichnis wie im Arbeitsverzeichnis.

Es müssen sich also alle selbst erstellten oder kopierten Grafiken, welche für neue Widgets verwendet werden sollen im Arbeitsverzeichnis befinden!

Originale GUI Laden

Nach dem erstellen des Arbeitsverzeichnisses öffnet sich der GUI Editor. Dieser ist bis auf das sogenannte *Root Widget* jedoch aber leer. Bei diesem Widget handelt es sich um ein *Container Widget* in welches weitere Widgets eingefügt werden können.

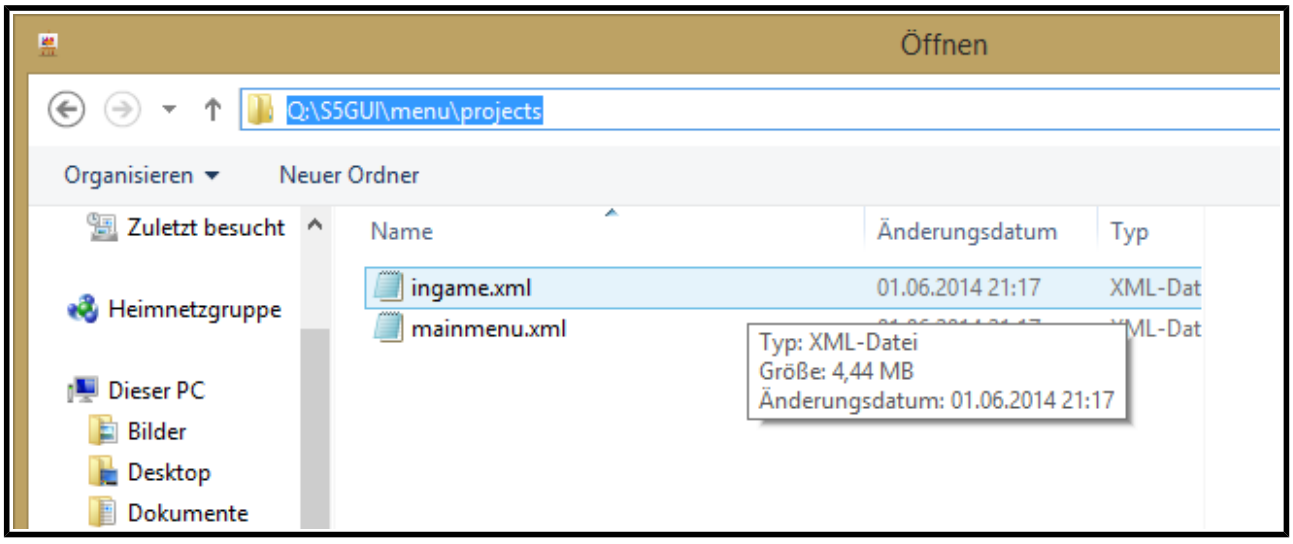
Natürlich kann man sofort mit dem Erstellen von neuen Widgets beginnen, was jedoch nicht unbedingt klug ist, da man ja nicht die gesamte GUI des Spiels neu designen will.



Aufgrund dessen wird nun die bereits vorhanden GUI in den Editor geladen. Man muss sich dabei zwischen *Ingame* und *MainMenu* entscheiden wobei Letzteres vorerst nicht empfohlen wird.

Es wird die ingame.xml aus dem Arbeitsverzeichnis geladen.

(In diesem Fall ist das Arbeitsverzeichnis: Q:\S5GUI)



Die GUI wird so wie in der .xml definiert in den Editor geladen, dabei werden jene Widgets welche von Spielstart aus sichtbar sind eingeblendet.

Buttons, Texte und Grafiken werden wie im Spiel dargestellt. Die unterschiedlichen Widgets können in der Baumstruktur auf der linken Seite des Editors durch An- und Abhaken ein- und ausgeblendet werden.

