

# **Studienprojekt**

Leveldesign im Spiel  
"Die Siedler - Das Erbe der Könige" -  
Dokumentation

von  
**Markus Rumpel**

Matrikelnummer 754378  
maruit00@hs-esslingen.de

Wintersemester 2023/2024

Prüfer HS-Esslingen: Harald Melcher

Projektzeitraum: 25.09.2023 – 10.02.2024

# Inhaltsverzeichnis

Eidesstattliche Erklärung:.....	3
Vorwort:.....	4
Voraussetzungen:.....	5
Aufsetzen des modifizierten Karteneditors:.....	6
Spielversion History Edition (Steam, Ubisoft Connect/Uplay).....	6
Spielversion Gold Edition (von ~ 2005).....	12
Der Karteneditor:.....	19
Grundwerkzeuge:.....	19
Selektions-Explorer:.....	20
Menü-Leiste:.....	20
Variables Werkzeug-Menü:.....	22
Das Erstellen der Karte:.....	23
Ideen und Konzepte:.....	23
Erstellen der leeren Karte:.....	24
Grundstrukturen:.....	25
Der ländliche Teil der Karte:.....	27
Entstehung einer Geschichte:.....	28
Der Bau der Stadt:.....	29
Das Banditenlager:.....	42
Der Verlauf der Karte:.....	43
Technische Spezifikationen:.....	43
Der Einstieg in das Skript:.....	45
Kapitel 1 Auf der Suche nach etwas:.....	48
Kapitel 2 Die große Stadt:.....	52
Kapitel 3 Den Einwohnern helfen:.....	55
Kapitel 4 Der Prinzessin helfen:.....	60
Kapitel 5 Die Offenbarung der Prinzessin:.....	63
Kapitel 6 Die Banditen vertreiben:.....	66
Kapitel 7 Ende gut, alles gut:.....	68
Testphase:.....	70
Aufsetzen der Testumgebung:.....	70
Technische Testphase:.....	71
Spielerische Testphase:.....	72
Karten-Info anpassen:.....	74
Spielen der Karte:.....	75
Exkurs Erweitertes Mapping:.....	76
Eigene Models/Entities/Animationen:.....	76
Eigenes Graphical User Interface:.....	76
Eigene Effekte:.....	76
Modloader:.....	76
Eigene Mechaniken:.....	77
Zusammenfassung:.....	78
Quellen/Verweise:.....	79

## **Eidesstattliche Erklärung:**

Hiermit erkläre ich, Markus Rumpel, dass ich die vorliegende Arbeit selbständig verfasst habe, dass ich sie zuvor an keiner anderen Hochschule und in keinem anderen Studiengang als Prüfungsleistung eingereicht habe und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.

Fellbach, 09.02.2024, *Markus Rumpel*

---

Ort, Datum, Unterschrift

## **Vorwort:**

Bei diesem Projekt handelt es sich um eine detaillierte Dokumentation über das Erstellen einer Spielerkarte im Spiel „Die Siedler – Das Erbe der Könige“. Es wird genauer auf das Mapping mit dem Editor, sowie dem Schreiben des Lua-Skripts eingegangen und welche Besonderheiten jeweils vorliegen. Dabei werden sowohl die Grenzen des Möglichen erläutert, sowohl auf Hardware-, als auch auf Software-Ebene. Desweiteren wird auf das Aufsetzen der benötigten Umgebung eingegangen.

## Voraussetzungen:

Als erstes wird eine der beiden Spielversionen gebraucht:

- Die alte Gold Edition mit dem Hauptspiel und den beiden Erweiterungen „Nebelreich“ und „Legenden“. Die Version kann auch aus den einzelnen Teilen bestehen statt der kompakten Edition. In dem Fall gilt sie auch als eine Gold Edition.
- Die History Edition, in der das Hauptspiel und beide Erweiterungen enthalten sind.

Zusätzlich wird der [S5Updater](#) von mcb, sowie der MPUpdater von Kimichura benötigt. Der MPUpdater kann vom [Discord](#) von Kimichura heruntergeladen werden.

**Vorsicht:** Die Dateien können als Viren erkannt werden. Der Grund hierfür ist, dass sie auf der Basis von DLL-Injection und weiteren Downloads aus dem Internet basieren und deshalb als False-Positive erkannt werden.

# Aufsetzen des modifizierten Karteneditors:

Das Aufsetzen des Karteneditors findet je nach Version auf unterschiedlichen Wegen statt. Folgend liegen zwei Anleitungen vor, die Schritt für Schritt durch die Installation von diesem durchgehen. Die Bilder vom S5Updater können hierbei ein wenig abweichen basierend auf der jeweiligen Version.

## Spielversion History Edition (Steam, Ubisoft Connect/Uplay)

In dieser Anleitung wird das Vorgehen zur Installation von Kimichuras Multiplayer Client für „Die Siedler Das Erbe der Könige – History Edition“ schrittweise erläutert. Hier wird davon ausgegangen, dass das Spiel nicht manipuliert wurde und bereits über Uplay/Ubisoft Connect installiert wurde.

### Vorbereitung: Herunterladen aller wichtigen Dateien

Zuerst müssen die wichtigen Dateien dafür heruntergeladen werden. Diese sind

- [S5Updater](#) von mcb
- MPUpdater von Kimichuras [Discord](#) im jeweiligen Channel (ganz oben)

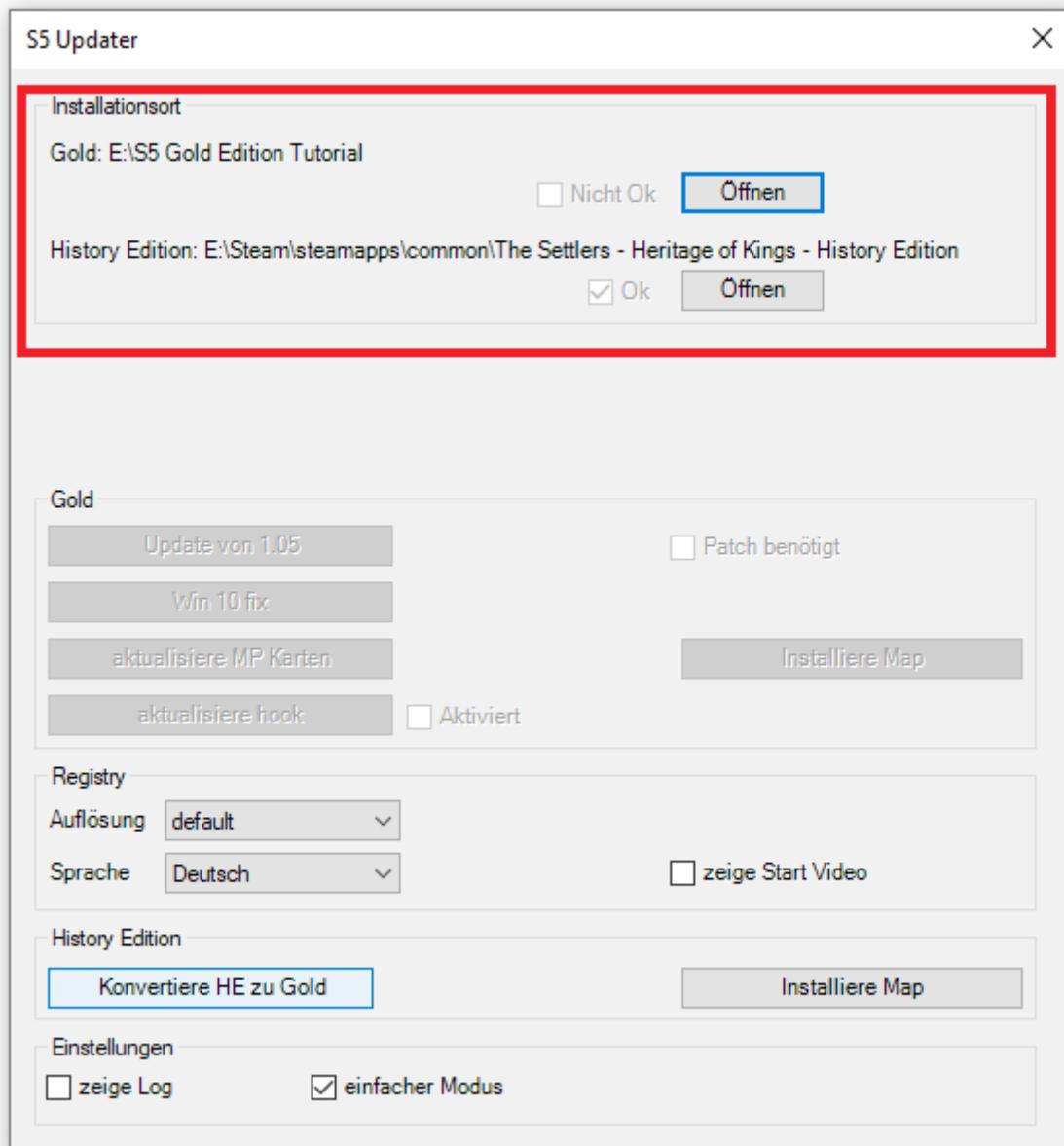
Name	Änderungsdatum	Typ	Größe
de	22.08.2021 14:19	Dateiordner	
AutoDownloader	03.07.2021 22:22	Anwendung	11 KB
bbaTools5	08.02.2021 20:42	Anwendung	468 KB
git2-106a5f2.dll	11.12.2019 08:38	Anwendungserwe...	919 KB
LibGit2Sharp.dll	11.12.2019 08:50	Anwendungserwe...	411 KB
S5Updater	03.07.2021 22:22	Anwendung	59 KB

Der S5Updater wird in einen Ordner der Wahl entpackt.

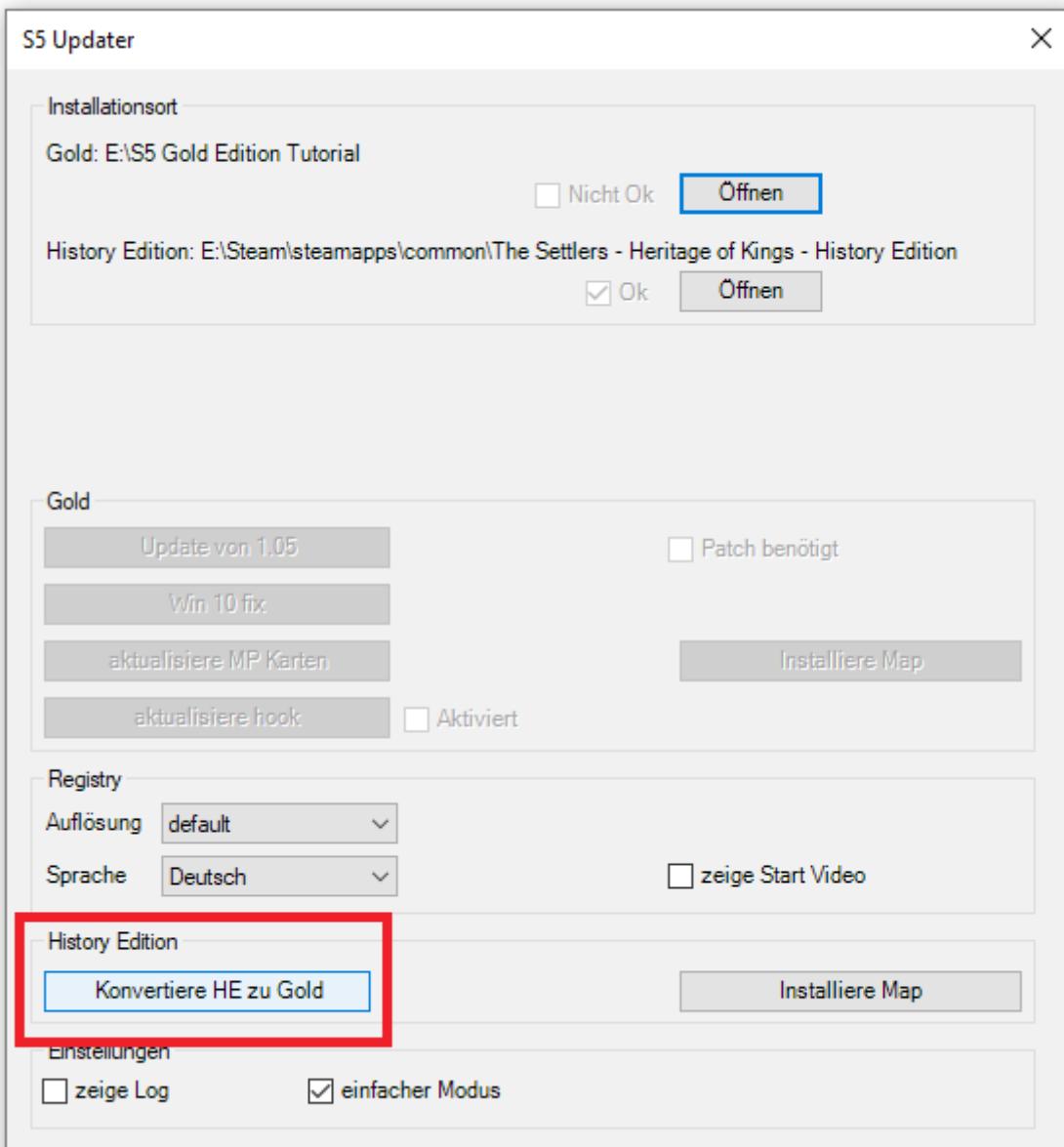
Der MPUpdater wird später benötigt. Dieser wird am besten irgendwo griffbereit abgelegt, bis er gebraucht wird.

## Schritt 1: Die History Edition zur Gold Edition konvertieren

Für die Gold Edition wird ein leerer Ordner erstellt, in den die konvertierte Edition kopiert wird. Nun wird der S5Updater geöffnet. Für die History Edition wird das Hauptverzeichnis eingetragen und für die Gold Edition wird der leere Ordner ausgewählt.



Anschließend wird der Knopf „Konvertiere HE zu Gold“ betätigt. Dabei öffnet sich ein kleiner Fortschrittsbalken. Je nach Internet- und Computerleistung kann dies einige Minute dauern.



Nachdem Der Prozess abgeschlossen und der Fortschrittsbalken verschwunden ist, wird nur noch mit der Gold Edition gearbeitet. Die konvertierte Version wird in Zukunft als Gold Edition bezeichnet.

Der S5Updater kann nun geschlossen werden. Er wird gegebenenfalls später für den LuaDebugger nochmal benötigt.

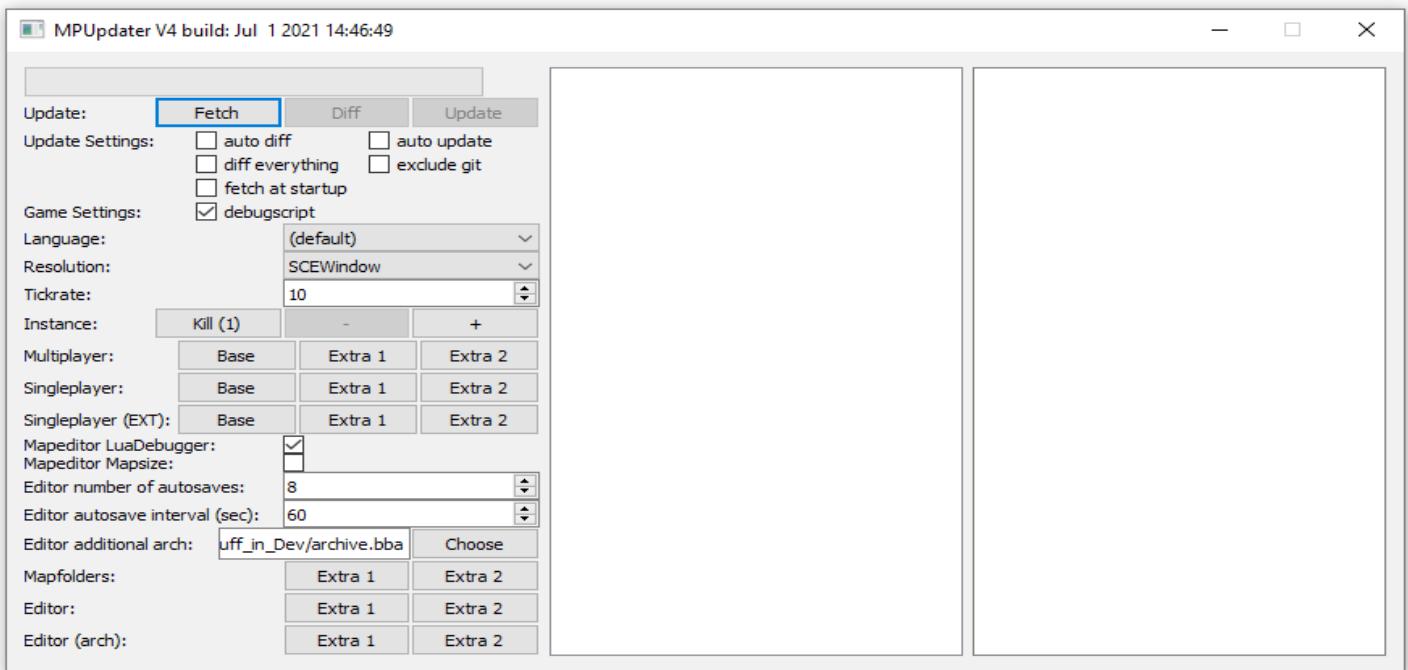
## Schritt 2: Installieren des MPUpdater

Nun wird Kimichuras Client installiert, der zuvor von seinem Discord heruntergeladen wurde. Hierzu werden alle Dateien in das Hauptverzeichniss der Gold Edition gepackt.

Wichtig: Das Hauptverzeichniss soll in dem jeweiligen Anti-Vierenprogramm als Ausnahme eingestellt werden. Es ist gut möglich, dass die MPUpdater.exe als Virus erkannt wird. Hierbei handelt es sich um einen False-Positiv; eine Falschmeldung. Durch das Erstellen einer Ausnahme unterbindet man dieses Verhalten.