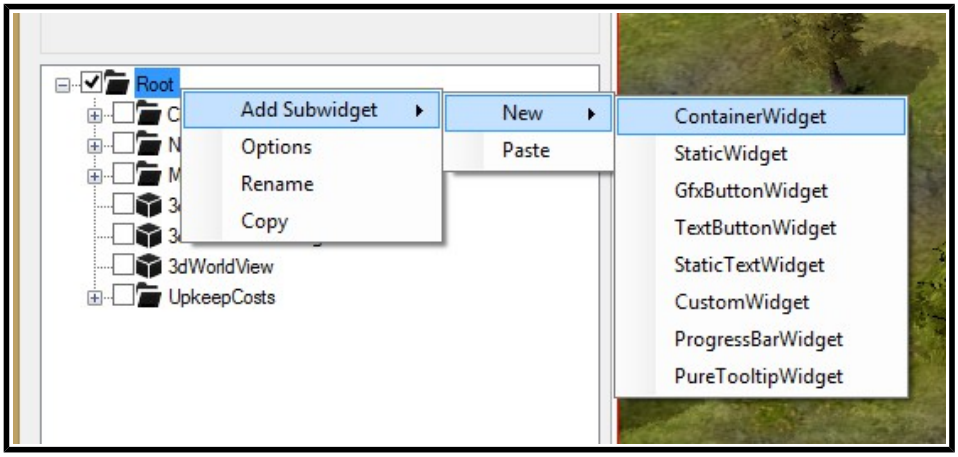
Um die Grundlagen zu verstehen werden die Ganzen Widgets des Spiels nicht benötigt und so haken wir jetzt alle Widgets bis auf das *Root Widget* ab.

Das erste eigene Widget

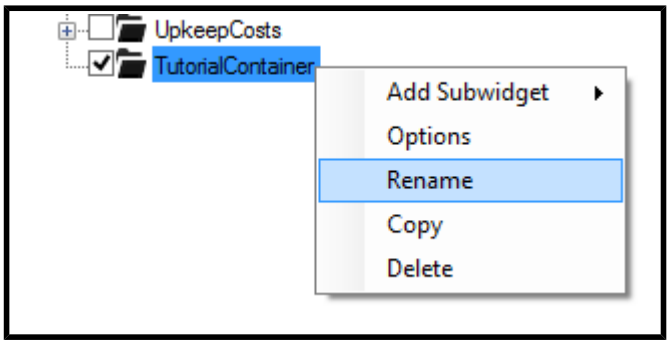
Widgets befinden sich immer in einem übergeordneten Container Widget und so können auch nur in diese, in der Baumstruktur mit einem Ordner-Symbol gekennzeichneten Widgets, Widgets eingefügt werden.

Um dies zu bewerkstelligen muss in der Baumstruktur ein Rechtsklick auf ein Container Widget gemacht werden. Man kann bei *Add Subwidget* zwei Möglichkeiten entscheiden. *New* und *Paste*, mit Ersterem kann ein komplett neues Widget erstellt werden, mit Letzterem wird ein zuvor durch *Copy* kopiertes Widget in den Container eingefügt.

Wir erstellen erst ein neues Container Widget.



Und benennen es zu "TutorialContainer" um.



Das erste Widget ist erstellt auch wenn es weder viel kann noch wirklich sichtbar ist.

Position und Größe von Widgets

Container sind immer unsichtbar, werden aber beim Bewegen der Maus über sie mit einem roten Rahmen gekennzeichnet.

Nach dem neu Erstellen eines Widgets erscheint dieses immer in der linken oberen Ecke des zugehörigen Containers. Im Falle unseres TutorialContainers ist es die linke obere Ecke des kompletten GUI-Views.

Um das Widget zu bewegen muss die linke Maustaste gedrückt und gleichzeitig über den GUI-View gefahren werden

Die Größe des Widgets kann verändert werden indem man die maus in die rechte untere Ecke des Widgets bewegt. Der Mauzeiger ändert sich und durch das Halten der linken Maustaste und Bewegen der Maus kann die Größe des Widgets verändert werden.