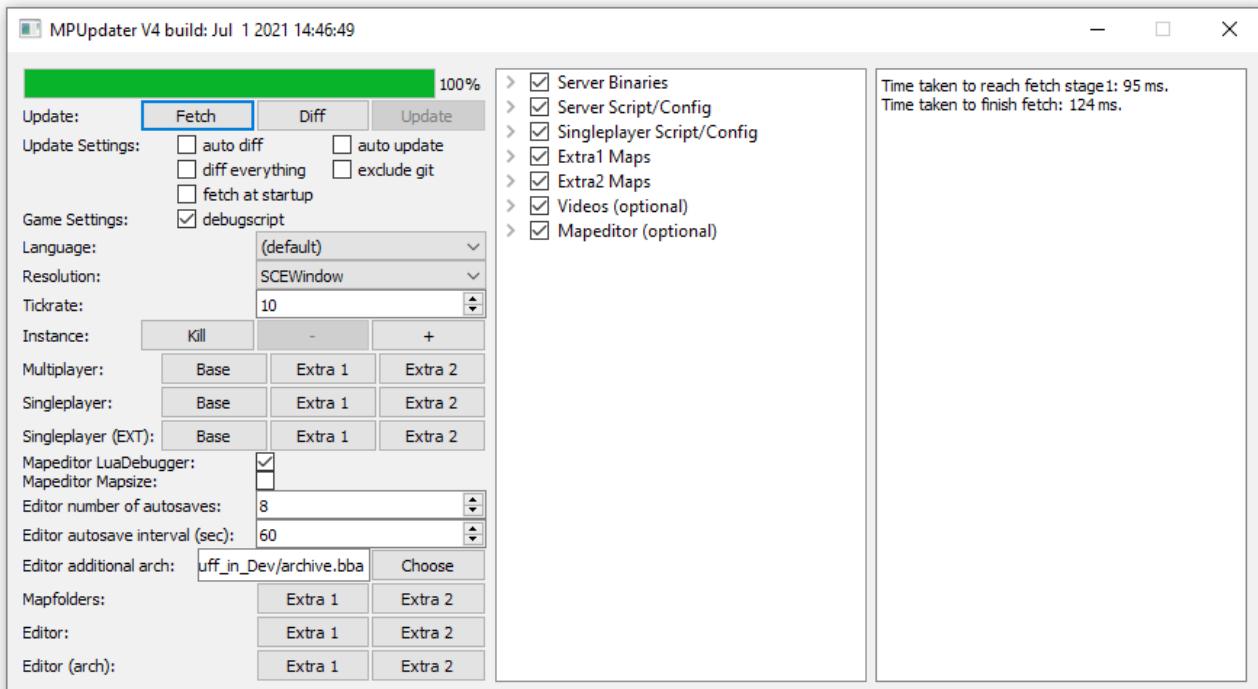
Name	Änderungsdatum	Typ	Größe
base	22.08.2021 14:32	Dateiordner	
bin	22.08.2021 14:34	Dateiordner	
extra1	22.08.2021 14:37	Dateiordner	
extra2	22.08.2021 14:37	Dateiordner	
platforms	22.08.2021 14:40	Dateiordner	
styles	22.08.2021 14:40	Dateiordner	
support	22.08.2021 14:37	Dateiordner	
install.vdk	29.12.2019 17:49	VDK-Datei	1 KB
lbeay32.dll	27.12.2019 13:57	Anwendungserwe...	1.241 KB
MPUpdater	01.07.2021 14:46	Anwendung	299 KB
Qt5Core.dll	27.12.2019 13:57	Anwendungserwe...	4.778 KB
Qt5Gui.dll	27.12.2019 13:57	Anwendungserwe...	5.077 KB
Qt5Network.dll	27.12.2019 13:57	Anwendungserwe...	962 KB
Qt5Widgets.dll	27.12.2019 13:57	Anwendungserwe...	4.326 KB
rastertable	27.02.2021 14:41	Datei	1 KB
settlershok.bin	29.12.2019 17:48	BIN-Datei	1 KB
ssleay32.dll	27.12.2019 13:57	Anwendungserwe...	271 KB

Anschließend wird die MPUpdater.exe als Administrator ausgeführt und folgendes Fenster (für die Version 4 des Updaters) sollte erscheinen.

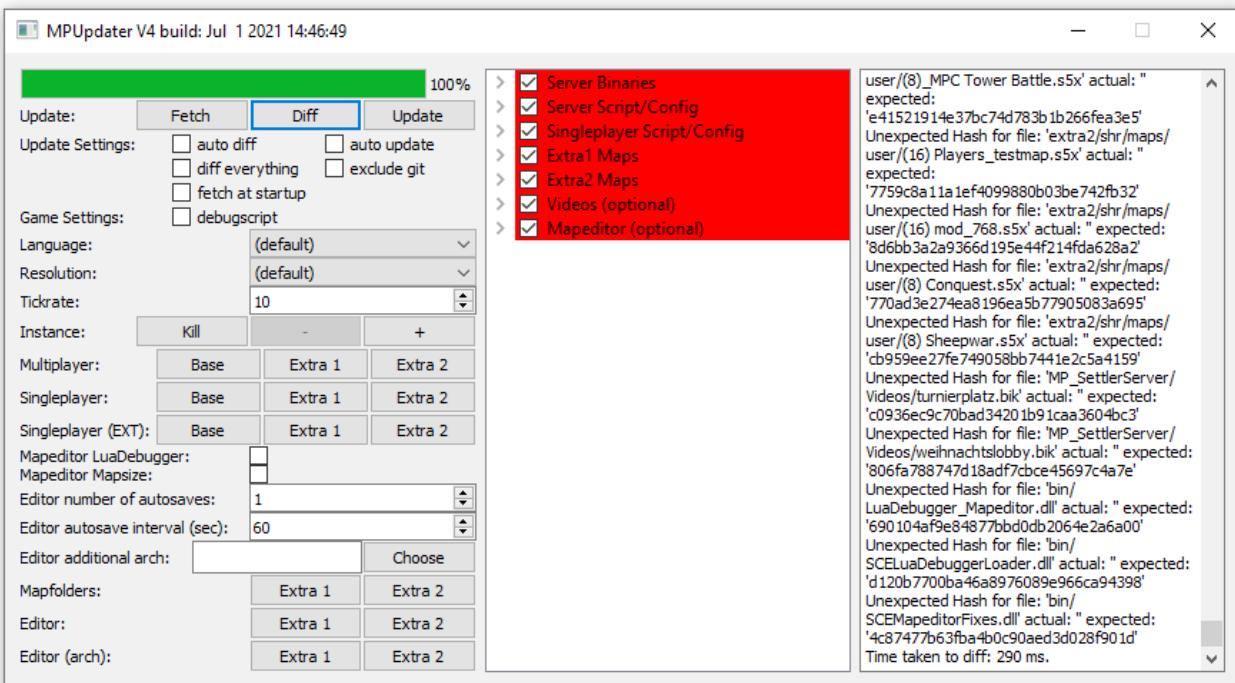


Für den Karteneditor werden die Reihen „Update“ und „Editor“ benötigt.

Zuerst „Fetch“. Damit wird geschaut was für Dateien es momentan gibt.

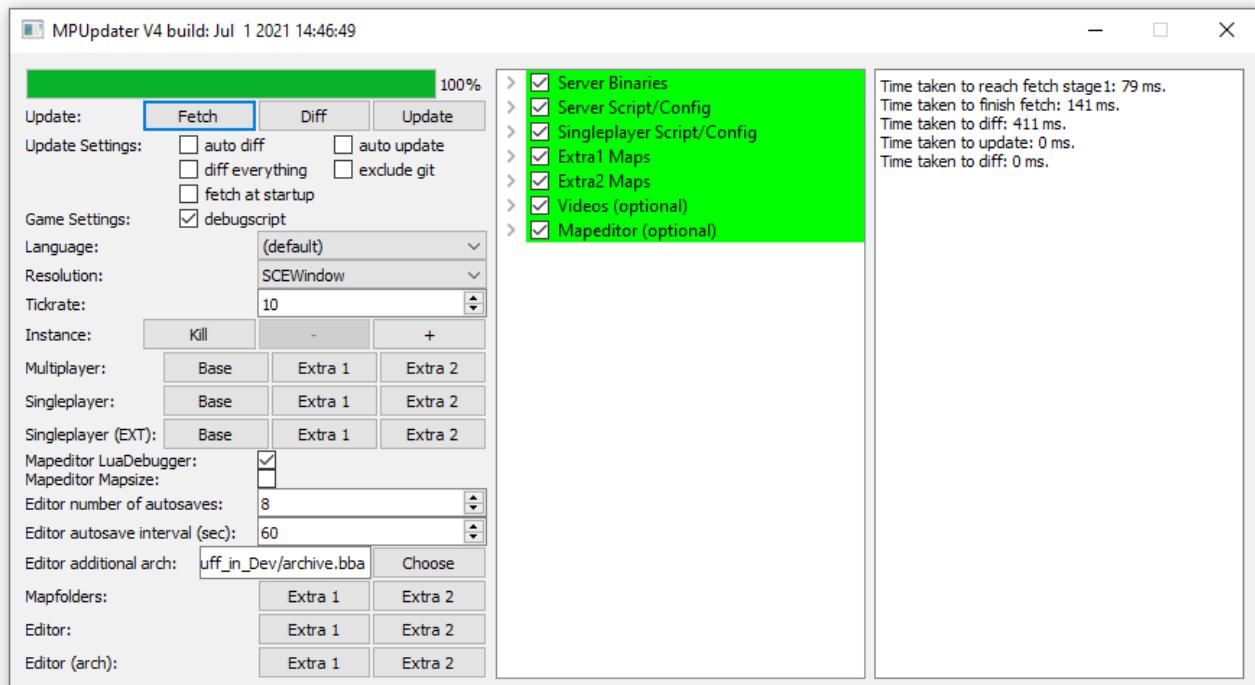


Dann auf „Diff“. Damit wird überprüft ob von allen Dateien die Aktuellste vorhanden ist.



Rot signalisiert, dass die aktuellste Version nicht vorhanden ist. Mit „Update“ werden dann die neuesten Dateien heruntergeladen. Je nach Internet- und Computerleistung kann dies einige Minuten dauern.

Nachdem alle Dateien heruntergeladen wurden, sollte alles in der Mitte grün sein.



Randfall: Falls eine gewisse Datei „S5ZLib.dll“ noch rot angezeigt wird, navigiert in den bin-Ordner im Hauptverzeichnis und löscht die „S5ZLib.dll“. Dann sollen die Schritte Fetch, Diff, Update erneut ausgeführt werden. Anschließend sollte auch dieser Eintrag grün sein.

Damit sollte alles fertig installiert sein und der Karteneditor kann über die Zeile „Editor“ ausgeführt werden und alle zusätzlichen Funktionen sollten vorhanden sein.

## Spielversion Gold Edition (von ~ 2005)

In dieser Anleitung wird das Vorgehen zur Installation von Kimichuras Multiplayer Client für „Die Siedler Das Erbe der Könige“ schrittweise erläutert. Es wird davon ausgegangen, dass das Spiel nicht manipuliert und mithilfe der Gold Edition – CD oder allen 3 einzelnen Spielversionen das Spiel komplett installiert ist.

### Vorbereitung: Herunterladen aller wichtigen Dateien

Zuerst müssen die wichtigen Dateien dafür heruntergeladen werden. Diese sind

- [S5Updater](#) von mcb
- MPUpdater von Kimichuras [Discord](#) im jeweiligen Channel (ganz oben)

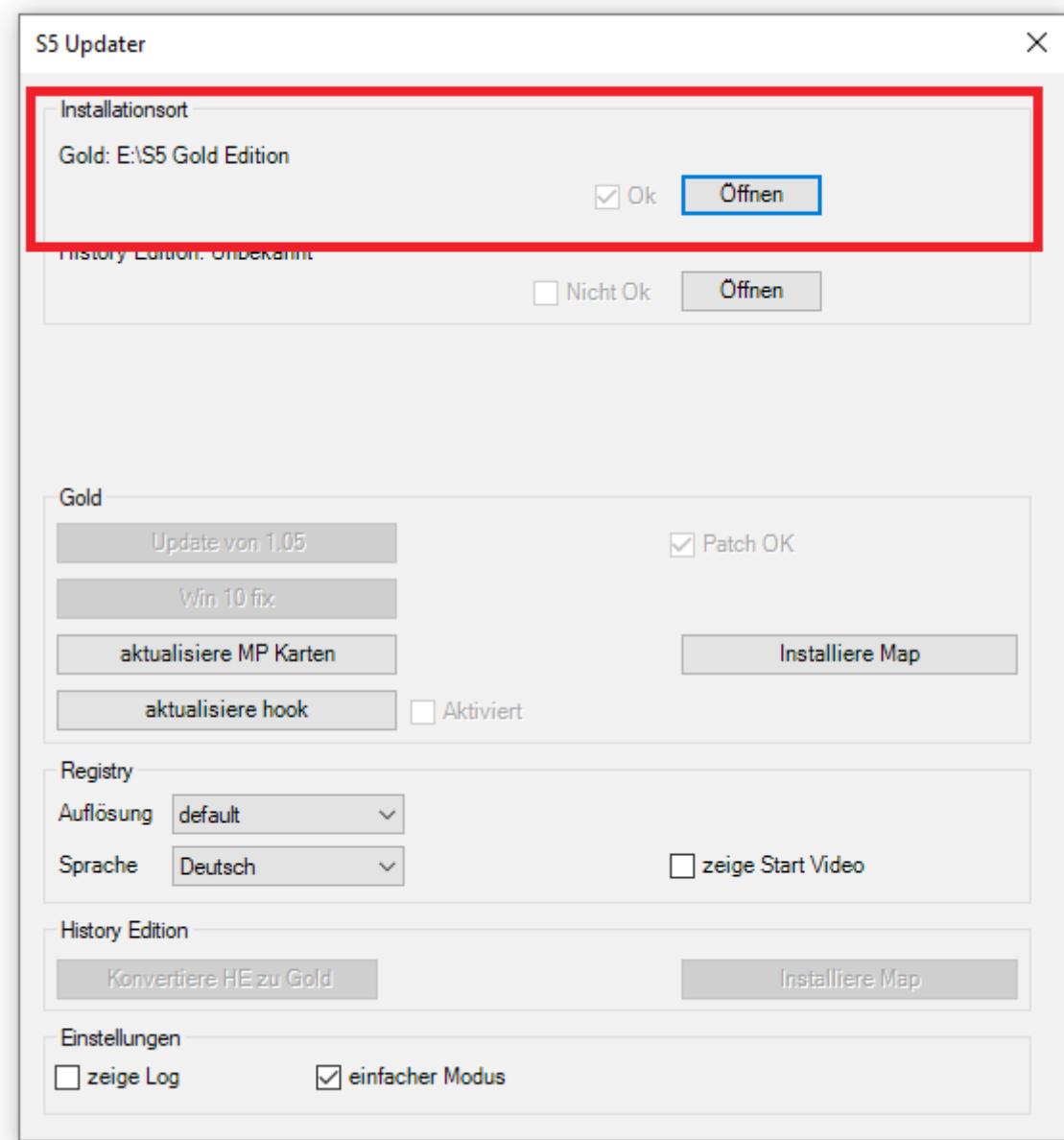
Name	Änderungsdatum	Typ	Größe
de	22.08.2021 14:19	Dateiordner	
AutoDownloader	03.07.2021 22:22	Anwendung	11 KB
bbaTools5	08.02.2021 20:42	Anwendung	468 KB
git2-106a5f2.dll	11.12.2019 08:38	Anwendungserwe...	919 KB
LibGit2Sharp.dll	11.12.2019 08:50	Anwendungserwe...	411 KB
S5Updater	03.07.2021 22:22	Anwendung	59 KB

Der S5Updater wird in einen leeren Ordner der Wahl entpackt, welcher zuvor erstellt wurde.

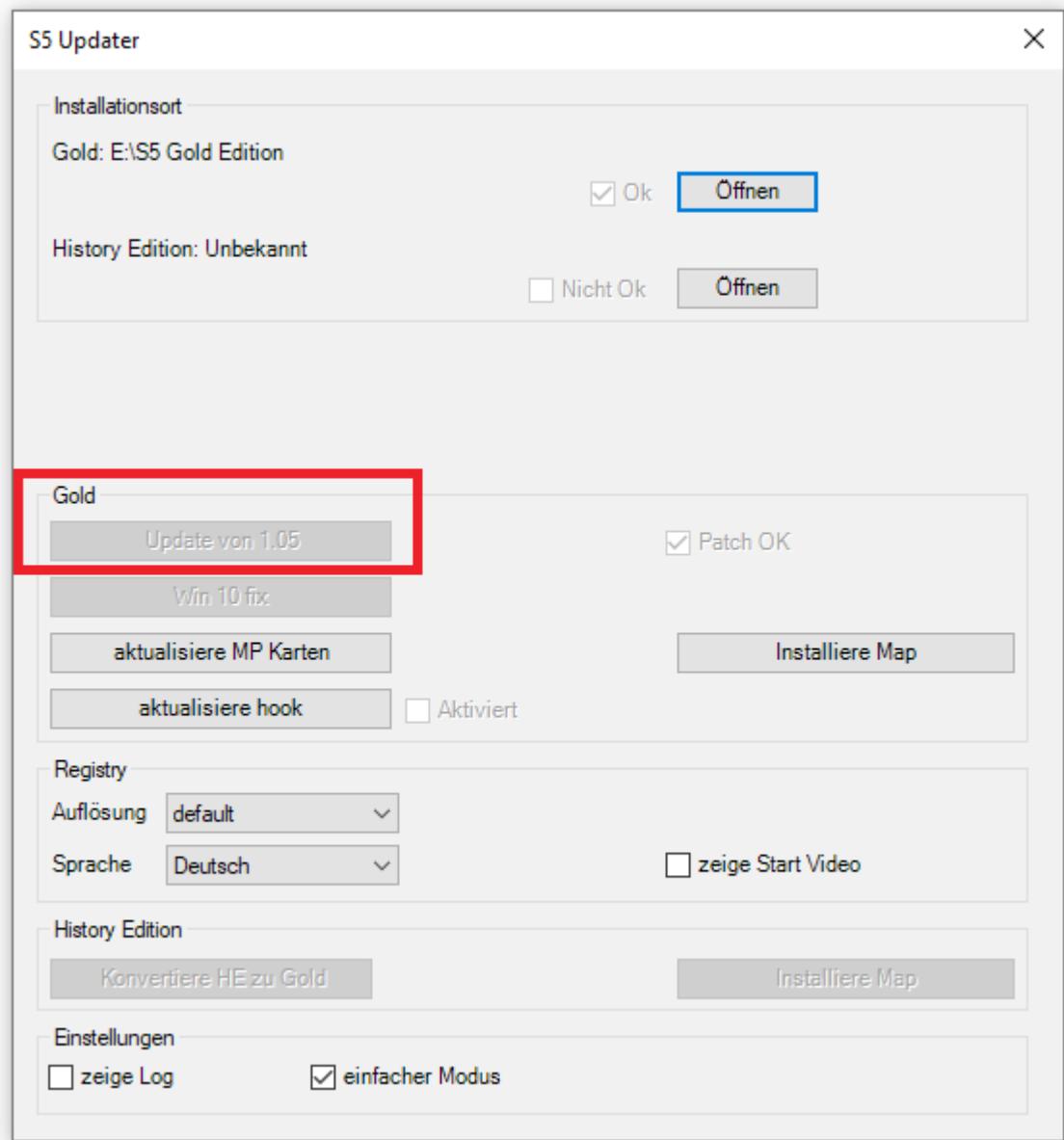
Der MPUpdater wird später benötigt. Diesen wird am besten irgendwo griffbereit abgelegt, bis er gebraucht wird.

## Schritt 1: Auf die Version 1.06 Updaten

Zuerst wird der S5Updater geöffnet und der Pfad der Gold Edition gesetzt.



Anschließend wird auf den Knopf „Update von 1.05“ gedrückt. Dadurch wird das Spiel von der montanen Version auf die Version 1.06 aktualisiert. Sollte bereits die Version 1.06 installiert sein, ist dieser Knopf ausgegraut.

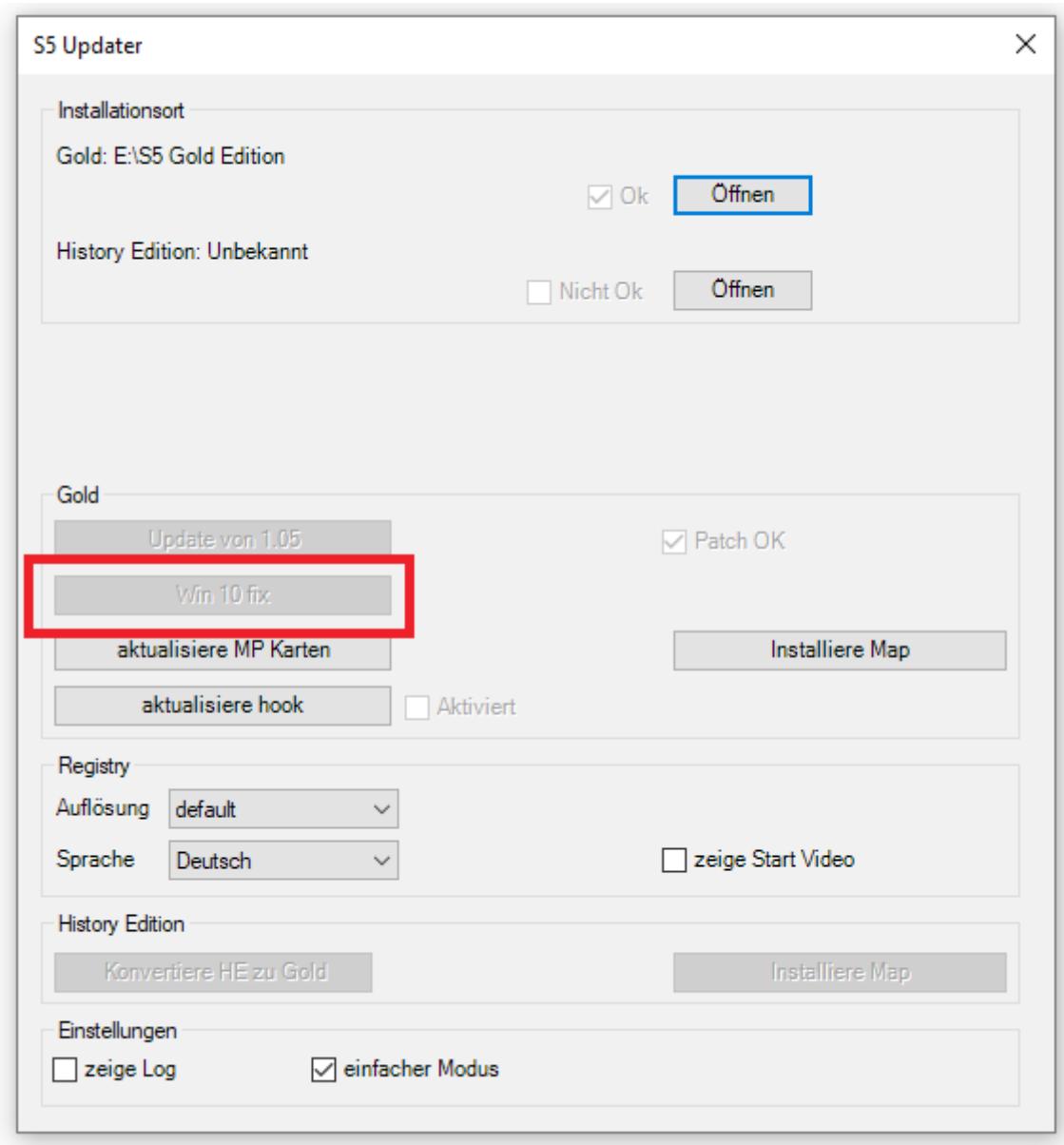


Falls dies nicht funktionieren sollte, muss dies manuell vollzogen werden. Dieser Fall wird im Folgenden behandelt:

Der Patch für die 1.06 muss [hier](#) (direkter Downloadlink) von der Siedler-Maps-Source heruntergeladen und per Doppelklick installiert werden.

Nachdem dieser installiert wurde, muss nun der Windows 10 Fix installiert werden. Dies hängt damit zusammen, dass bestimmte DLL-Dateien, die es in früheren Betriebssystemen gab, mit Windows 10+ nicht mehr vorhanden sind und somit fehlen. Deshalb wird ohne diesen Fix das Spiel in der Version 1.06 auch nicht in der Lage sein zu starten.

Um diesen zu installieren, muss auf den Knopf „Win 10 fix“ im S5Updater gedrückt. Danach wird mit Schritt 2 weitergemacht.



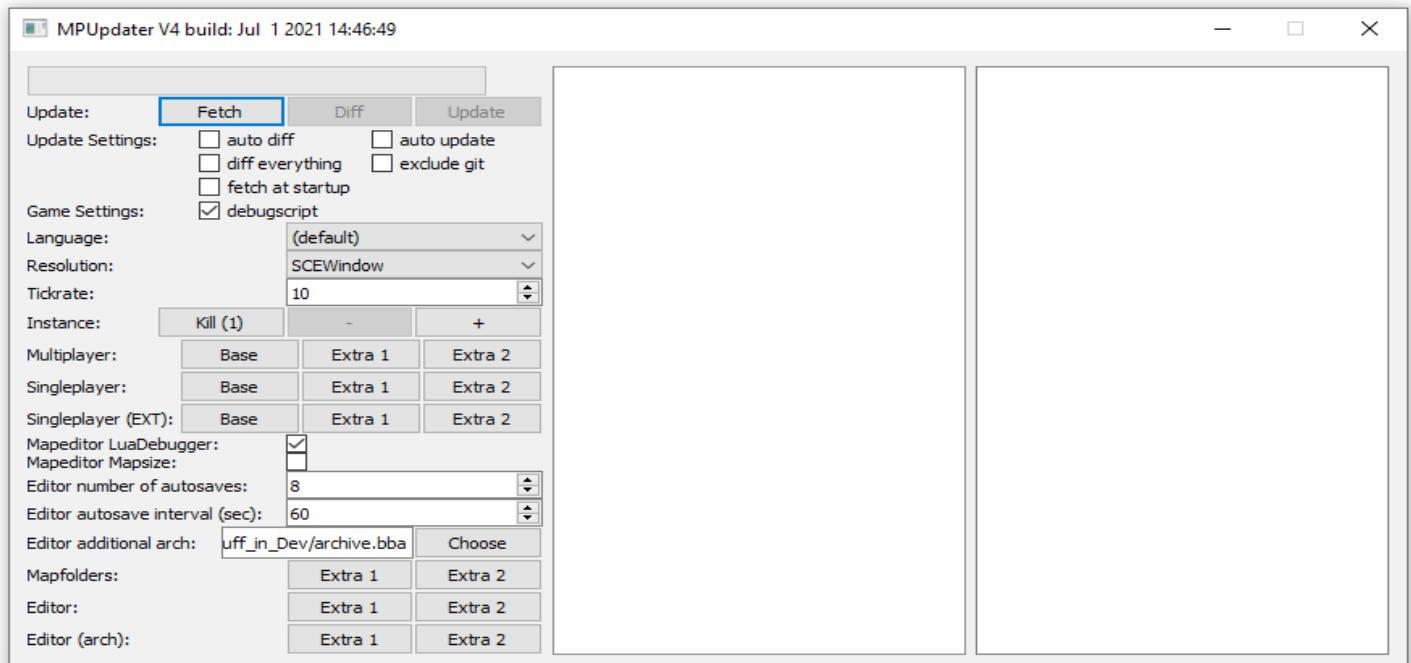
## Schritt 2: Installieren des MPUpdator

Nun wird Kimichuras Client installiert, der zuvor von seinem Discord heruntergeladen wurde. Hierzu werden alle Dateien in das Hauptverzeichnis der Gold Edition gepackt.

Wichtig: Das Hauptverzeichnis soll in dem jeweiligen Anti-Vierenprogramm als Ausnahme eingestellt werden. Es ist gut möglich, dass die MPUpdator.exe als Virus erkannt wird. Hierbei handelt es sich um einen False-Positiv; eine Falschmeldung. Durch das Erstellen einer Ausnahme unterbindet ihr dieses Verhalten.

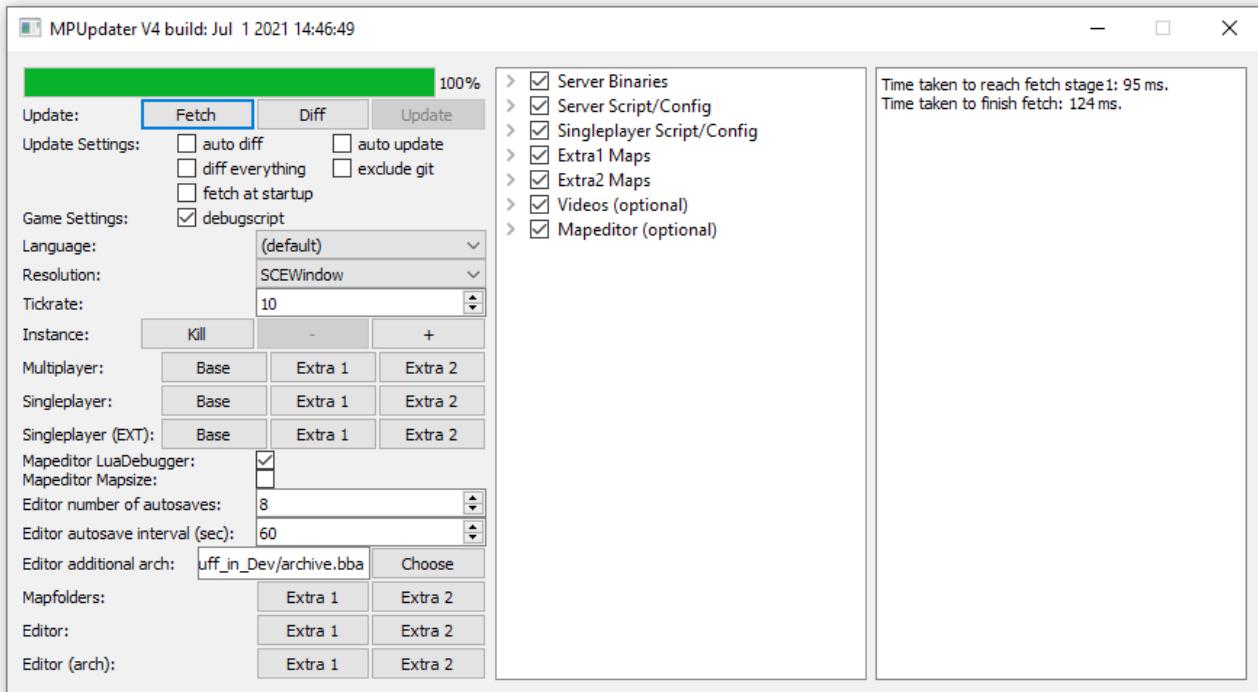
Name	Änderungsdatum	Typ	Größe
base	22.08.2021 14:32	Dateiordner	
bin	22.08.2021 14:34	Dateiordner	
extra1	22.08.2021 14:37	Dateiordner	
extra2	22.08.2021 14:37	Dateiordner	
platforms	22.08.2021 14:40	Dateiordner	
styles	22.08.2021 14:40	Dateiordner	
support	22.08.2021 14:37	Dateiordner	
install.vdk	29.12.2019 17:49	VDK-Datei	1 KB
libeay32.dll	27.12.2019 13:57	Anwendungserwe...	1.241 KB
MPUpdater	01.07.2021 14:46	Anwendung	299 KB
Qt5Core.dll	27.12.2019 13:57	Anwendungserwe...	4.778 KB
Qt5Gui.dll	27.12.2019 13:57	Anwendungserwe...	5.077 KB
Qt5Network.dll	27.12.2019 13:57	Anwendungserwe...	962 KB
Qt5Widgets.dll	27.12.2019 13:57	Anwendungserwe...	4.326 KB
rastertable	27.02.2021 14:41	Datei	1 KB
settlershok.bin	29.12.2019 17:48	BIN-Datei	1 KB
ssleay32.dll	27.12.2019 13:57	Anwendungserwe...	271 KB

Anschließend wird die MPUpdater.exe als Administrator ausgeführt und folgendes Fenster (für die Version 4 des Updaters) sollte erscheinen.

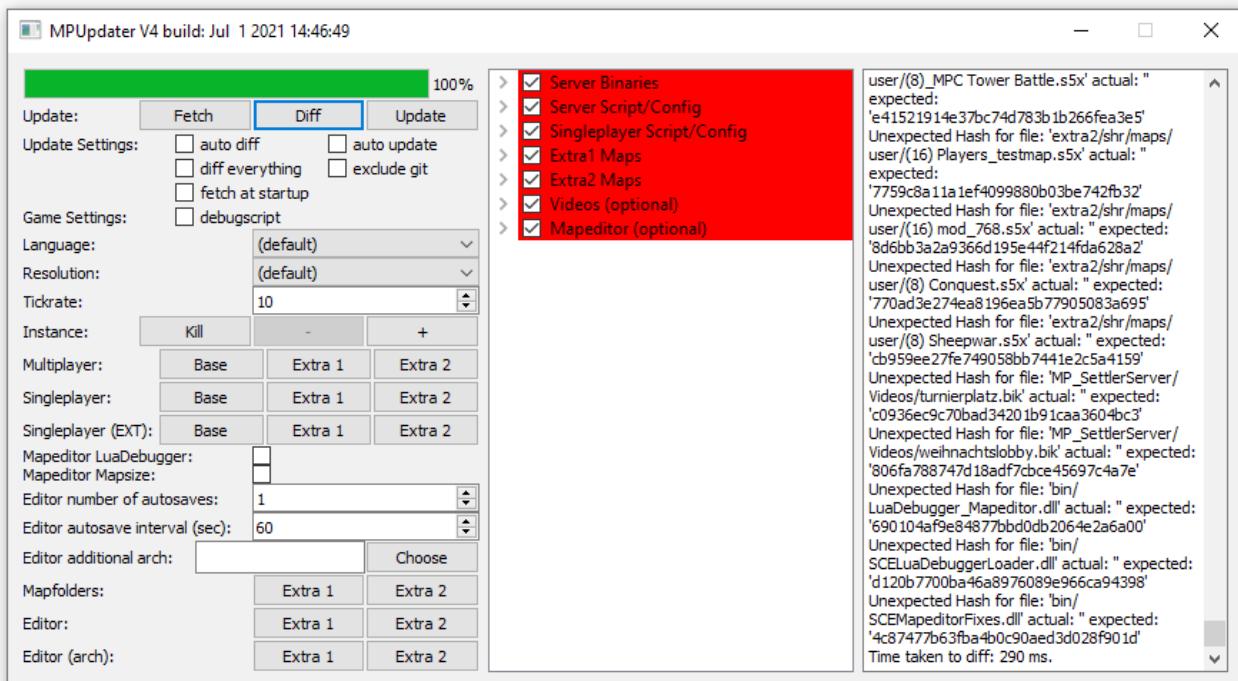


Für den Karteneditor werden die Reihen „Update“ und „Editor“ benötigt.

Zuerst „Fetch“. Damit wird geschaut was für Dateien es momentan gibt.

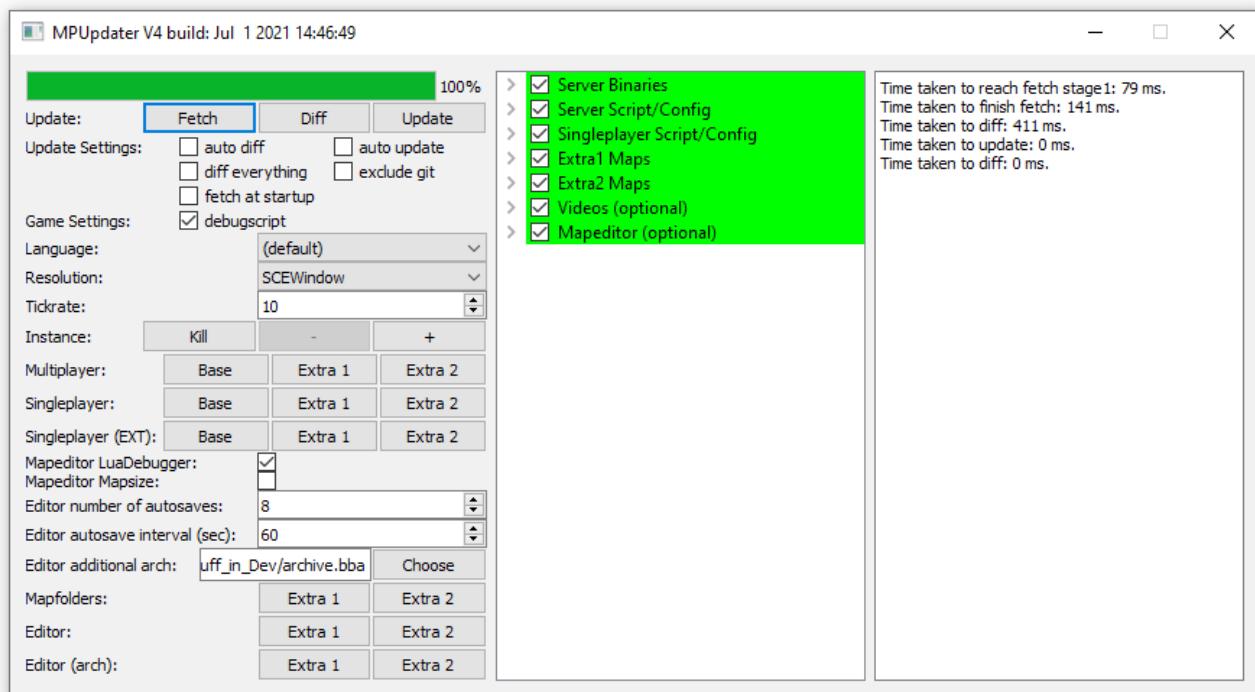


Dann auf „Diff“. Damit wird überprüft ob von allen Dateien die Aktuellste vorhanden ist.



Rot signalisiert, dass die aktuellste Version nicht vorhanden ist. Mit „Update“ werden dann die neuesten Dateien heruntergeladen. Je nach Internet- und Computerleistung kann dies einige Minuten dauern.