So bringen wir den TutorialContainer auf eine angemessene Größe und können Fortfahren

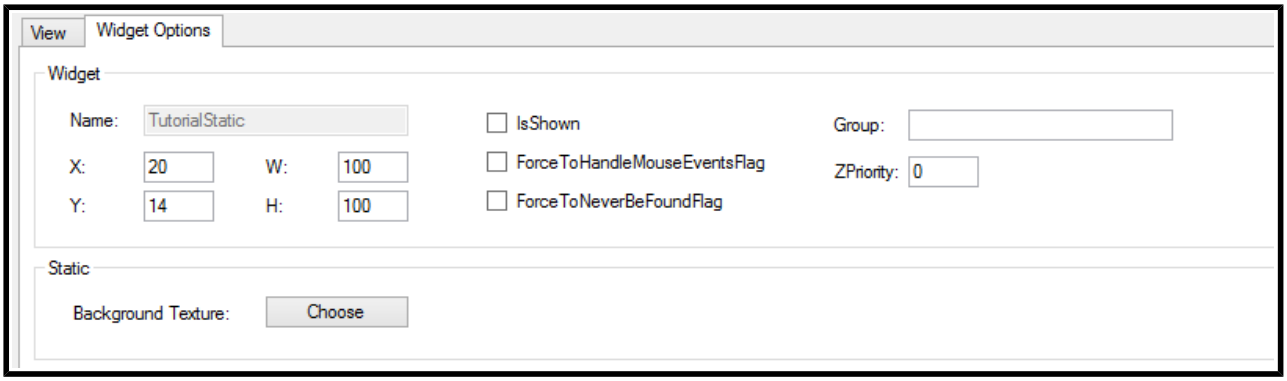
Laden von Grafiken

Die einfachste Methode um eigene Grafiken ins Spiel zu laden ist ein Static Widget zu erzeugen und die Hintergrundgrafik zu verändern. Eine Grafik im Spiel besteht immer aus einem Bild (möglichst png) und einer Tönung im RGB Format.

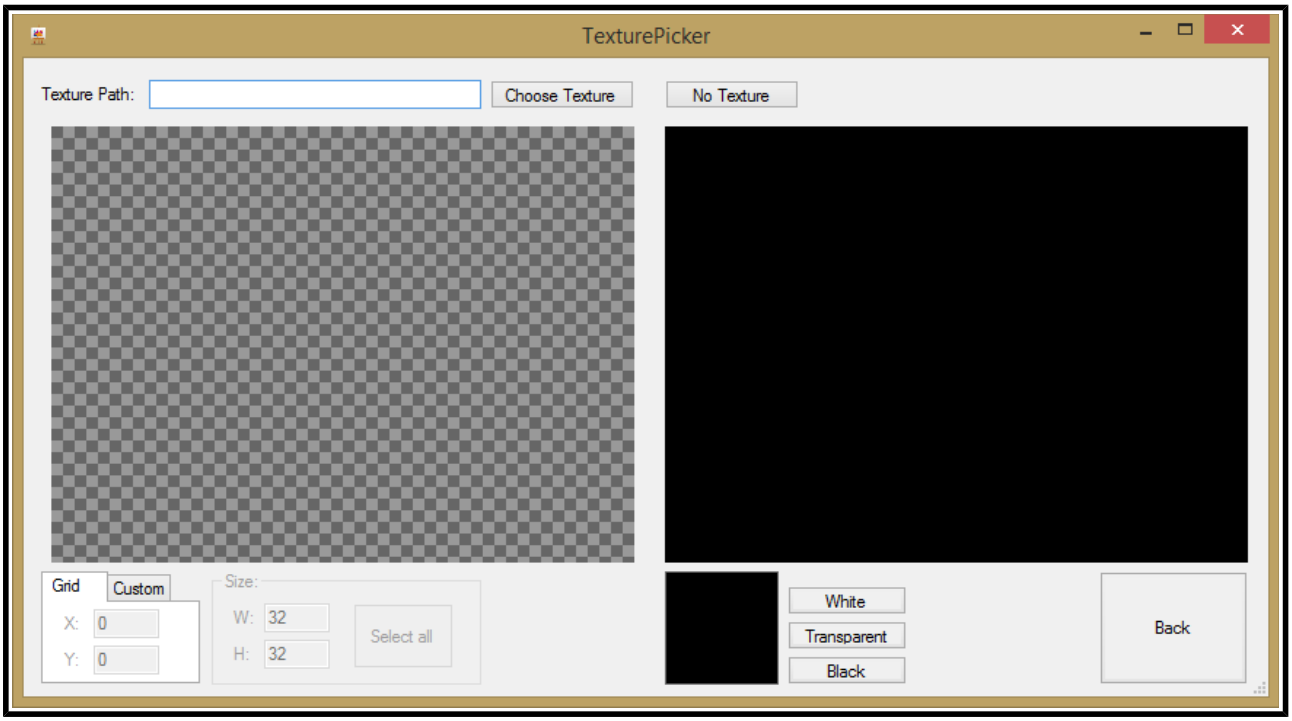
Wir fügen unserem TutorialContainer nun ein Static Widget hinzu. Welches als schwarzer Fleck in der linken oberen Ecke des Containers erscheint und wir erst einmal zu TutorialStatic umbenennen.