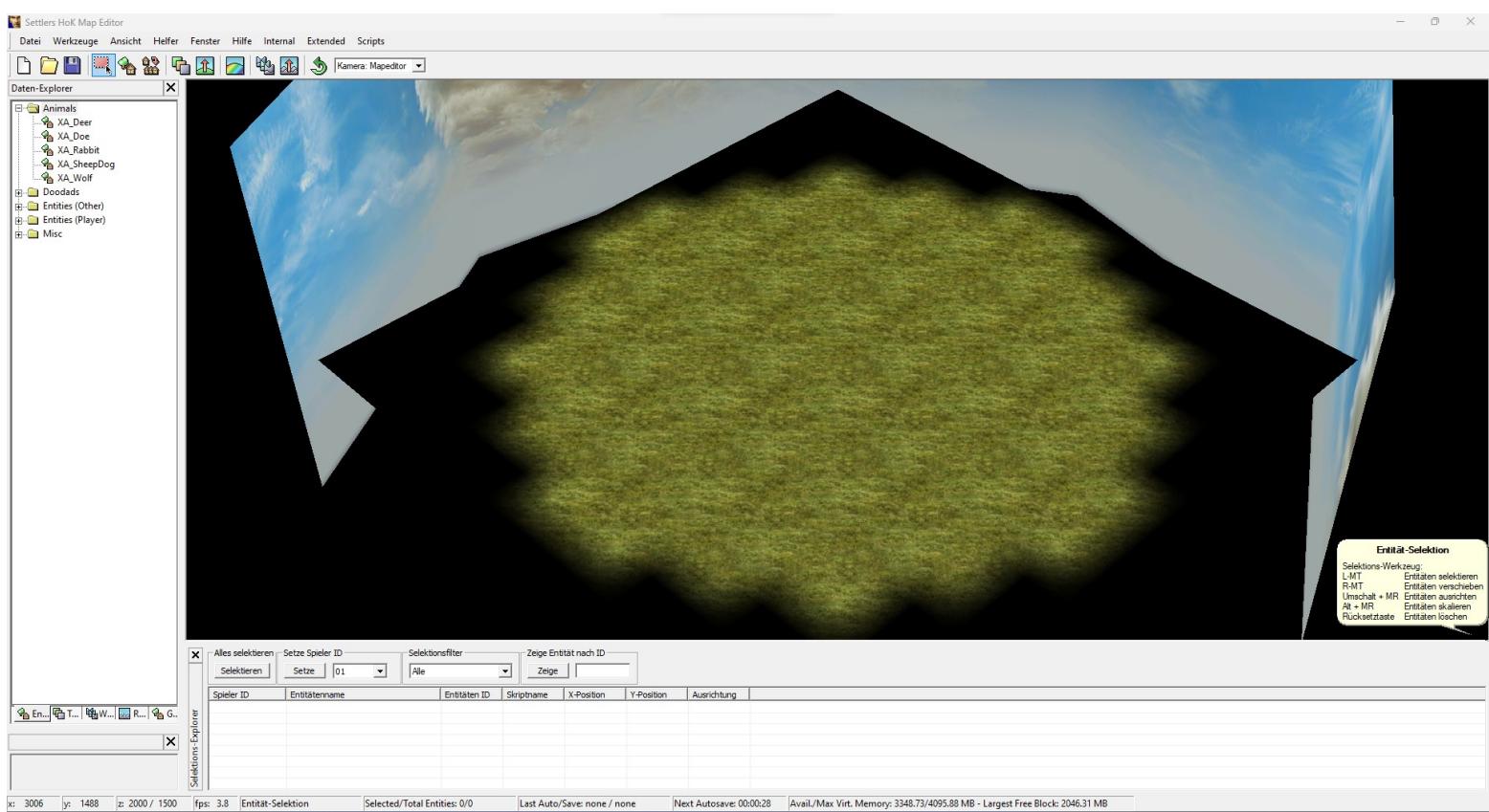
Nachdem alle Dateien heruntergeladen wurden, sollte alles in der Mitte grün sein.



Randfall: Falls eine gewisse Datei „S5ZLib.dll“ noch rot angezeigt wird, navigiert in den bin-Ordner im Hauptverzeichnis und löscht die „S5ZLib.dll“. Dann sollen die Schritte Fetch, Diff, Update erneut ausgeführt werden. Anschließend sollte auch dieser Eintrag grün sein.

Damit sollte alles fertig installiert sein und der Karteneditor kann über die Zeile „Editor“ ausgeführt werden und alle zusätzlichen Funktionen sollten vorhanden sein.

## Der Karteneditor:



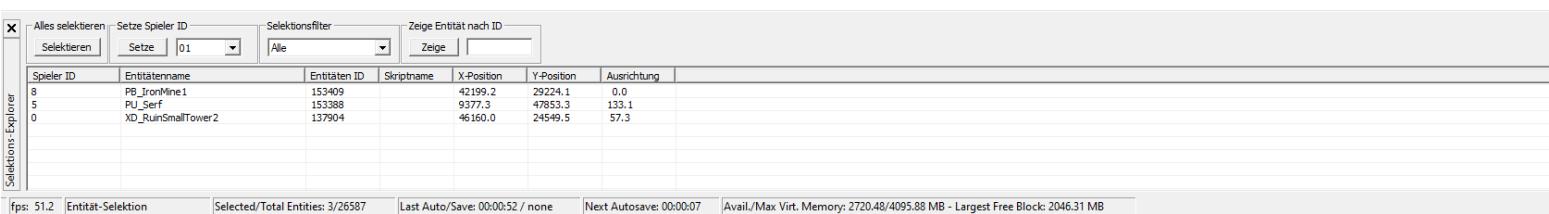
## Grundwerkzeuge:



- Eine neue, leere Karte erstellen mit einigen vordefinierten Eigenschaften, wie Größe und Name.
- Das Öffnen einer Karte, welche als s5x bzw. Karten-Archiv gespeichert ist.
- Das Speichern einer Karte in das s5x bzw. Karten-Archiv.
- Über das Selektionsmenü. können Entities auf der Karte selektiert, bewegt, gedreht und skaliert werden.
- Entities können aus einer internen Liste ausgewählt, vor dem Setzen gedreht und positioniert werden.
- Eine Kopiervorlage erstellen bzw. eine bereits vorhandene Kopiervorlagen mit unterschiedlichen Parametern (spiegeln, Textur, Objekte und Terrain überschreiben) nutzen.
- Die Terraintextur kann verändert werden. Hierbei kann blockierender und nicht blockierender Boden gesetzt werden.

- Die Terrainhöhe kann zwischen 0 und *relativ* unendlich verändert werden.
- Auf das Terrain kann Vertexfarbe im RGB-Spektrum und in unterschiedlicher Intensität platziert werden. Ebenso kann es aufgehellt und verdunkelt werden.
- Der Wassertyp kann aus einer Liste von Typen ausgewählt und gesetzt werden.
- Die Wasserhöhe kann zwischen 0 und *relativ* unendlich variiert werden.
- Sollte man sich vertan haben, können über den Knopf, oder Strg + Z, Aktionen rückgängig gemacht werden.
- Die Kamera kann zwischen 3 Modi umgeschaltet werden: Einen für den Mapeditor, bei dem man mehr sieht. Einen für die Kamera, wie sie im Spiel später angewandt wird und einen von oben für eine Draufsicht auf die Karte.

## Selektions-Explorer:



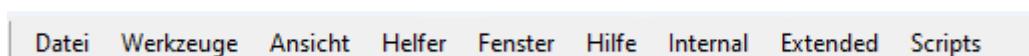
The screenshot shows the Selections-Explorer window with a toolbar at the top containing buttons for 'Alles selektieren' (Select All), 'Setze Spieler ID' (Set Player ID) with a dropdown set to '01', 'Selektionsfilter' (Selection Filter) with a dropdown set to 'Alle' (All), and 'Zeige Entityät nach ID' (Show Entity by ID) with a text input field. Below the toolbar is a table with the following data:

Spieler ID	Entitätenname	Entitäten ID	Skriptname	X-Position	Y-Position	Ausrichtung
8	PB_IronMine1	153409		42199.2	29224.1	0.0
5	PU_Serf	153388		9377.3	47853.3	133.1
0	XD_RuinSmallTower2	137904		46160.0	24549.5	57.3

At the bottom of the window, there is a status bar with the following information: 'fps: 51.2 Entität-Selektion Selected/Total Entities: 3/26587 Last Auto Save: 00:00:52 / none Next Autosave: 00:00:07 Avail./Max Virt. Memory: 2720.48/4095.88 MB - Largest Free Block: 2046.31 MB'

Der Selektions-Explorer zeigt alle selektierten Entities auf der Karte an. Es können Informationen, wie Spieler, Entitytyp, die EntityID, die genaue Position und die Ausrichtung abgelesen werden. Des Weiteren kann hier auch die visuelle Größe einer Entity geändert werden über einen Rechtsklick auf das Entity in der Liste und einer anschließenden Auswahl von „Edit as XS\_Ambient“. Der Radius gibt dabei die Größe des Models an und kann beliebig verändert werden.

## Menü-Leiste:



In der Menüleiste befinden sich nützliche Werkzeuge zu einer allgemeinen verbesserten Handhabung der Karte.

Unter der Rubrik **Datei** können Karten ebenfalls als Archive abgespeichert und geladen, sowie neue leere Karten erstellt werden. Zusätzlich kann die Karteninformation bearbeitet werden, welche in der Kartenauswahl später im Spiel sichtbar ist. Das Verwenden des Eintrags **Skript bearbeiten** ist abzuraten. Im sich anschließend öffnende Fenster kann

zwar das komplette Skript später geschrieben werden, doch wird nur ein bestimmte Zeichensatz davon gespeichert. Dies führt später zu erheblichen Lua-Fehlern.

Unter **Werkzeuge** befinden sich die bereits erklären Tools zur Bearbeitung der Karte, welche sich in der Leiste der Grundwerkzeuge befinden. Zusätzlich lassen sich zwei Kategorien öffnen mit denen offene und geschlossene Rohstoffschächte platziert und Textur-/Entität-Gruppen verwendet werden können, um Texturen und Entitäten in Relation zu einander willkürlich verteilen zu lassen.

Mit **Ansicht** lassen sich bestimmte Aspekte an- und ausschalten. Wichtig sind hierbei die beiden Aspekte „Skriptentitäten“ und „Schnee“. Auf den Karten werden sehr viele Skriptentitäten sichtbar sein, welches es teilweise schwer macht, bestimmte Entitäten auszuwählen. Mithilfe der Schneeeansicht ist es möglich zu sehen, welche Gewässer im Winter zufrieren und so weitere Wege erschlossen werden können.

Mithilfe des Reiters **Helper** können unterschiedliche Aktionen getätigigt werden, wie z.B. das Spiegeln einer Kartenhälfte auf die Andere basierend auf der X- und Y-Achse, auch Höhen-, sowie Texturen-Karten geladen werden (wird mittlerweile nicht mehr benutzt). Der Ressourcenmanager zeigt Informationen über die Karte an und man kann dabei überprüfen, wie viele Entities bereits auf der Karte vorhanden sind, zumal man weitere Entities im Laufe des Spiels mit einberechnen sollte, um das Limit nicht zu überschreiten, welches sich ungefähr bei ~ 65000 Entitäten befindet.

Im Untermenü von **Fenster** kann man die verschiedenen Teilfenster an- und ausschalten. Der Menüpunkt „Raster“ ist wichtig und seine Tastenkombination Strg + G sollte im Hinterkopf behalten werden. Gerade beim Erstellen von Terrain und beim Platzieren von Entitäten ist es später wichtig zu sehen an welcher Stelle welcher Pfad durch was genau blockiert wird um keine Unregelmäßigkeiten beim Manövrieren von Einheiten zu erhalten.

Der Punkt „Über“ in **Hilfe** ist lediglich zum Nachschauen der Version gedacht.

Der **Internal-Tab** ist bereits im Karteneditor der History Edition vorhanden und war in der ursprünglichen Version versteckt, sodass man ihn nicht benutzen konnte. Trotzdem sind hier einige Unterpunkte vorhanden, die sehr hilfreich sind. Karten können als s5x oder als Ordner abgespeichert werden. Um sie als Ordner abzuspeichern und zu laden, werden jeweils „Save Folder“ und „Open Folder“ benutzt und dann die jeweiligen Ordner ausgewählt. Im Verlauf des Projekts werden diese Funktionen benutzt um eine Karte zu erstellen.

Unter **Extended** finden wir nun die neuen Funktionen, die durch den modifizierten Karteneditor bereit gestellt werden. Hierbei fallen, neben dem Exportieren von bestimmten Sachen, auch das mögliche Deaktivieren von Platzierungsüberprüfungen, sowie weitere Komfortabilitäten auf:

- Aufhebung der Bilder pro Sekunde Grenze
- Exportmöglichkeiten von der Karte im Ganzen oder in Teilen komplett als Lua-Skript in die Zwischenablage

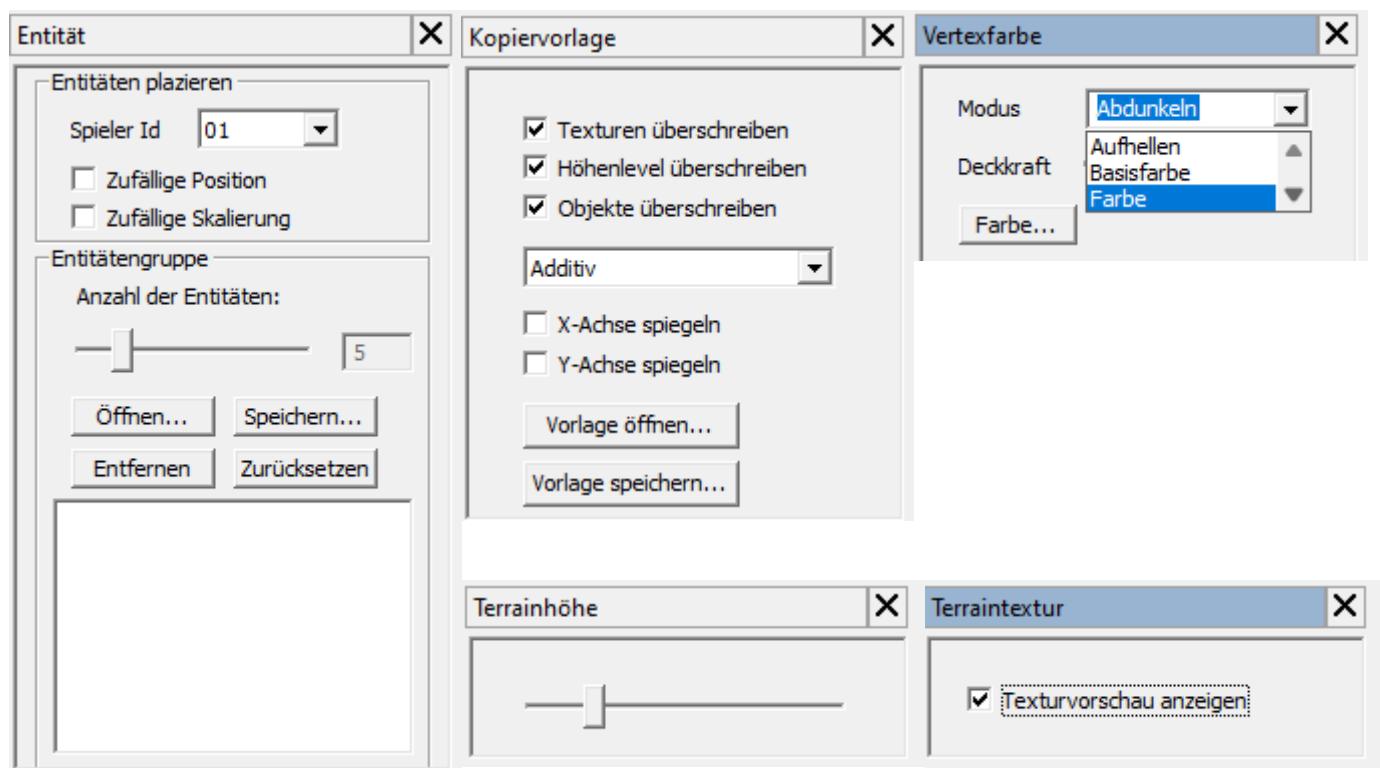
- Rundung der Richtungsorientierung von gedrehten Entitäten
- Deaktivierung von Platzierungschecks
- Verbesserung der Rotation beim Spiegeln der Karten
- Setzen von Skalierungsfaktoren für Models von Entitäten
- An- und Ausschalten einer farbigen Kennzeichnung der Spiegelachsen.

Diese Funktionen erleichtern das Erstellen ungemein, vor allem die Platzierungschecks, und wirken sich somit effizient und zeitsparend auf den Vorgang aus.

Mit dem letzten Menüpunkt **Scripts** können Lua-Skripte im Editor ausgeführt werden, sobald sie im Order <Siedler-Hauptverzeichnis>/MapEditor/Scripts/ abgelegt werden.

## Variablen Werkzeug-Menü:

In der linken unteren Ecke können basierend auf dem momentan ausgewählten Werkzeug verschiedene Einstellungen aktiviert werden. Darunter fallen die Stärke an Terrainveränderungen, Intensität der Wasseränderung, Vertexfarbeinstellungen und viele weitere Möglichkeiten.



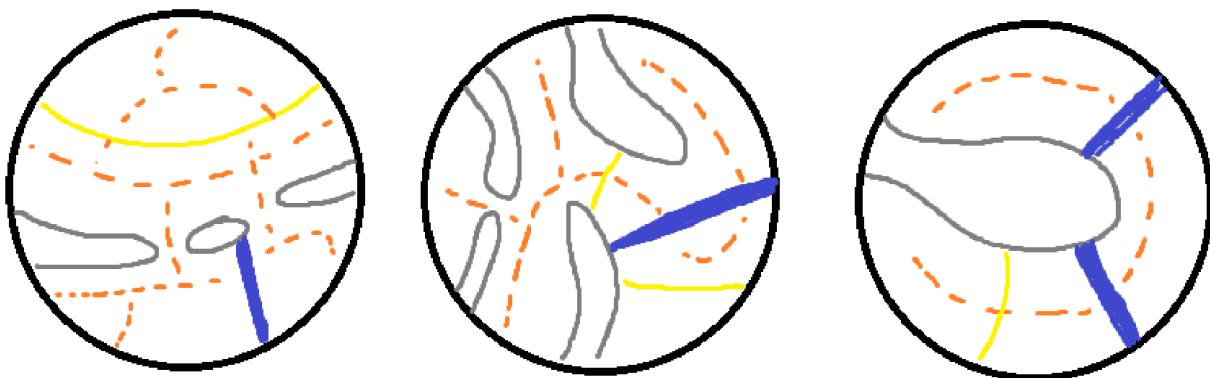
## Das Erstellen der Karte:

### Ideen und Konzepte:

Bei dem Spiel handelt es sich um ein Echtzeit-Strategiespiel, was eigentlich einen großen Fokus auf Wirtschaft und Militär legt. Allerdings gibt es diesem Spiel viele unterschiedliche Möglichkeiten der Freiheit. Die einfachste Freiheit hierbei ist eine Kampfkarte gegen einen Gegner; ein Startplatz, bestimmte Rohstoffe und einen Gegner. Eine andere Freiheit sind die Questkarten. Diese Karten stechen durch ihre Dynamik und ihre Geschichte hervor. Sie beinhalten eine größere und tiefgreifendere Geschichte mit unterschiedlichen, mal größeren, mal kleinen Aufgaben; nicht unbedingt basierend auf Kampf, sondern auf Interaktionen.

In diesem Fall soll eine kleine Questkarte gebaut werden, welche eine kleine Geschichte erzählt. Worüber die Geschichte im Detail sein wird, ist bis dato noch unbekannt, aber es können bereits erste Skizzen gezeichnet werden, die eine grobe Orientierung und bereits erste Bearbeitung der Karte realisieren.

Die gelben Linien symbolisieren Stadt- bzw. Landgrenzen um grobe Ländereien zu kennzeichnen. Das Städtedesign wird im Laufe des Baus spontan entstehen. Alles innerhalb der grauen Linien sind Berge und damit nicht begehbares Terrain. Blau steht für Gewässer. Ob diese im Winter gefrieren können oder nicht, bleibt derweil unbehandelt; ebenso woher die Gewässer kommen. Orangebraune gestrichelte Linien gestalten sich als ungefähre Wege die entstanden sein können.



## **Erstellen der leeren Karte:**

Um eine neue, leere Karte zu erstellen, wird mithilfe des Eintrags „Neu“ (ein leeres Blatt Papier) der Karten- und Skriptassistent geöffnet. Dieser leitet einen wie ein Installationsprogramm von einem Programm früher durch. Die Kartengröße muss dabei bereits im vorhin gesetzt werden. Der Kartename sowie die Beschreibung werden im Nachhinein angepasst.