Daraufhin vollführen wir einen Doppelklick auf das schwarze quadratische Static Widegt und finden uns in dessen Optionen wieder. Im oberen Teil befinden sich immer die Optionen für das Widget allgemein und darunter je nach Widget-Typ spezifische Optionen.

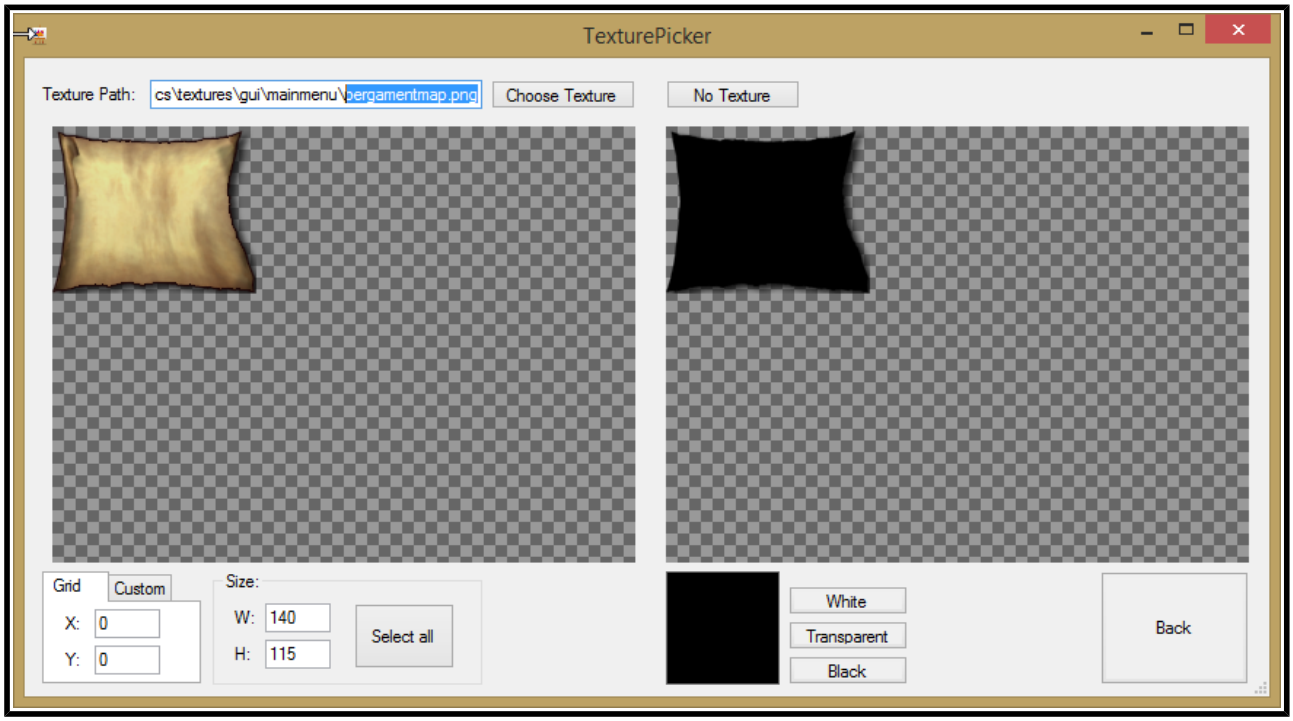
Das Static Widget selbst besitzt als spezifische Option nur die Auswahl einer Hintergrund-Grafik. Der Button *Choose* wird nun betätigt um diese auszuwählen.



Dadurch öffnet sich nun der sogenannte *TexturePicker* mit welchem man eine Grafik aus dem Arbeitsverzeichnis laden kann, einen bestimmten Teil davon ausschneiden und schlussendlich auch noch einfärben kann.



Mithilfe von *Choose Texture* wählen wir nun die *pergamentmap.png* welche wir im mainemenu Ordner finden aus.



In der linken unteren Ecke befinden sich die Koodinaten sowie die Breite u. Höhe des Ausschnitts aus dem Bild links, welche im rechten Bild dargestellt und schlussendlich auch verwendet werden.

Das heißt: Das linke Bild ist das Originalbild und das rechte die Vorschau wie es aussehen wird.

Über das Bild wird mommentan eine Farbe mit den Werten *Rot = 0, Grün = 0, Blau = 0, Alpha = 0* gelegt. Beim einfärben einer Textur wird jedes Pixel des Bildes mit der angegebenen Farbe eingefärbt dadurch wird unsere Pergament Map so gut wie schwarz.

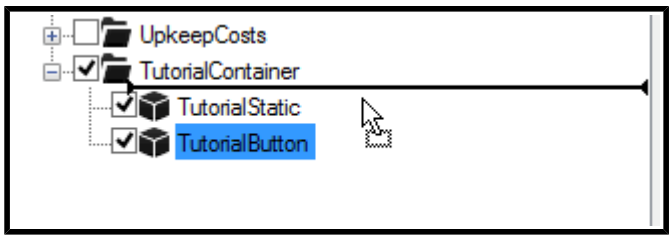
Um keinerlei Einfärbung zu erzielen muss die Farbe Weiß verwendet werden. Die drei Buttons, White, Transparent und Black sind dazu da um diese oft verwendeten Farben schnell einzustellen. Genauere Farbeinstellungen können durch das Klicken auf die Farbvorschau vorgenommen werden.

Wir betätigen nun erst den Button White um keine Färbung zu haben und anschließend Back da wir mit unserer Wahl der Textur fertig sind. Daraufhin wechseln wir wieder in den Game-View und erhalten dieses Bild:

Buttons

Wir erstellen in unserem TestContainer einen neuen Button welchem wir einen passenden Namen geben,und ihm in den Optionen die Größe 32x32 zuweisen. Eine übliche Größe für einen Button in S5 da auch die meisten Grafiken diese Größe besitzen.

Wir stellen fest das der Button unter unserem zuvor erstllten Static Widget gezeichnet wird. Um die Anordnung der Widgets zu veröndern können diese in der Baumstruktur verschoben werden. Das erste Element in einem Container wird dabei immer auch als erstes gezeichnet.



Den neuen Button platzieren wir mittig auf unserer Pergament Map.

Diese Button ist jetzt schon voll funktionstüchtig, es gibt jedoch einige Einstellungen die vorgenommen werden müssen damit er auch wirklich von Nutzen sein kann.

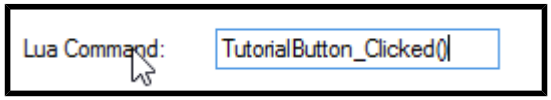
Dabei gibt es die verschiedenen Texturen der States in dem sich der Button befinden kann (Normal, Hover, Pressed, Disabled, Highlighted), der Callback des Buttons sowie der Tooltip welcher ein jeder Button auslösen kann.

Wir begeben uns in die Optionen des Buttons und sehen deutlich mehr Optionen als zuvor. Die meisten davon sollten Selbsterklärend sein, andere widerrum auch nicht wirklich von nützlich. Am wichtigsten ist dabei das *Lua Command*, *der Tooltip* und die *Button Textures*.

Lua Commands

Diese sind Befehle die bei entweder einen Callback oder eine Update Funktion darstellen. Sie können Argumente enthalten und werden vom Spiel aus aufgerufen.

Update Commands sind deutlich schneller als Jobs, da sie jeden Frame aufgerufen werden. Mit diesen sollte man es jedoch nicht übertreiben und auch keine all zu komplizierten Abfragen vollziehren.



Wir tragen ein von uns gewähltes Lua Command in die Zeile ein und kommen nun zum Tooltip.