Da es sich bei dem Level um eine Einzelspieler-Karte handel soll, befindet sich im Feld „Mehrspielerkarte“ kein Haken. Die restlichen vier Optionen können so bleiben. Sie wirken sich nicht auf die Generierung aus.

Ebenso ist es egal, welche Werte bei den Rohstoffen zu Partiebeginn vorliegen. Die Rohstoffe werden später per Lua-Skript übergeben.

Die Aufgabe-Felder werden auch leer gelassen. Die Aufgaben werden später im Spiel gestellt und behandelt, sodass eine Geschichte entsteht.

Die Siegbedingung(en) sind irrelevant. Es werden später eigene Siegbedingungen geschrieben.

Die KIs können übersprungen werden. Die Truppen, sowie deren Verhalten werden manuell geschrieben bzw. das Siedler-interne System genutzt.

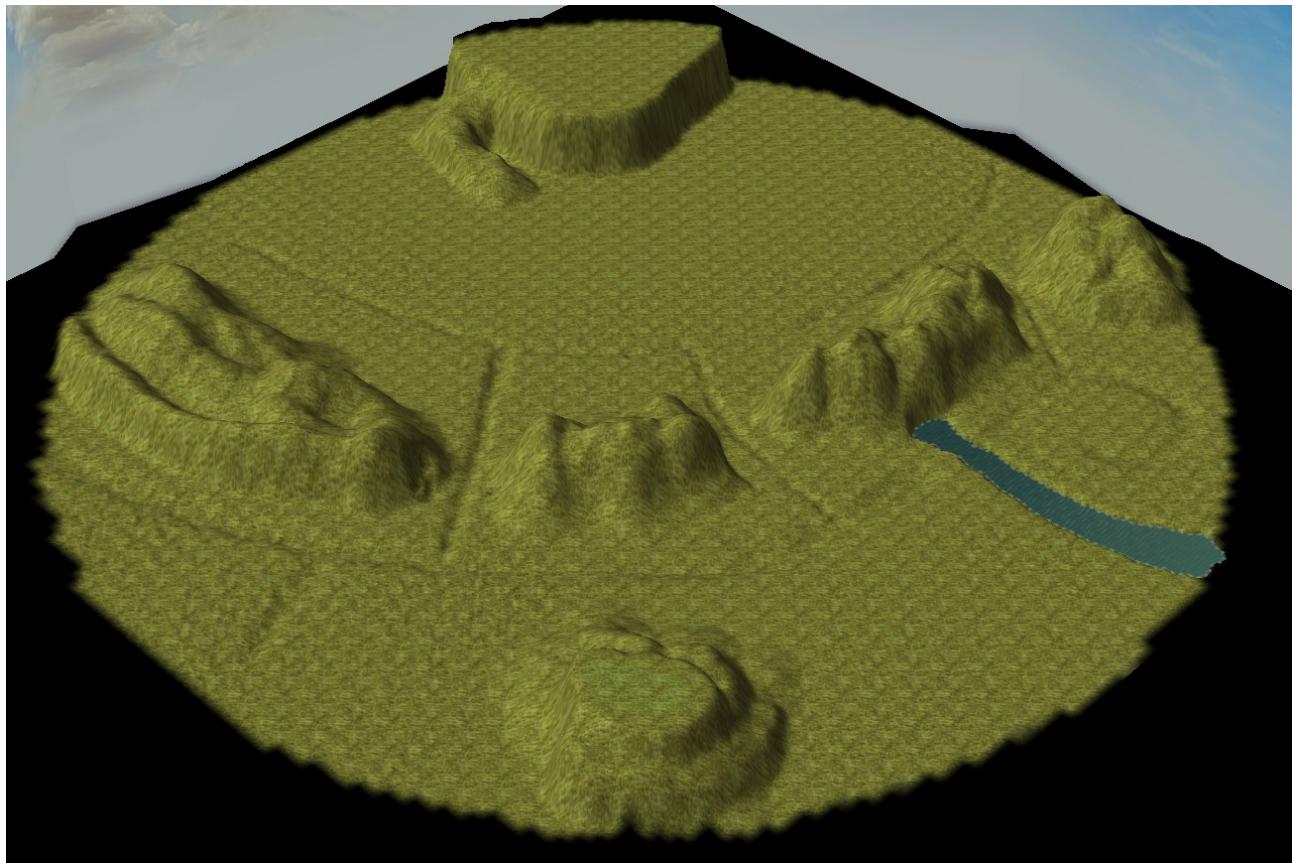
Die Skriptvorschau soll ignoriert werden. Das Skript wird später selbst geschrieben. Hier wird nicht mit vorgefertigten Einträgen gearbeitet.

Zum Schluss wird bestätigt, dass die momentane Karte gelöscht wird um die neue, leere zu erstellen.

Für die Karte wird eine Größe von 512 Einheiten benutzt. Dann ist die Karte nicht zu groß und nicht zu klein. Bei zu großen Karten gestaltet es sich schwer genug Inhalt aufzubringen, während es bei zu kleinen Karten der Platz zu knapp wird.

## Grundstrukturen:

Mit der Auswahl der ersten Idee beginnt auch gleich die Grundstrukturierung des Terrains.



Dabei fallen drei Begebenheiten auf:

1. Im Städtebereich befindet sich eine Erhöhung. Sie ist entstanden um der Stadt eine gewisse Dynamik zu geben und nicht nur auf einer Ebene entstehen zu lassen.
2. Im Süden ist eine Erhöhung entstanden, die der Fantasie freien Lauf geben soll. Je nach Entwicklung kann dieser Berg bleiben oder entfernt werden.
3. In der rechten Gebirgskette ist ein weiterer Zugang nach Norden geformt worden. Anstatt den Spieler immer in die Mitte laufen zu lassen, macht ein weiterer Zugang es dem Spieler nur angenehmer.

Diese Strukturen dienen dazu der Karte ihre erste grobe Form zu geben. Anhand dieser Umgebung können weitere Entscheidungen getroffen werden entsprechende Texturen für welche Bereiche zu verwenden. Ebenso können weitere Ideen der Karte zugerechnet und auf ihre Tauglichkeit überprüft werden.

Folglich wurde festgelegt, dass es sich südlich der Berge und westlich des Flusses um ein Waldstück handeln soll. Das Waldstück soll relativ licht sein und beinhaltet einzelne blockierende Steine.

Vereinzelt sollen sich Leute im aufhalten und die Erhöhung soll eine Anhöhe sein, auf der Spieler eine oder mehrere Aufgabe zu bewältigen hat. Der Schnee ist vorerst provisorisch dort. An die Anhöhe sollen vermehrt Steine gesetzt werden, um den Eindruck von losen herabfallendem Geröll zu erwecken.

Weiterhin soll die Zugänge von Süden nach Norden durch Mauern mit Toren versperrt sein; zumindest zum Spielstart. Im Laufe des Spiels sollen diese sich dann öffnen.

Im nördlichen Gebiet, außerhalb der Stadtmauern, wird eine Baustelle für ein Dorfzentrum gesetzt um eine spätere kleine Siedlung zu markieren.

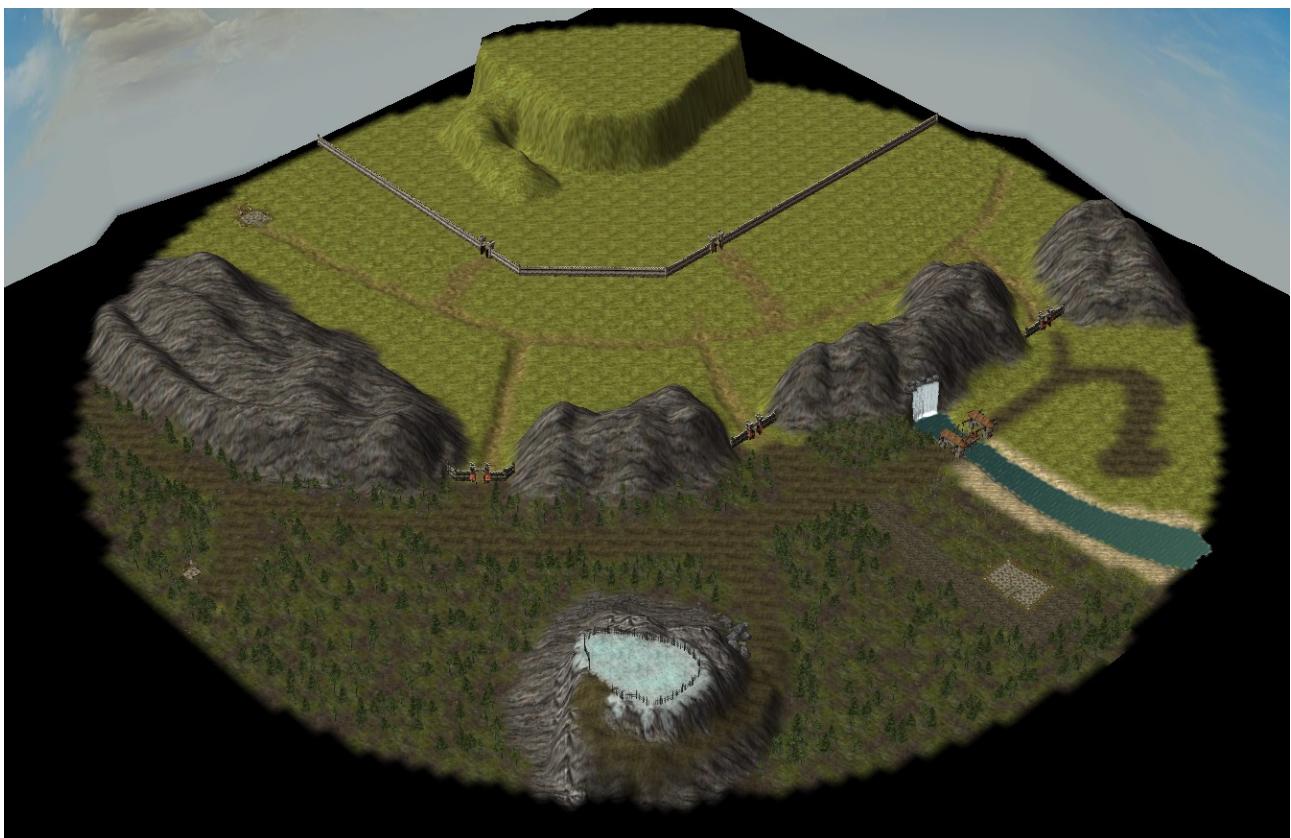
Mauern im nördlichen Gebiet werden die Stadtgrenzen symbolisieren. Diese können im Laufe der Erstellung dynamische Formen als nur gerade annehmen.

Damit der Fluss nicht aus dem Nirgendwo entspringt, werden Wasserfälle aus dem Berg den Fluss mit Wasser optisch speisen. Die Wasserfälle selbst entstehen durch Wasserquellen im Berg oder aus Felsen heraus.

Über den Fluss soll eine Brücke führen. Damit diese nicht zu lang und nicht zu städtische aussieht, wird eine Zugbrücke aus Holz statt einer Steinbrücke genutzt. Zusätzlich soll das Ufer des Flusses aus Sand bestehen.

Um diese Ziele zu erreichen helfen uns unterschiedliche Methoden. Zum einen können die Placementchecks beim Platzieren von Entitäten ausgeschaltet werden (über den Extended Tab), weshalb Entitäten einfacher in unwegsamen Gelände platziert werden können. Zum anderen können Entitäten und Texturen in guter Abmischung über das Werkzeug der Gruppen platziert werden und so der erste Teil der Karte aufgebaut werden.

## Der ländliche Teil der Karte:



Mit der Erfüllung der gestellten Anforderungen, nimmt die Karte allmählich Gestalt an. Die Gebirge sind mit einer relativ monotonen Textur versehen worden, welche später verfeinert werden kann, sodass auch Berge natürlich aussehen. Dafür wurde dem Boden ein Profil gegeben, welches leichte Höhen und Tiefen einbaut.

Der südliche Teil hat durch unterschiedliche Bäume und Terrain Texturen ein Waldleben bekommen und auch einzelne Siedler haben sich im Wald mit ihren mehr oder weniger futuristischen Häusern angesiedelt. Der Wald ist nur teilweise dicht.

Die Standarten, die an den Toren angebracht wurden, wurden als einzelne Entitäten mit einer höheren Größenskalierung in die Tore platziert. Dadurch wirken sie wie Tor-Banner, welche aufgehängt wurden.

Der Weg wurde mit einer dunkleren Erdtextur gebaut um ihn vom Waldboden zu differenzieren.

Der Zaun auf der Anhöhe soll eine dekorative Abgrenzung zum steilen Hang sein.

## **Entstehung einer Geschichte:**

Mit dem Voranschreiten des Designs der Karte kommen bereits erste Ideen für die Karte. Man spielt mit Gedanken herum, z.B. welche Personen von woher auf der Karte erscheinen können um einen Anfang zu bieten und wie sich das Ganze dann weiter entwickeln kann.

In diesem Fall hat sich Folgendes ergeben:

Es soll um eine kleine Beziehungsgeschichte gehen. Eine Person läuft durch den Wald. Auf dem Weg durch den Wald trifft er einzelne Personen, die ihm etwas über die Region erzählen, darunter auch über eine Stadt in der Nähe, welcher von einer Prinzessin regiert wird. Auf den Wegen findet er Zugänge Richtung Norden, die allerdings versperrt sind und von einzelnen Wachen bewacht werden. Die Wachen weisen ihn darauf hin, dass er keinen Zutritt bekommt. Deshalb schaut er sich weiter um, auch über einen Fluss hinweg. Am anderen Ufer des Flusses haben sich Banditen angesiedelt. Diese möchte die Person, nennen wir sie einfach mal Herbert (kann sich ändern), genauer unter die Lupe nehmen. Bei einer Auseinandersetzung mit den Banditen flüchtet er wieder auf das andere Flussufer. Nach diesem Aufeinandertreffen sucht er weiter den Wald ab und findet eine Erhebung. Auf dieser Erhebung zieht er die Aufmerksamkeit der Prinzessin auf sich. Sie befiehlt einigen ihrer Leute nach dem Rechten zu sehen. Die Wachen finden Herbert und bringen ihn unter Geleitschutz in die Stadt im Norden zur Prinzessin. Diese misstraut ihm noch für den Moment und er solle sich Beweisen, indem er Menschen in und um ihrer Stadt bei den unterschiedlichsten Aktivitäten hilft und so das Vertrauen der Bürger gewinnt.

Nachdem Herbert den Menschen der Stadt geholfen hat, vertraut die Prinzessin auch ihm. Herbert erzählt der Prinzessin von den Banditen und was sie für ein Problem in der südlichen Region darstellen. Daraufhin entsendet die Prinzessin einige ihrer Soldaten, inklusive Herbert, um die Banditen zu vertreiben.

Als die Banditen in die Flucht geschlagen waren, ziehen Herbert und die Soldaten unter Feuerwerk und Erfolgsmusik durch die Stadt zur Prinzessin. Nicht nur hatte die Prinzessin Gefallen an Herbert gefunden, sondern ihn auch wirklich lieb gewonnen.

Herbert und die Prinzessin sind ein Paar geworden und leben fort an glücklich und zufrieden bis an ihr Lebensende zusammen.

Diese Geschichte bildet den Grundbaustein für alle Handlungen im späteren Verlauf und für den roten Faden.

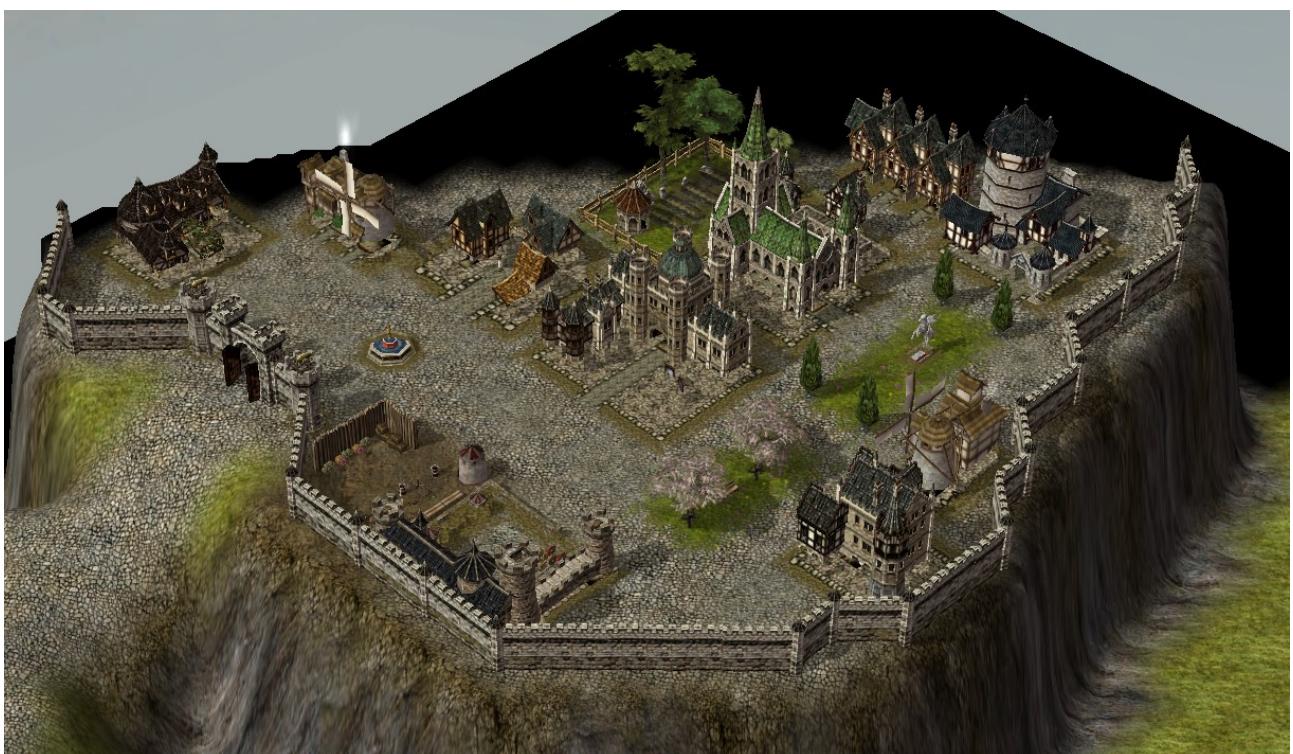
## Der Bau der Stadt:

Die Stadt im Norden besteht im Grunde aus 3 Ringen. Die Ringe sind mit Mauern von einander getrennt. Der äußere Ring beinhaltet überwiegend die ärmeren, arbeitende Bevölkerung. Dazu zählen Bauern, Arbeiter und Handwerker, sowie viel freie Fläche und eine nicht als zu dichte Besiedlung.

Der mittlere Ring beinhaltet die Mittelklasse. Hier sind Arbeiter in höheren Berufen tätig, sowie einzelne themenspezifische Distrikte vorhanden, wie Militär oder Veredelung und Handel.