Tooltips

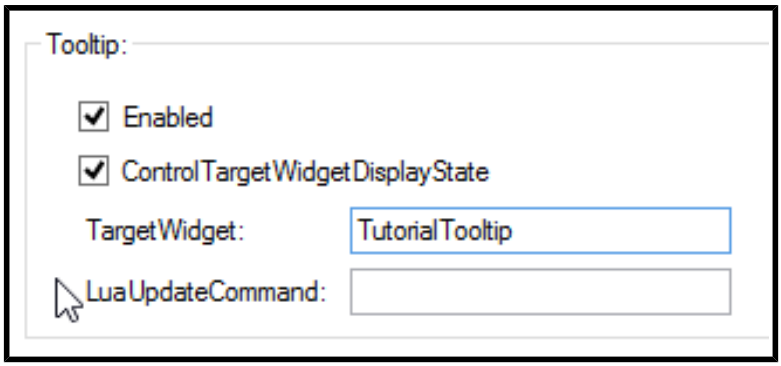
Tooltips können entweder von Buttons oder von PureTooltipWidgets aufgerufen werden wobei Letzteres nichts anderes außer dies tut.

Ein Tooltip bleibt inaktiv solange das Enabled Flag nicht gesetzt ist.

Tooltips beinhalten ein Target Widget welches beim Eintritt der Maus angezeigt werden kann, wenn das entsprechende Flag (ControlTargetWidgetDisplayState) gesetzt ist. Tritt die Maus aus dem Widget aus wird das Target Widget wieder ausgeblendet.

Zusätzlich kann ein Lua Update Command gesetzt werden welcher kontinuierlich Nach Eintritt der Maus aufgerufen wird. Mithilfe dieses Commands können Text Widgets innerhalb des Target Widgets angepasst werden.

Für unseren Button setzen wir folgende Einstellungen:



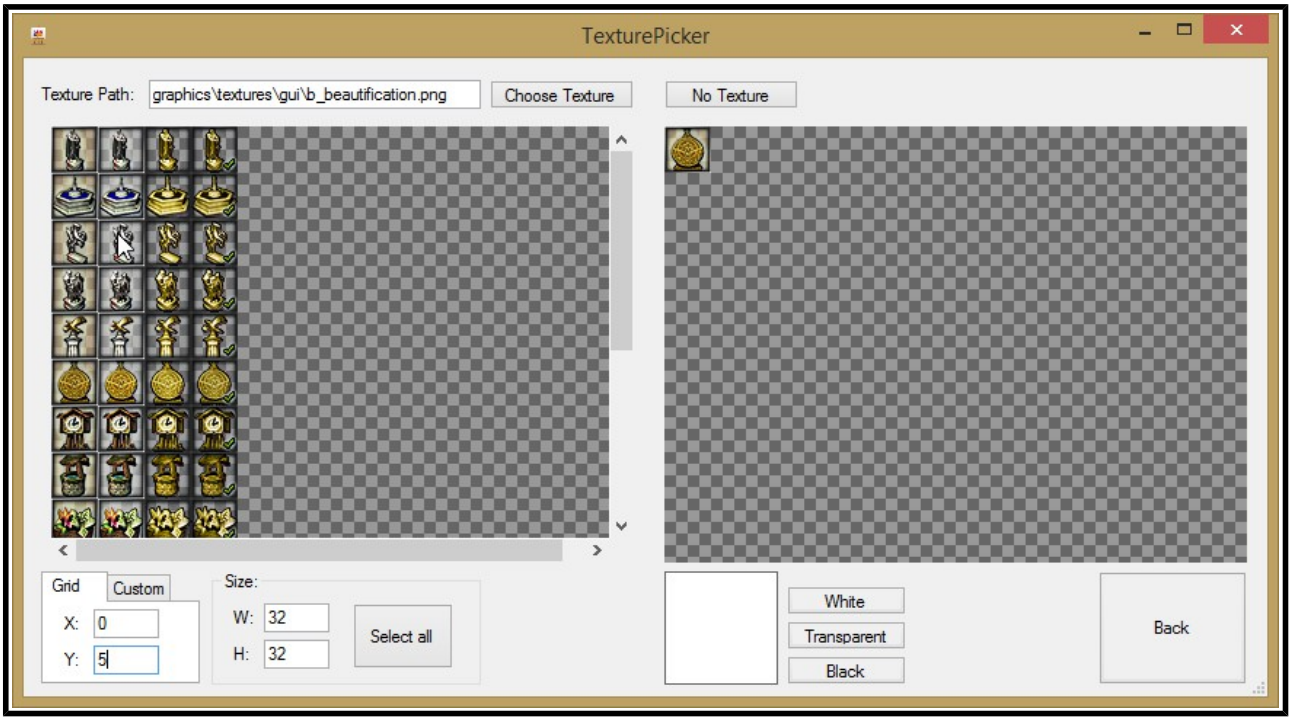
Ausschneiden von Texturen

Wir beginnen mit dem wählen der Button Textur "Normal".