Im Inneren Ring befinden sich die adeligen Bewohner. Er soll mit viel Zierde geschmückt sein und auch prachtvolle Bauten, wie Kathedralen, Schatzkammern und eine Burg beinhalten. Ebenso sind die Grünflächen mit Statuen dekoriert. Ein Militärgebäude dient als Darstellungspunkt der royalen Garde.

Erstmal wird der innere Ring gebaut:



Der Boden wird mit 2 Texturen, einmal Pflasterstein und einmal Kies, durchmischt um dem Boden eine gewisse Tiefe und Struktur zu verleihen. Wichtig ist hierbei, dass am Ende keine Lagerfeuer entstehen, weshalb es genügend Plätze zum Essen und Schlafen in der Umgebung geben muss. Dies wird durch Gutshöfe, das Wirtshaus und die Wohnhäuser sicher gestellt. Schließlich passen Lagerfeuer auf der Straße nicht in den Adelsbezirk hinein. Die Schatzkammer stellt hierbei den Wohlstand da. Die Universität zeugt von der Bildung der Menschen in diesem Distrikt. Hinter der Kathedrale wurde ein Friedhof gebaut. Normalerweise existiert die Kathedrale im Spiel ohne einen Friedhof, allerdings passt dieser gut und konnte durch viele einzelne Entitäten schön gebaut werden.

Nachdem der innere Ring fertig gestellt wurde, geht die Arbeit am mittleren Ring weiter. Der mittlere Ring sticht vor allem durch seine Mittelschicht heraus. Die Gebäude sind mittelmäßig ausgebaut und beherbergen überwiegend Arbeiter aus dem zweiten Wirtschaftssektor, die die Materialien benutzen um sie weiter zu verarbeiten (bzw. zu veredeln für mehr Rohstoffe des selben Typs).

Der Boden besteht hierbei aus einer ähnlichen Textur wie im inneren Ring um an viel Pflasterstein anzulehnen. Neben vielen Wohnhäusern im mittleren Ring sollen auch einige Distrikte hier ihren Sitz finden. Dabei wird das Gebiet unterteilt in

- einen produktiven Bezirk, bestehend aus Schwefel- und Eisenveredelung
- einen militärischen Bezirk angrenzend an den produktiven Bezirk
- einen Wohnbezirk mit vielen Wohnhäusern unterschiedlicher Strukturen, sowie einzelne Arbeitergebäude
- einen administrativen Bereich mit einer Festung, einer Kathedrale, inklusive eines grünen Bereiches mit Natur, und einer Wettermaschine, die rein zur Dekoration vorläufig existiert. Eventuell kann ihr später noch eine Funktion zugeordnet werden.

Verteilt über die gesamte Stadt sollen einfache Banken sich erstrecken um zu zeigen, dass in der gesamten Stadt gewirtschaftet wird; nur an einer gewissen Stelle, an der sich dann das Gold sammelt.

Zuerst wird mit dem produktiven Bezirk angefangen. Bei der Schwefel- und Eisenveredelungen stechen vor allem gewisse Gebäudetypen heraus: Schmieden, Alchemistenhütten und Büchsenmachereien. Durch geschicktes Wenden und Drehen der Gebäude lassen sich Gebäudekomplexe erschaffen, wobei die Arbeitsstätten trotzdem von den Arbeitern aufgesucht und benutzt werden können. Dafür muss überwiegend der Eingang frei betretbar sein.

Die Alchemistenhütten und die Büchsenmachereien machen hierbei den Anfang:



Aneinander gereiht wirken sie wie eine größere Fabrik und können gut ineinander gebaut werden. Ein Astrolabium ziert eine blanke Stelle und dient der Verschönerung.

Nebenan wurden verschiedene Schmieden gesetzt. Durch ihre unterschiedlichen Stufen können sie gut ineinander gesetzt werden und so für eine schöne Ästhetik gesorgt werden. Zusätzlich wurde ein Eisenhaufen in eine Ecke der Schmieden gesetzt, da dieser thematisch perfekt passt. Dieser wurde mit einem Zaun umrandet für den Fall, dass später die Leibeigenen des Spielers 2 diesen abbauen wollen und jener dies verhindert. Fässer und auf der Rückseite der Häuser zeigen eine Lagerung, auch wenn dies nur dekorativ ist.



Insgesamt besteht der produktive Bezirk also aus Schwefel- und Eisenveredelern verschiedener Stufen, sowie einer wirtschaftlichen Bank. Zusätzlich befinden sich Häuser und Essenplätze in der Umgebung, damit die Arbeiter dort wohnen und essen können. In mitten aller Arbeiter wurde ein Lager platziert. Aus dem Lager holen die Arbeiter sich ihre Rohmaterialien. Wenn das Lager nicht dort wäre, würden sich die Arbeiter ihre Materialien aus der Burg/Festung holen, welche sich nicht in der Nähe befindet und deshalb lange Menschenzüge entstehen würden. Das wirkt sich kontraproduktiv auf ein Stadtbild aus und wäre somit nachteilig gegenüber der städtischen Atmosphäre.

Neben einem Lager sind auch Verzierungen angebracht worden. So finden sich am Lager und an den Gutshöfen Blumen, Uhren und Brunnen wieder. In das Lager sind auch Wohnhäuser gesetzt worden, und zwar so, dass ihre Eingänge frei zugänglich sind und somit nutzbar sind. Außerdem sieht das Lager so größer aus, als es eigentlich wäre und es kann Platz eingespart werden.



Zum produktiven Bezirk ist ein militärischer Bezirk geplant. Zu einem militärischen Bereich gehören Ausbildungsplätze für verschiedene Arten von Truppen; von Nahkämpfern über Fernkämpfer bis hin zu schweren Geräten, wie Kanonen. Die Gebäudetypen für diese Truppen sind Kasernen, Schießplätze und Kanonengießereien. Die Gruppe der gehobenen Varianten passen besser ins Stadtbild, weshalb ausgebauten Bauten, wie Garnisonen, Schießanlagen und Kanonenmanufakturen benutzt werden. In der Theorie gibt es noch Ställe und Reitereien, welche allerdings aufgrund von Platzproblemen entweder in den äußeren Ring oder gar nicht platziert werden.

In Kombination mit Türmen und einiger militärischer Kunst kann angrenzend an den produktiven Bezirk der Militärbezirk entstehen.



Obwohl Strukturen sich innerhalb der Ausbildungsgebäude befinden, könnten die Truppen, welche dort ausgebildet werden, frei trainieren und werden nicht blockiert, was zu einer potentiellen Löschung der Einheit geführt hätte. Einheiten, die sich im Blocking anderer Entitäten bewegen, laufen Gefahr zu sterben.

Die Strukturen innerhalb der Gebäude bestehen aus Ballistatürmen, die eine militärische Präsenz darstellen sollen. Hinzu wurde eine Statue „Kameraden“ an eine Seite der Schießanlagen gestellt. Trotz, dass sie zu den Dekorationen gehört, passt sie gut in diesen Distrikt aufgrund ihrer Beschreibung als Gedenken an gefallene Soldaten.

Um diesen Distrikt von stark anders-thematisierten Distrikten abzugrenzen wurde eine Mauer gebaut. Einen wirklichen Nutzen hat sie nicht, als mehr der Schönheit und der Ästhetik zu dienen. Als Vorbild für derartige Aufteilungen gelten verschiedene Städte aus Spiel und Film, wo man klare Abgrenzungen zu thematischen Gebieten in einer Stadt erkennen kann.

Angrenzend an den militärischen Distrikt befindet sich der Wohnbezirk. Der Wohnbezirk zeichnet sich, wie der Name schon sagt, durch überwiegend Wohnhäuser unterschiedlicher Größe und Form.



Die Wohnkomplexe bestehen dabei aus den 3 Wohnhaustypen, Refektorien, Skriptorien und den sogenannten Minenarbeiterhütten (welche nur Hütten im Spiel bezeichnet werden). Während die kleinen, mittleren und großen Wohnhäuser Arbeiter beherbergen können, haben die Skriptorien, Refektorien und Hütten keinen Platz um darin zu wohnen und sind somit nur kosmetisch.

In den Wohnkomplex sind einzelne Arbeitergebäude gesetzt worden, sowie Orte, an denen sich die Arbeiter ausruhen können. So sind z.B. eine Sägemühle, in die ein inaktives Signalfeuer platziert wurde, welches als dekorativer Holzstapel fungiert, eine Ziegelbrennerei und eine Steinmetze (im Bild in linken oberen Ecke). Sie geben dem Wohnviertel eine bunte Mischung als nur Wohngebäude und auch einige Leute, die durch die Arbeitergebäude umherlaufen. Um den Distrikt nicht zu heftig durch bestimmte Arbeitertypen zu charakterisieren, wurden unterschiedliche Arbeitergebäude genommen. In Mitten des Wohnviertels ist ein Marktplatz gesetzt worden. Da der Marktplatz die Ausbaustufe vom Lager ist, können die Arbeiter dort auch ihre Waren zum veredeln abholen. Ein Marktplatz ist in vielen Spielen meistens ein Mittel- bzw. Ankerpunkt von Städten. Diverse Leute und Personen beleben den Raum mit ihrer Präsenz und sorgen so für ein städtisches Gefühl; ebenso auch die Händler des Marktplatzes. Anbei sind einige Dekorationen, wie eine mechanische Uhr nahe der Wohnungen und ein Brunnen nahe des Marktplatzes, wie man es aus Bildern und Filmen wahrnimmt. Um dabei noch für Abwechslung im Stadtbau zu sorgen sind die Wege zwischen den Häusern unterschiedlich gehalten; mal sind die Straßen schmal und eng, und mal breit und offen, fast schon wie öffentliche Plätze.

Neben dem Stadtprofil sticht eine Sache heraus: Der Aufgang in den inneren Ring ist durch Säulen betont. Hier wurde anstatt einer weiteren Mauer oder starken Begrenzung eine sowohl optische als auch offene Variante benutzt. Durch die Zwischenräume der

Säulen können Einheiten weiterhin durchlaufen und doch symbolisieren sie eine gewisse Trageweite, dass dies der Aufgang in den inneren Ring ist.

Insgesamt erscheint der Wohnbezirk von dem selben Typ an Arbeitern nicht überfüllt, sondern gut durchmischt, was ebenso für die Gebäude, wie auch engen und breiten Straßen und Plätzen gilt.



Im letzten Abschnitt des mittleren Ringes, dem administrativen Bereich, befinden sich allgemeine Gebäude. Die Steinmetze gehört noch in den Wohnbezirk, sodass die Grenze zwischen diesen beiden Bereichen verschmilzt. Da im inneren Ring es kein Hauptgebäude eines Spielers gibt (Zitadelle, Festung, Burg), wurde eins passend in den mittleren Ring integriert. Neben der Burg wurde eine Kirche gesetzt mit einer passenden Grünanlage. Es wurde explizit keine Kathedrale genommen, da diese bereits im inneren Ring vorhanden ist und auch nicht zum Design des mittleren Ringes passt. Die Grünanlage daneben wurde mit Pflanzen, Bäumen und einem Pavillon ausgestattet, zu dem eine kleiner Weg, gebaut aus 4 verkleinerten Holzpfeilern und 14 Treibhölzern, läuft. Der Pavillon ist umringt von einer kleinen Wasseranlage. Dabei wurde der Wassertyp an dieser Stelle passend geändert und auf eine leicht niedrigere Höhe als der Boden angehoben. Anschließend wurde der Boden an dieser Stelle, der sich zu dem Zeitpunkt noch über dem Wasser befindet, abgesenkt, sodass das Wasser hervor tritt. Dies wurde fast einmal um den Pavillon herum gemacht, sodass ein fast geschlossener Ring aus Wasser entstand. Der letzte Teil wurde über Wasser gelassen und mit den Holzpfeilern und Treibhölzern optisch zu einer kleinen Brücke gebaut.



Dazu sind vereinzelt kleine Steine um die Wasseranlage, sowie auf der anderen Seite ein Refektorium und zwei Holzbänke gesetzt worden. Insgesamt steht dieser Ort als ein guter Kontrast zum Rest der Stadt mit einer friedlichen und idyllischen Atmosphäre.

Gegenüber der Steinmetze befindet sich ein Wetterturm, benachbart von zwei Wetterkraftwerken. Auch wenn diese im Spiel eigentlich dazu da sind, um das Wetter zwischen Sonne/Sommer, Regen und Winter zu wechseln, sind sie momentan noch reine Dekorationselemente. Zwischen den Gebäuden ist jeweils eine Dampfmaschine um eine Verbindung zwischen den Gebäuden zu symbolisieren; ebenso nur Dekoration.

Was über den kompletten mittleren Ring auffällt: Die Mauer hat sich verändert.

Zuvor war die Mauer gerade und hatte lediglich zwei Ecken. In der Zwischenzeit hat sie eine Struktur und ein Profil bekommen. In regelmäßigen Abständen sind Einbuchtungen in die Mauer verbaut, in denen Ballistaturme hineingestellt wurden.



Der linke und der rechte Teil der Mauer wurde mit dem gleichen Muster überarbeitet, sodass in gleichen Abständen ein Turm dazwischen passt. Im mittleren Teil wurde dagegen ein anderes Aussehen gewählt: Zu den Ecken hin wurde erst ein Ballistatum in jeweils eine Einbuchtung gesetzt und nach den Ecken ist ein individuelles Profil für die Mauer entstanden.

Eine weitere Besonderheit in sowohl dem inneren Ring, als auch im mittleren Ring, ist die Anzahl an Wohnhäuser und Bauernhöfen/Farmen/Gutshöfen. Sobald Arbeiter keinen Schlaf- oder Essensplatz haben, gehen sie nach der Arbeit an ein Lagerfeuer um sich dort zu erholen. An sich ist das nicht weiter schlimm, da in diesem Fall weder auf die Anzahl an Rohstoffen durch die Arbeiter, noch die Effizienz der Arbeiter selbst geachtet werden muss. Allerdings machen Lagerfeuer in einer relativ bewohnten Umgebung wenig Sinn und sind auch im wirklichen Leben nicht in Städten auf den Straßen und großen Plätzen vorhanden. Deshalb wurde explizit darauf geachtet, dass so viele Schlaf- und Essensplätze so zur Verfügung stehen, sodass keine Lagerfeuer entstehen sollten.

Nach der Vollendung des inneren und mittleren Ringes geht es nun an den äußeren Ring.

Der äußere Ring ist weitgehend und der größte von allen. Hier befinden sich Bereiche, die eigene Thematiken haben, wie Produktionsstätten für einen bestimmten Rohstoff, aber

auch Ackerfelder und friedliche Strukturen, wie kleine Klöster; Objekte, die man überwiegend in ländlichen Regionen findet.

Zuerst wird in einer Ecke angefangen: In der rechten Ecke wird ein Holzfällerlager entstehen. Zu einem Holzfällerlager gehören immer Sägemühlen, viele Bäume und Lagerholz. Die Holzhaufen können, wie schon im mittleren Ring, durch inaktive Signalfeuer imitiert werden.

Es werden sowohl Sägemühlen Stufe 1 als auch Stufe 2 platziert und mit vielen inaktiven Signalfeuern als dekorative Holzstapel. Zu den Arbeitsstätten werden Häuser und Bauernhöfe dazugestellt um die Arbeiter zu unterhalten, damit keine Lagerfeuer entstehen. Die Vermeidung von Lagerfeuer ist hierbei nicht notwendig, da es sich hier mehr um die niedrigere Bevölkerung handelt als die normale Bevölkerung im mittleren oder inneren Rang handelt. Zu den Sägemühlen wurde ein Lager platziert, damit die Sägewerker nicht den gesamten Weg bis zum Marktplatz im mittleren Ring laufen müssen, um sich Holz zu holen, sondern ihre Ware direkt neben sich abholen können.



Umringt ist das Waldlager von verschiedenen Bäumen, die der Region entsprechend; überwiegend Tannen und Nadelbäume. Da es sich um eine äußere Region handelt, ist an dieser Stelle auch nur ein Dorfzentrum statt eines Gemeinde- oder Stadtzentrums.

In der anderen Ecke des äußeren Ring wird eine andere Thematik orientiert sein. Während sich in der rechten Ecke sich eine Arbeitersiedlung an Holzfällern sich angesiedelt hat, befindet sich in der linken Ecke eine mehr geistlich orientierte Abteilung; nämlich ein kleines Kloster.



Der geistliche Bezirk besteht überwiegend aus dementsprechenden Gebäuden, wie einer Kapelle und einer Abtei. Zusätzlich wurden Häuser und Bauernhöfe zur Verfügung gestellt, um Lagerfeuer zu verhindern. Neben der Kapelle und Abtei wurde ebenfalls ein Dorfzentrum platziert, sowie weitere dekorative geistliche Gebäude; nämlich Refektorien und Skriptorien. Mit einer Kirche/Abtei wird oftmals auch ein Friedhof assoziiert. Ein solcher Friedhof wurde hinter der Abtei mit Gräbern, einem kleinen Kiespfad (Bodentextur), sowie Grünzeug (Bäume) und einem Brunnenhaus angelegt. Die ländliche Region widerspiegeln wurde an einen Bauernhof ein Feld mit Raps und Sonnenblumen angelegt. Umringt wurde die Siedlung von Apfel- und Orangenbäumen, um eine friedlichere Atmosphäre zu verstärken; auch weil Kloster oftmals noch Anlagen haben, in denen sie Obst und Gemüse anbauen.

Zwischen der geistlichen linken Ecke und der handwerklichen Holzfällerei-Ecke rechts wurde in die Mitte noch eine Handwerker Siedlung gebaut. Diese Siedlung sticht durch ihren starken Fokus auf Lehm hervor. Der Boden besteht komplett aus unterschiedlichen Erdtexturen und somit wurde in die Mitte dementsprechend eine der Lehmgruben mit eingesetzter Lehmmine Stufe 3 gesetzt. Da ein Weg durch das Gebiet führt, teilt dieser es in zwei Teile. Man könnte zwei von einander getrennte Gebiete bauen, wobei das untere allerdings kleiner und damit weniger Platz hätte. Deshalb wurde auf den Bau von zwei unterschiedlichen Gebieten verzichtet und stattdessen die Lehmproduktion auf die untere Hälfte erweitert, sodass dort eine weitere Lehmmine ihren Platz gefunden hat.

Passend zu den Lehmminen wurden Ziegelhütten und Ziegelbrennereien dazu gestellt, jeweils auch mit Haus und Bauernhof um die Arbeiter zu versorgen. Genauso wie in den beiden anderen Regionen, wurde auch hier ein Dorfzentrum platziert.



Um ein wenig Abwechslung in die Siedlung zu bringen wurde im unteren Gebiet eine Art kleines Lager mit Kisten, Fässern und kleinen Zelten gebaut. An sich haben diese Elemente keine Bedeutung und dienen dadurch nur der Dekoration und auch der Abwechslung. Um die einfachen Verhältnisse noch zu verdeutlichen befindet sich in mitten der Dreier-Ziegelhütten-Konstellation ein brennendes Signalfeuer. Ob man das nun als Lagerfeuer, oder allgemeine Feuerstelle für irgendetwas interpretiert, bleibt jedem selbst überlassen.

All diese Gebiete stehen in direktem Kontrast zu einander. Während im linken Drittel die geistlichen in ihrem Kloster friedlich leben, sind die anderen beiden Dritteln am Arbeiten mit Werkzeug und Maschinen. Und auch hier gibt es Unterschiede von den Begebenheiten. Die einen arbeiten im Wald mit Holz und Bäumen, während die anderen in den Minen und mit Erde arbeiten.