Dazu wählen wir ein passende Grafik mit mehreren Buttons aus. Um in der großen Graik nun aber einen kleinen Ausschnit für unseren Button zu erlangen verwenden wir das Grid. Wir stellen die Breite sowie die Höhe auf 32 Pixel.

Mithilfe der Grid Einstellungen können wir nun in 32er Schritten über die Grafik springen und so ganz einfach unsere gewünschte Textur aussuchen.

Die Texturen für einen Button stehen in diesen Grafiken immer in einer Reihe, daher suchen wir uns eine davon aus und entfernen die Färbung des Buttons.



Der Normal State ist in den Grafiken immer ganz Links d.h. Grid X = 0.

Wir fahren fort mit den nächsten Button States und laden immer die Grafik stellen die Breite und höhe ein un verändern unser X im Grid so das er die nächste Textur bekommt.

Die Disabled ist in der Grafik oben nicht vorhanden, also muss sie selbst irgendwie gebastelt werden. Z.B. durch das einfärben des Pressed States mit einer leicht grauen Farbe.

So werden unserem TutorialButton die einzelnen Texturen für die verschiedenen States zugewiesen.

Icon Material bei GfxButtons

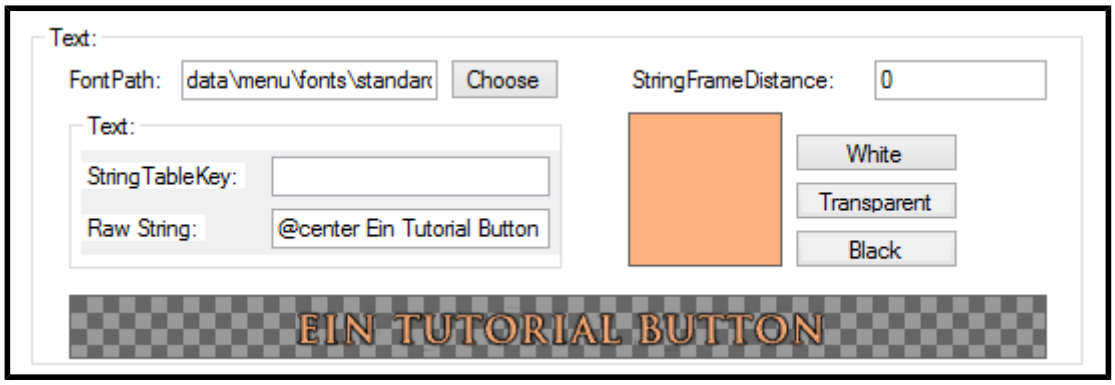
Das Icon Material beim GfxButton ist eine spezielle Textur welche einfach über jeden eingestellten State gezeichnet wird. Dies macht nicht oft Sinn und so kann die Farbe der Textur einfach auf Transparent gesetzt werden.

Einen Tooltip erstellen

Ist sehr einfach. Wir beginnen mit einem neuen Container in welchen wir ein StaticWidget und ein StaticTextWidget einfügen. Den Container nennen wir wie im Button vorher definiert "*TutorialTooltip*".

Dem StaticWidget geben wir eine Textur und verwenden ihn als Hintergrund für den Tooltip.

Das Static Text Widget hat einige neue selbsterklärende Einstellungen und dazu noch eine Möglichkeit die Schriftart, Schriftfarbe sowie einen Raw String zu setzen welcher den normalen Text im Widget angibt. Der Text kann natürlich später beliebig durch die XGUIEng Funktionen verändert werden.



Der Text und Schriftart wird nach belieben gesetzt und ein kurzer Text in den Raw String eingegeben.

Fertig ist der Tooltip und wir erhalten folgendes Bild:

IsShown Flag

Ob die Widgets im Spiel tatsächlich angezeigt werden hängt nicht vom GUI Editor ab sondern wird durch eine Flag in den Optionen definiert.