An den Grenzen dieser Siedlungen im äußeren Ring zueinander verfällt die Thematik der Region und es bleibt Platz für allgemeine Strukturen bzw. Strukturen, die nicht unbedingt ein gewisses Thema haben. So hat sich zwischen der Lehmproduktion und dem Kloster eine kleine Gartenanlage mit Blumenfeldern, einem Brunnen, einem Brunnenhaus und Bänken angelegt. Die Blumen sind dabei von leicht vergrößerten Steinen umringt um das Beet zu zeichnen.

Auf der anderen Seite wurde in die angrenzenden Bäume eine Reiterei, umzingelt von Holzpalisaden platziert. Die Reiterei kann für zwei Möglichkeiten später genutzt werden: Entweder sie wird als die militärische Reiterei benutzt, die in der Stadt weder Platz noch Ästhetik hatte, oder sie ist ein Ort von Freizeitreitern, die dort ihre Niederlassung haben. Die jeweilige Aufgabe der Reiterei kann später im Verlauf der Erstellung des Skriptes entschieden werden.



Mit der Fertigstellung des äußeren Rings, bestehend aus den drei Gebieten, dem Geistlichen, der Lehmproduktion und der Holzproduktion, sowie den Strukturen zwischen den Gebieten, ist somit die Stadt vollendet. Sie besitzt einen inneren Ring bestehend aus edlen Gebäuden und Straßen, einen mittleren Ring, der überwiegend die Administration, sowie den militärischen Aspekt in Form von Ausbildungsbauten und militärisch orientierten Produktionsgebäuden und einem Wohnbezirk beinhaltet. Mit dem äußeren Ring sind auch die ländlichen Regionen in Form von Materialproduktionsstätten und einer kirchlichen Einrichtung vorhanden.



## Das Banditenlager:

Der letzte Abschnitt der Karte, welcher noch nicht bearbeitet wurde, soll ein Banditenlager beinhalten. Da die Karte nicht von großer Schwierigkeit in Bezug zu Spielmechaniken und Können gestaltet sein soll, ist das Lager dementsprechend auch einfach gehalten. Die Truppen sind nicht zu stark, sollen im Laufe der Karte aber eine Bedeutung bekommen, von der man zumindest sagen kann, dass sie nicht unterschätzt werden sollen.

Wenn man an ein Banditenlager denkt, kommen vor allem folgende Punkte in den Sinn:

- Der Ort ist abgelegen
- Es besteht nicht aus städtischen Häusern
- Mögliche Verteidigungsanlagen

Der Ort des Banditenlagers befindet sich im östlichen Teil der Karte in einem Waldgebiet außerhalb des Stadtgebiets. Damit ist der Ort abgelegen. Als Gebäude werden Zelte und Banditentürme benutzt. In der freien Wildnis stehen nicht häufig solide Gebäude, sondern mehr kleinere, instabillere Zelte bzw. Gebäude mit einer gewissen Thematik, wie eben Türme, die wie Gefängnistürme aussehen. Bei den Verteidigungsanlagen können für den Anfang Aussichtstürme genutzt werden. Diese haben keinen Mehrwert bezüglich ihrer Funktion, allerdings lassen sie sich zu Türmen mit einer Balliste aufwerten, wodurch sie, gerade für den geschichtlichen Verlauf der Karte, eine große Rolle spielen. Um das ganze abzurunden wird ein Palisade um das Lager herum gebaut als eine zusätzliche Schutzmaßnahme um nur von einer Seite später angreifen zu können. Der rote Teich, der nebenan angesetzt wurde, ist noch ein Artefakt aus der Planung, aber als dekoratives Element eigentlich ganz passend, gerade wenn man bedenkt, dass es sich um wilde Banditen handelt.



## Der Verlauf der Karte:

Bevor es an die Entwicklung der Karte durch ein Skript geht, werden erstmal die Begebenheiten des Spiel selbst genauer erklärt. Nicht jedes Spiel hat die gleichen Voraussetzungen und die selbe interne Struktur um gewisse Elemente zu verbauen. Deshalb wird hier auch auf Besonderheiten eingegangen.

## Technische Spezifikationen:

Das Spiel „Die Siedler – Das Erbe der Könige“ hat 2 Ebenen, auf denen das Spiel basiert. Das ist einmal die C-Ebene, auf der die ganze Logik des Spiels programmiert ist, mit der man nicht viel interagieren kann (Modifikationen ausgenommen), und einmal die Lua-Ebene. Hierbei wird eine spezielle Lua-Version benutzt, nämlich eine händisch Veränderte basierend auf der Version 5.0.0. Wichtig ist hierbei zu wissen, dass nicht alle Funktionen, die es in der 5.0.0 gab, übernommen wurden, wie z.B. die Funktion `require()` oder die Module um IO, welche für Input und Output verschiedenster Arten zuständig ist. Gleichermaßen ist es wichtig zu wissen, welche Basis-Funktionen das Spiel mit sich bringt und wie das Lua-Interface zu den C-Funktionen verbaut ist. Hierzu gibt es von jemandem, namens mcb, in der Siedler 5 Multiplayer Community ein Github-Repository, welches die gesamte Lua-Reference mit Dokumentation basierend auf dem Spiel auflistet. Nicht nur erfasst sie Standard-Funktionen, sondern auch erweiterte Funktionen, welche durch Modifikationen hervorgebracht werden, aber dies nur am Rande.

Das Repository kann man hier finden: [mcb's S5LuaReference](#)

Neben Lua-Kenntnissen ist auch ein gewisses Wissen über das Spiel selbst nützlich. Das Spiel arbeitet auf der Basis von Ticks, wobei 10 Ticks eine Sekunde sind. Das heißt es gibt insgesamt 3 Tick-Varianten, wenn man von anderen Events absieht:

- Ein Tick (1/10 Sekunde)
- 1 Sekunde (10 Ticks)
- 1 Bild pro Sekunde

Neben den Ticks gibt es auch Funktionen im Graphical User Interface, die, anstatt an die Ticks, an die Bilder pro Sekunde gebunden sind. Dadurch fallen deren Funktionsaufrufe zwischen die Ticks des Spiels und können dadurch zeitlich genauer abgepasst werden.

Eine nützliche Informationen über das Spiel ist auch, dass es Single-Thread läuft. Heißt, verläuft sich das Skript in eine sich wiederholende Sequenz von Aufrufen, friert das Spiel ein und der Prozess muss von außen beendet werden.

Das Spiel wird standardmäßig mit vielen Skripten und Funktionen geliefert, wie `Comfort.lua` oder `Support.lua`, in denen viele Funktionen definiert sind. Man kann die Skripte unter `base` (dem Basisspiel), `extra1` (Nebelreich) und `extra2` (Legende) unter `data/script` nachlesen, wobei `data` das Hauptverzeichnis der Spielversion (`base`, `extra1`, `extra2`) ist. Sollten an der Stelle von `data` (oder auch `shr` genannt) sich eine `bba`-Datei

befinden, so muss diese erst mit dem sogenannten bba-Tool geöffnet werden (ein externes Programm).

Neben Skripten, die im Spiel bereits vorhanden sind, ist es auch möglich weitere Skripte hinzuzuladen. So gibt es eine S5CommunityLib, ebenfalls in einem Repository von mcb, in der sich viele unterschiedliche Funktionen aus der Community befinden, die mal größer, mal kleiner ausfallen. Im Laufe der Karte werden auch Funktionen aus dieser Bibliothek verwendet.

Das Repository zur S5CommunityLib kann man hier finden: [mcb's S5CommunityLib](#)

Ebenfalls ist der Inhalt einer Karte von Bedeutung um zu wissen, was man auf welche Art und Weise alles verändern kann. Eine Karte beinhaltet 5 wichtige Dateien neben der initialen Skriptdatei:

- Info.xml (beinhaltet die Informationen zu welcher Spielversion die Karte gehört, eine einzigartige ID und Informationen bezüglich Einzelspieler/Mehrspieler und den Titel, sowie die Beschreibung für die Auswahl der Karte im Spiel)
- TerrainData.bin (beinhaltet die Terrain Daten, wie Höhen, Tiefen, Texturen und Wasser)
- MapVertexColors.bin (beinhaltet die Vertexcolors, die auf der Karte verwendet wurden)
- <Kartenname>.png (Ist die Karte als Bild in der Vorschau)
- MapData.xml (beinhaltet alle Entitäten in Platzierungsreihenfolge mit vielen weiteren Einträgen, wie Modelgröße, Rotation, Spieler, Scriptcommandline\*, X und Y Positionen, und weitere)

\*mit der Scriptcommandline kann man bereits bei der Erstellung der Entität Luakommandos ausführen, wie z.B. das Füllen der Minen.

Als letztes beinhaltet die Karte die Skriptdatei, welche beim Starten der Karte aufgerufen wird, nämlich die MapScript.lua. Die Datei muss so heißen, andernfalls wird die Karte ohne ein Skript gestartet und verweilt in ihrem Ausgangszustand. Sie ist der Einstiegspunkt einer jeder Karte und sie wird als erstes ausgeführt, nachdem alle Entitäten platziert wurden.

Nebenbei ist es noch praktisch den LuaDebugger der Community über den S5Updater (den einfachen Modus ausschalten) zu installieren. Er bietet sowohl die Möglichkeit Skripte zu lesen, als auch Aus- und Eingaben zu tätigen und bleibt bei Ausnahmen hängen und gibt einem den Stacktrace zurück um nachzuvollziehen, warum etwas im Skript nicht funktioniert hat.

Weiterhin ist zu beachten, dass die Karte noch weiter bearbeitet wird mit so genannten Skriptentities. Diese sind für den Spieler unsichtbar und dienen dazu für das Skript als Punkte, die benannt und ausgewählt werden können. So müssen Koordinaten nicht hardcoded sein.

## Der Einstieg in das Skript:

Mit der MapScript.lua fällt auch das erste Stück Code in die Karte. Nun könnte man direkt los legen, seine eigenen Funktionen reinschreiben und alles im Kopf selbst zusammenbauen. Man könnte aber auch die Vorlage von Siedler nutzen, die sich unter data/script/maptools/mapscript.lua liegt, und den Code

```
-- MapName: XXX
--
-- Author: XXX
--

-- Include main function
Script.Load( Folders.MapTools.."Main.lua" )

-----+
-----+
-- This function is called from main script to initialize the diplomacy states
function InitDiplomacy()
end

-----+
-----+
-- This function is called from main script to init all resources for player(s)
function InitResources()
end

-----+
-----+
-- This function is called to setup Technology states on mission start
function InitTechnologies()
end

-----+
-----+
-- This function is called on game start and after save game is loaded, setup your
weather gfx
-- sets here
function InitWeatherGfxSets()
    SetupNormalWeatherGfxSet()
end
```

```

--+
--+
-- This function is called on game start you should setup your weather periods here
function InitWeather()
    AddPeriodicSummer(10)
end

--+
--+
-- This function is called on game start and after save game to initialize player colors
function InitPlayerColorMapping()
end

--+
--+
-- This function is called on game start after all initialization is done
function FirstMapAction()
    -- debugging stuff

    EnableDebugging()
end

```

entweder in eine neu zu erstellende Datei mit dem Namen MapScript.lua packen oder die Datei gänzlich kopieren. Dies wird auch gemacht, sodass nun ein funktionierendes Skript schon mal vorhanden ist. Hierzu eine Erklärung, was genau eigentlich in diesem Codeblock passiert:

Zuerst wird ein Skript Main.lua geladen. Die Main.lua ist die einfache Variante vom Laden vieler bereits vorhandener Skripte und Funktionen, sodass man diese nicht manuell selbst alle einzeln laden muss, darunter auch den Callback, der beim Spielstart aufgerufen wird. In diesem Callback werden weitere Skripte geladen, sowie Variablen gesetzt und auch die Init-Funktionen aus dem Mapscript aufgerufen. Zum Schluss wird die FirstMapAction (FMA) aufgerufen, wodurch die Karte ins Laufen gerät. Alles was nach dem Start der Karte passieren soll, welches der Kartenersteller selbst schreibt, kommt in die FMA bzw. auf Routen, die durch die FMA aufgerufen werden.

Nachdem das Initialskript aufgesetzt wurde, können Ordner und Skripte für Kapitel angelegt werden. Die S5CommunityLib wird dabei auch in den Kartenordner gelegt. In der dieser Community-Bibliothek befindet sich ein Skript, welches das abhängige Laden von Skripten stark vereinfacht und ein equivalent zu Packagemanagern darstellt; der mcbPacker, welcher sich unter S5CommunityLib/packer/devLoad.lua finden lässt. Dieses Skript wird über `Skript.load` geladen, sodass mit folgender Pfadsetzung alle benötigten Skripte aus der Community von alleine geladen werden können, sobald das gewünschte Skript über das `mcbPacker.require()` geladen wird.

```
Script.Load(Folders.Map.."s5CommunityLib/packer/devLoad.lua")
mcbPacker.Paths = {{Folders.Map,".lua"}}
mcbPacker.require("s5CommunityLib/lib/UnlimitedArmy")
```

Neben den Skripten muss durch eine Abhängigkeit, dem Triggerfix, eine Zeile ausgeführt werden, nachdem alle Skripte geladen wurden.

#### **TriggerFix.AllScriptsLoaded()**

Eine weitere Sache, die bedacht werden muss, ist, dass der Held „Pilgrim“ auch Fähigkeiten hat, die es ihm auch möglich machen, bestimmte Ziele zu erreichen, die er nicht erreichen sollte. An dieser Stelle kann entweder die Ausführung der Fähigkeit verhindert oder der Knopf für die Fähigkeit gänzlich deaktiviert werden. Der Einfachkeithalber wird der Knopf unsichtbar geschalten. Da es sich bei dem Knopf um die GUI handelt, wird dieser mit jedem erneuten Laden zurückgesetzt; heißt wieder sichtbar geschalten.

Um die GUI zu bearbeiten bzw. zu wissen um welchen Zweig es sich handelt, kann der [S5GUIEditor](#) benutzt werden; ebenfalls von mcb. Oder man versucht es aus der XML des Spiels herauszulesen, was nicht empfehlenswert ist.

```
function HidePilgrimAbilities()
    if not NotHideAbilities then
        XGUIEng.ShowWidget(XGUIEng.GetWidgetID("Commands_Hero2"), 0)
    else
        XGUIEng.ShowWidget(XGUIEng.GetWidgetID("Commands_Hero2"), 1)
    end
end
AddMapStartAndSaveLoadedCallback(HidePilgrimAbilities)
TriggerFix.AllScriptsLoaded()
```

## Kapitel 1 Auf der Suche nach etwas:

Der Spieler beginnt die Karte mit einer Figur, die durch den Wald streift und nach Zeichen von Leben sucht. Dabei findet die Figur unterschiedliche Leute und unterhält sich mit denen.

Die unterschiedlichen Leute sind:

- Der Einsiedler, der etwas über eine Prinzessin auf der anderen Seite der Mauern erzählt.
- Leonardo, der Erfinder, der mit Pflanzen und anderen Materialien experimentiert und dabei ein Problem mit Wölfen hat.
- Die ungehobelten Banditen, welche den Spieler zurück drängen auf die anderen Flussseite.
- Die Wachen an den Toren, die dem Spieler sagen, dass sie hier nicht weiter kommen.

Da die Figur unter die Helden-Kategorie fällt, kann er mit Figuren interagieren und so kann das ebenfalls als Trigger durch einen integrierten Callback dienen.

Bis auf das Banditencamp bekommen alle Figuren ihre Interaktion, indem als sie NPC erstellt werden.

```
NPCHermit={}
NPCHermit.name ="Hermit"
NPCHermit.callback=CreateNPCHermitBriefing

CreateNPC(NPCHermit)
```

Anschließend werden Briefings gestartet. Briefings sind relativ einfach gehalten und werden seitenweise geschrieben

```
BriefingWache1 = {}
BriefingWache1.finished = BriefingWacheFinished
local page = 1

BriefingWache1[page] = {}
BriefingWache1[page].title = "Wache"
BriefingWache1[page].text = "Hier kommt keiner durch! Verschwindet!"
BriefingWache1[page].position = GetPosition("Wache1")

StartBriefing(BriefingWache1)
```

Mit fast jedem Briefing (Wachen ausgenommen, da zählt es nur einmal), welches im ersten Kapitel abläuft, wird ein Zähler hochgezählt, der bei einer bestimmten Zahl ein Ereignis auslöst; nämlich die Fortsetzung der Geschichte beim Leuchtturm auf dem südlichen Hügel.