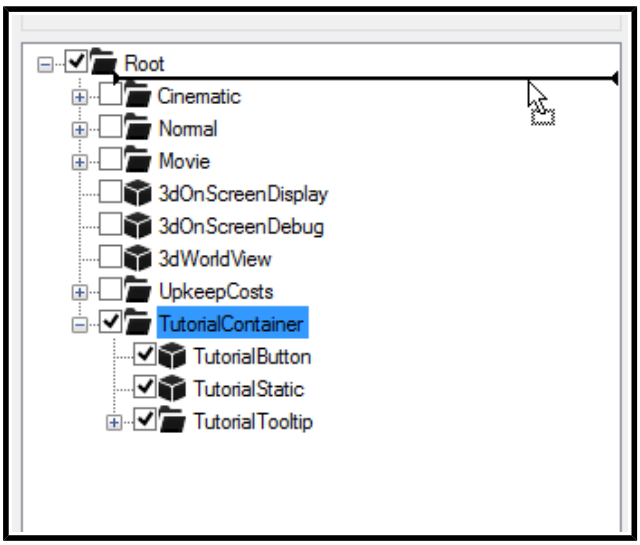
Für all unsere erstellten Widgets, bis auf den TutorialContainer (die Widgets im Container jedoch schon) setzen wir diese Flag nun.

Somit hätten wir unser eigenes neues Widget vollendet.

Einbinden ins Spiel

Nach der Fertigstellung des TutorialWidgets kommen wir zum wichtigsten Teil. Das Einbinden der neuen GUI ins Spiel.

Um im Spiel unser Widget jedoch überhaupt zu sehen setzen wir es in der Zeichenreihenfolge vor alle Anderen.

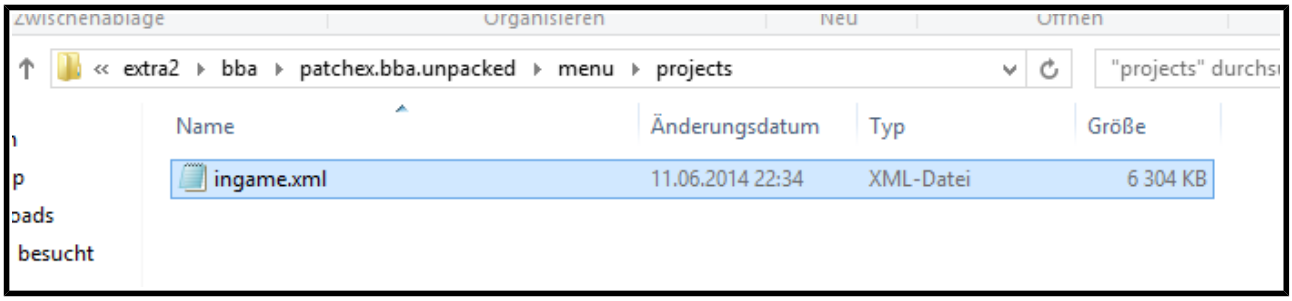


Die neue GUI xml wird nun an einen beliebigen Ort gespeichert und mithilfe des bbaTools in eine Map oder die patchex.bba geladen werden.

Wird die GUI xml aber in die Map gepackt muss im Skript das S5Hook enthalten sein mit dessen Hilfe man eine solche Date aus einem Pfad heraus laden kann. Dabei kann auch nach Belieben ein Testpfad angegeben werden um die Datei nicht immer zu verpacken.

```
function On_S5HookLoaded()
    S5Hook.LoadGUI("Q:/S5GUI/heroinventory.xml")
    S5Hook.ReloadEntities()
end
```

Dabei darf man keinesfalls vergessen die richtige Ordnerstruktur in der bba oder s5x Datei wiederherzustellen und die xml zu ingame oder mainmenu umzubenennen:



Falls im neuen Widgets selbst erstellte Grafiken verwendet wurden müssen diese ebenfalls wie im exakt selben Verzeichnis wie im Arbeitsverzeichnis in der bba oder s5x gespeichert werden.

Andere Widgets

Bis auf das Custom Widget basieren die restlichen im Guide nicht gezeigten Widgets auf den selben Prinzipien wie die Gezeigten und es sollte daher kein Problem sein diese zu verstehen.

Anderenfalls gibt es zu den bestehenden Widgets auch eine weitere Dokumentation welche auf b0bby.tk erhältlich ist.