Sobald die Interaktionen fertig sind, müssen die Wachen beachtet werden, sodass deren Interaktionen entfernt werden. Schließlich sollen die Wachen nicht sagen, dass die Tore geschlossen bleiben, obwohl die Tore in diesem Moment geöffnet werden.

```
DisableNpcMarker("Wache1")
DisableNpcMarker("Wache2")
DisableNpcMarker("Wache3")
```

Die Banditen, welche im Banditencamp angesiedelt sind, werden über eine UnlimitedArmy gesteuert. Die UnlimitedArmy (UA) ist ein Armeensystem, welches von mcb entwickelt wurde, und verfügt über die Eigenschaften von objektorientierter Programmierung. Die UA soll dabei das Camp verteidigen.

```
Bandits = UnlimitedArmy:New({
    Player = 3,
    Area = 2900,
    AutoDestroyIfEmpty = true,
    TransitAttackMove = true,
    Formation = UnlimitedArmy.Formations.Chaotic,
    PrepDefense = false,
    DestroyBridges = false,
    LeaderFormation = 4,
    AIActive = true,
    DefendDoNotHelpHeroes = false,
    AutoRotateRange = 10000,
    DoNotNormalizeSpeed = true,
    IgnoreFleeing = false,
    HiResJob = false
}
)

local _spawnpos = GetPosition("BanditCampSpawn")
Bandits>CreateLeaderForArmy(Entities.PU_LeaderBow3,8,_spawnpos,0)
Bandits>CreateLeaderForArmy(Entities.PU_LeaderBow3,8,_spawnpos,0)
Bandits>CreateLeaderForArmy(Entities.CU_BanditLeaderSword1,8,_spawnpos,0)
Bandits>CreateLeaderForArmy(Entities.CU_BanditLeaderSword1,8,_spawnpos,0)
Bandits>CreateLeaderForArmy(Entities.CU_BanditLeaderSword1,8,_spawnpos,0)
Bandits>CreateLeaderForArmy(Entities.CU_BanditLeaderSword1,8,_spawnpos,0)
Bandits>AddCommandDefend(GetPosition("BanditCampSpawn"),2800,,false)
```

Sobald die Interaktionen alle zusammengerechnet wurden kann das zweite Kapitel am ersetzen, aktivierten Leuchtturm starten.

```

function InteractionsFoundFinished()
    if InteractionsFound >= 4 then
        CreateBriefingGoToLighthouse()
        return true
    end
    return false
end

```

---

```

function BriefingActivateLighthouseFinished()
    ReplaceEntity("Leuchtturm", Entities.CB_LighthouseActivated)
    CreateGuardBriefing()
end

```

Ein graphischer Zusatz wurde noch am Ende von Leonards Briefing hinzugefügt, nachdem man die Wölfe in seinem Umfeld getötet hat. Beim Sprengen der Steine wurden visuelle Sprengeffekte erzeugt.

```

function BriefingLeonardo2Finished()
    ResolveBriefing(BriefingLeonardo)
    for i = 1,6,1 do
        local _stonepos = GetPosition("stein"..i)
        Logic.DestroyEntity(GetID("stein"..i))
        Logic.CreateEffect(GGL_Effects.FXExplosion, _stonepos.X, _stonepos.Y,0)
    end
    InteractionsFound = InteractionsFound + 1
    InteractionsFoundFinished()
    return true
end

```

Das Überprüfen auf lebendige Wölfe und Erstellen von ihnen wird zuvor über einen Funktionsaufruf und einen SimpleJob geregelt, welcher jede Sekunde einmal tickt.

```

function CreateWolfs()
    local _wolfPos = GetPosition("WolfSpawn1")
    Logic.SetEntityName(Logic.CreateEntity(Entities.CU_AggressiveWolf,
        _wolfPos.X,_wolfPos.Y,0,5),"Wolf1")

    local _wolfPos = GetPosition("WolfSpawn2")
    Logic.SetEntityName(Logic.CreateEntity(Entities.CU_AggressiveWolf,
        _wolfPos.X,_wolfPos.Y,0,5),"Wolf2")

    local _wolfPos = GetPosition("WolfSpawn3")
    Logic.SetEntityName(Logic.CreateEntity(Entities.CU_AggressiveWolf,
        _wolfPos.X,_wolfPos.Y,0,5),"Wolf3")

    StartSimpleJob("CheckIfWolfsDead")
end

function CheckIfWolfsDead()
    if IsDead("Wolf1") and IsDead("Wolf2") and IsDead("Wolf3") then
        CreateLeonardoBriefing2()
        return true
    end
    return false
end

```

## Kapitel 2 Die große Stadt:

In Kapitel 2 geht es darum, dass Pilgrim erst mit den Wachen, die ihn nach dem Entzünden des Leuchtturms auffinden, mit in die Stadt reist. Dabei soll er immer beim nächsten Wegpunkt bis auf eine gewisse Distanz heran kommen, damit die Truppen weiter laufen. Erst beim letzten Wegpunkt wird die Interaktion mit der Prinzessin frei geschalten. Ebenso sind nun die Tore offen, wodurch Pilgrim in die Stadt laufen kann.

```
function WaitForPilgrimThenMoveToNext(_waypointNumber)
    if GetDistance("Pilgrim","Waypoint".._waypointNumber) <=1000 then
        if _waypointNumber+1 >= 9 then
            Logic.GroupAttackMove(GetID("General"),
                GetPosition("GeneralLastPos").X,
                GetPosition("GeneralLastPos").Y,
                90)
            for i = 1 ,4, 1 do
                Logic.GroupAttackMove(GetID("Soldier"..i),
                    GetPosition("Soldier"..i.."LastPos").X,
                    GetPosition("Soldier"..i.."LastPos").Y,
                    270)
            end
            LetPrincessTalk()
            return true
        end
        _newPoss = {}
        CallFuncWithCirclePositions("Waypoint".._waypointNumber+1, 500, 72,
            ReturnCirclePosToTable)
        Logic.GroupAttackMove(GetID("General"), _newPoss[1].X,
            _newPoss[1].Y, 0)
        for i =1,4,1 do
            Logic.GroupAttackMove(GetID("Soldier"..i), _newPoss[i+1].X,
                _newPoss[i+1].Y, 72*i)
        end
        _newPoss = nil
        StartSimpleJob("WaitForPilgrimThenMoveToNext",_waypointNumber + 1)
        return true
    end
    return false
end

function OpenDoors()
    ReplaceEntity("gate1",Entities.XD_DarkWallStraightGate)
    ReplaceEntity("gate2",Entities.XD_DarkWallStraightGate)
    ReplaceEntity("gate3",Entities.XD_DarkWallStraightGate)
end
```

Nachdem Pilgrim mit der Prinzessin gesprochen hat, wird ihm die Aufgabe gegeben die Gebiete der Stadt zu erkunden. Um dies zu realisieren wird eine Tabelle mit Key-Value-Paaren angelegt. Die Keys sind dabei die Orte, an denen er sich einmal befindet haben muss und der Value eine Flag als boolean.

Gleichzeitig wird noch eine sogenannte Questinformation in der oberen, linken Ecke des Bildschirms erstellt, woran der Spieler erkennen kann, in wievielen Bezirken er bereits war. Die Bezirke haben jeweils eine Skriptentity gesetzt bekommen, welche den Namen des Bezirks besitzt.

```
CityAreas = {}
CityAreas["innererRing"] = false
--CityAreas["Administration"] = false
CityAreas["wohnung"] = false
CityAreas["admin"] = false
CityAreas["militaer"] = false
CityAreas["geistlich"] = false
CityAreas["lehm"] = false
CityAreas["holz"] = false
CityAreas["waffenproduktion"] = false
Visited = 0

GUIQuestTools.StartQuestInformation("Serf", "Bereits besuchte Stadtgebiete", 1, 1)
XGUIEng.SetText("QuestInformationTooltipText", "Bereits besuchte Stadtgebiete")
GUIQuestTools.UpdateQuestInformationString(Visited .. " / 7")
StartSimpleJob("CheckIfPilgrimInAreaInCity")
```

Die Abfrage ob Pilgrim bereits dort gewesen ist, wird über den SimpleJob geregelt.

```
function CheckIfPilgrimInAreaInCity()
    for _places,_been in pairs(CityAreas) do
        if IsNear("Pilgrim",_places,2000) and not _been then
            CityAreas[_places] = true
            Visited = Visited + 1
            GUIQuestTools.StartQuestInformation("Serf", "Bereits besuchte Stadtgebiete",
                                              1, 1)
            XGUIEng.SetText("QuestInformationTooltipText", "Bereits besuchte
                           Stadtgebiete")
            GUIQuestTools.UpdateQuestInformationString(Visited .. " / 7")
        end
    end
    if Visited >= 7 then
        GoBackToPrincess()
        return true
    end
    return false
end
```

Jede Sekunde wird überprüft ob Pilgrim in einem der Bezirke ist und dieses auch noch nicht besucht wurde. Sobald diese Bedingungen übereinstimmen, wird die Flag von false auf true gesetzt, eine globale Variable um eins hochgezählt und die Questinformationen aktualisiert. Sobald die globale Variable die Anzahl an Bezirken erreicht hat, wird eine Interaktion mit der Prinzessin freigeschalten, wodurch die Geschichte in dem dritten Kapitel weiter spielt.

## Kapitel 3 Den Einwohnern helfen:

Nachdem Pilgrim die Stadt erkundet und wieder mit der Prinzessin gesprochen hat, gibt sie ihm die Aufgabe den Bewohnern der Stadt zu helfen. Zu den Aufgaben gehören:

- Der Steinmetz braucht Steine und Pilgrim soll sie sprengen
- Einem Sägewerkmitarbeiter ist sein Hund entlaufen und Pilgrim soll ihn finden.
- Die Banditen haben die Hütte des Einsiedlers niedergebrannt und Pilgrim soll ein neues Haus bauen.
- Der Steuereintreiber kann das Geld aufgrund seines gebrochenen Beines nicht von den Banken einsammeln. Pilgrim soll den Karen zu den Banken führen und das Geld abholen.
- Da die Banditen immer aggressiver werden, soll Pilgrim für den Captain mithilfe eines Kundschafters die Banditen auskundschaften.

Es werden für alle Interaktionen NPCs erstellt, sodass der Spieler selbst entscheiden kann in welcher Reihenfolge er die Aufgaben abarbeiten möchte. Ebenso wird über die Questinformation dem Spieler gezeigt, wie oft er bereits Personen geholfen hat.

Das Sprengen der Steine durch Pilgrim wird über einen Entity-Destroyed-Trigger abgefragt, der jedes mal tickt, sobald eine Entität zerstört.

```
function CheckStonesExploded()
    NotHideAbilities = true
    XGUIEng.ShowWidget(XGUIEng.GetWidgetID("Commands_Hero2"), 1)
    Trigger.RequestTrigger(Events.LOGIC_EVENT_ENTITY_DESTROYED,
                           nil, "StonesDestroied", 1, nil, nil)
end

Rocks = 0
function StonesDestroied()
    local _entID = Event.GetEntityID()
    if Logic.GetEntityType(_entID) == Entities.XD_RockDestroyableMedium1 then
        Rocks = Rocks + 1
        if Rocks >= 3 then
            XGUIEng.ShowWidget(XGUIEng.GetWidgetID("Commands_Hero2"), 0)
            IncreaseHelpedPeople()
            return true
        end
    end
    return false
end
```

Sobald alle drei Steine gesprengt wurden, wird der Counter für die Anzahl an geholfenen Leuten um eins erhöht.

Damit dem Captain geholfen ist, wird ein Kundschafter erschaffen, mit dem man durch seine Fähigkeit Den Banditeturm und die beiden Aussichtstürme vom Lager betrachten soll. Dafür sind Positionsabfragen und Abfragen auf Sicht auf diese Stellen nötig. Diese Abfragen werden in einem SimpleJob, der jede Sekunde tickt, getätigt.

```
function PlayerMustViewBanditCamp()
    local _scoutPos = GetPosition("militaer")
    scoutID = Logic.CreateEntity(Entities.PU_Scout,_scoutPos.X,_scoutPos.Y,180,1)
    StartSimpleJob("CheckForViewOnBandits")
end

_bandittowerPos = GetPosition("Banditeturm")
banditenDef1 = GetPosition("BanditenDef1")
banditenDef2 = GetPosition("BanditenDef2")

function CheckForViewOnBandits()
    if IsDead(scoutID) then
        Defeat()
        return true
    end
    if Logic.IsMapPositionExplored(1, _bandittowerPos.X, _bandittowerPos.Y) == 1 and
        Logic.IsMapPositionExplored(1, _banditenDef1.X, _banditenDef1.Y) == 1 and
        Logic.IsMapPositionExplored(1, _banditenDef2.X, _banditenDef2.Y) == 1 then

        CreateScoutedBriefing()
        return true
    end
    return false
end
```

Sobald die Positionen sichtbar sind, wird ein Briefing abgespielt, welches veranschaulicht, dass Pilgrim dem Captain geholfen hat.

Der Sägewerker sagt Pilgrim, dass ihm der Hund entlaufen ist. Sobald er angesprochen wurde, wird der Hund im südlichen Teil der Karte erstellt. Nebenan läuft ein Distanzencheck in einem SimpleJob, der den Hund Pilgrim folgen lässt, sobald er bei ihm war. Geholfen ist der Person, wenn der Hund beim Sägewerkmitarbeiter angekommen ist.

```

function PilgrimNearDog()
    if GetDistance("Pilgrim","Hund",1000) then
        CreateDogFoundBriefing()
        return true
    end
    return false
end

function DogFollowPilgrim()
    if GetDistance("Hund","serfWolfGone") <= 700 then
        CreateBriefingDogBroughtBack()
        return true
    end
    if Counter.Tick2("DogFollow",10) then
        Logic.GroupGuard(GetID("Hund"),GetID("Pilgrim"))
    end
    return false
end

```

Der Steuereintreiber möchte von Pilgrim, dass die Steuern von jeder Bank/Schatzkammer und den kirchlichen Einrichtungen (Kapellen, Kirchen, Kathedralen) abgeholt werden; heißt ein Karren wird erschaffen, der Pilgrim folgt, und Pilgrim muss mit diesem Karren zu allen Einrichtungen. Hierbei eignet sich eine Key-Value-Table.

```

function KarrenFollowPilgrim()
    GeldOrte = {}
    GeldOrte["Bank1"] = false
    GeldOrte["Bank2"] = false
    GeldOrte["Bank3"] = false
    GeldOrte["Bank4"] = false
    GeldOrte["Bank5"] = false
    GeldOrte["Kirche1"] = false
    GeldOrte["Kirche2"] = false
    GeldOrte["Kirche3"] = false
    StartSimpleJob("FollowPilgrim")
end

```

Nach dem Erschaffen folgt der Karren Pilgrim auf Schritt und Tritt durch die Stadt. Bei jeder Bank/Schatzkammer und jeder Kircheneinrichtung wird sowohl ein Ton abgespielt als auch eine Nachricht im Nachrichtenfenster angezeigt, dass das Geld abgeholt wurde.

Sind alle Anstalten besucht, soll der Karren zurück zum Steuereintreiber gebracht werden. Es kann durch eine Funktion, die einen boolean Wert (false oder true) zurückgibt, getätigigt werden, die den Wert an das Returnstatement im Trigger zurückgibt und so den Trigger beendet oder weiterlaufen lässt.

```

function FollowPilgrim()
    local _beenCount = 0
    for _builidng,_been in pairs(GeldOrte) do
        if GetDistance("Steuerkarren",_builidng)<= 900 and not _been then
            GeldOrte[_builidng] = true
            Sound.PlayUISound(Sounds["OnKlick_PB_Bank1"], 50)
            Message("Geld abgeholt!")
        end
        if _been then
            _beenCount = _beenCount + 1
        end
    end
    if Counter.Tick2("Steuerkarren",10) then
        Logic.GroupGuard(GetID("Steuerkarren"),GetID("Pilgrim"))
    end
    if _beenCount >= 8 then
        return WaitForKarrenToBeBack()
    end
    return false
end

function WaitForKarrenToBeBack()
    if GetDistance("Steuerkarren","geldmeister") <= 800 then
        CreateGeldMeisterGoldBroughtBriefing()
        return true
    end
    return false
end

```

Wenn der Karren auf der Rückfahrt den Steuereintreiber erreicht wird somit ein Briefing ausgelöst und die Anzahl an geholfenen Personen wird nach dem Briefing um eins erhöht.

Als letztes soll dem Einsiedler noch geholfen werden. Seine Einsiedlerhütte wurde von den Banditen niedergebrannt und er braucht ein neues Zuhause. Nach der Interaktion mit ihm muss Pilgrim zuerst einige Arbeiter finden. Im inneren Ring ist ein Architekt lokiert, der Pilgrim ein paar Leibeigene gibt, die in der Zwischenzeit Holz und Lehm besorgen.

```

function MasterBuilderBriefingFinished
    for i = 1,4,1 do
        _serfSpawnPos = GetPosition("Serf"..i)
        Logic.CreateEntity(Entities.PU_Serf,_serfSpawnPos.X,_serfSpawnPos.Y,180,1)
    end
end

```

Die Abfrage, ob das Haus gebaut wurde, fällt hierbei in die Sammlung der Comfort-Funktionen vom Spiel selbst. So können bereits verschiedene Quests auf einfachem Weg abgebildet werden, so auch die Quest ein Gebäude an einer bestimmten Stelle zu bauen.

```
quest.AreaPos = "HermitHut"
quest.AreaSize = 1000
quest.EntityTypes ={{Entities.PB_Residence,1}}
quest.Callback = establishHouseDone

SetupEstablish(quest)

local _hutPos = GetPosition("HermitHut")
_arrowID = Logic.CreateEffect(GGL_Effects.FXTerrainPointer,_hutPos.X,_hutPos.Y,1)
```

Die Leibeigenen können den Lehm von den Lehmhaufen in der Lehmproduktion und das Holz von buchstäblich jedem Baum abbauen. Ob sie dabei den einen oder anderen ästhetischen Aspekt von Bäumen zerstören sei dem Spieler selbst überlassen.

Wenn genug Holz und Lehm zusammengetragen wurde, kann an der angegeben Stelle das Wohnhaus errichtet werden. Sowie das Gebäude fertig ist, wird ein kleines Briefing abgespielt, in dem sich der Einsiedler bedankt. Anschließend werden die Leibeigenen zerstört und auch er gildet als geholfen.

Im weiteren wird jedes Mal, wenn einer Person geholfen wird, eine Funktion aufgerufen, die auch die nötige Anzahl abgleicht und bei genug Personen ein Briefing mit der Prinzessin freischaltet.

```
function IncreaseHelpedPeople()
    Helped = Helped + 1
    GUIQuestTools.UpdateQuestInformationString(Helped .. " / 5")
    if Helped >= 5 then
        CreateNPCPrincessOthersHelped()
        GUIQuestTools.DisableQuestInformation()
    end
end
```

## Kapitel 4 Der Prinzessin helfen:

Nachdem den Personen der Stadt geholfen wurde, gilt es nun noch einer Person zu helfen, nämlich der Prinzessin selbst. Sie möchte, dass ein Karren mit Waren, der auf dem Weg ist, sicher in die Stadt geleitet wird. Damit der Spieler nicht komplett aufgeschmissen ist und direkt zum Anfang rennen muss, damit der Karren gerade noch so erreicht wird, wird der Wagen mit einer Verzögerung erschaffen.

```
GUIQuestTools.ToggleStopWatch(300,1)
StartSimpleJob("ControlTimerKarren")

function ControlTimerKarren()
    if Counter.Tick2("KarrenArrives",300) then
        local _karrenSpawnpos = GetPosition("KarrenSpawn")
        local _karrenid = Logic.CreateEntity(Entities.PU_Travelling_Salesman,
                                            _karrenSpawnpos.X,_karrenSpawnpos.Y,290,4)
        Logic.SetEntityName(_karrenid,"KarrenMitWaren")
        StartSimpleJob("CreateUeberfallBanditen",{1,10})
        StartSimpleJob("CreateUeberfallBanditen",{2,30})
        StartSimpleJob("CreateUeberfallBanditen",{3,30})
        StartSimpleJob("ControllKarren")
        Logic.MoveSettler(GetID("KarrenMitWaren"),
                          GetPosition("Karrenwaypoint1").X,
                          GetPosition("Karrenwaypoint1").Y)
        GUIQuestTools.ToggleStopWatch(0,0)
        return true
    end
    return false
end
```

Das Erscheinen des Karren wird über einen Stoppuhr in der linken oberen Ecke visualisiert mithilfe der GUIQuestTools vom Spiel; die selbigen, die auch eine QuestInformation in der linken oberen Ecke darstellen.

Anschließend wird er Karren kontrolliert gesteuert um einige Wegpunkte abzufahren. Dies wird mithilfe von Distanzchecks zu den jeweiligen Wegpunkten realisiert. Nach 2 Wegpunkten bewegt der Karren sich zum Marktplatz. Sobald er dort erscheint, wird er zerstört und ein neues Briefing wird bei der Prinzessin freigeschaltet

Sollte der Karren auf dem Weg sterben, verliert man das Spiel → Defeat()

```

function ControllKarren()
    if IsDead("KarrenMitWaren") then
        Defeat()
    end
    if IsNear("KarrenMitWaren","Karrenwaypoint1",1000) then
        if Counter.Tick2("KarrenAnWaypoint1",20) then
            Logic.MoveSettler(GetID("KarrenMitWaren"),
                GetPosition("Karrenwaypoint2").X,
                GetPosition("Karrenwaypoint2").Y)
        end
    end
    if IsNear("KarrenMitWaren","Karrenwaypoint2",1000) then
        if Counter.Tick2("KarrenAnWaypoint2",10) then
            Logic.MoveSettler(GetID("KarrenMitWaren"),
                GetPosition("wohnung").X,
                GetPosition("wohnung").Y)
        end
    end
    if IsNear("KarrenMitWaren","wohnung",1000) then
        Logic.DestroyEntity(GetID("KarrenMitWaren"))
        PrrocessHelped()
        return true
    end
    return false
end

```

Gleichzeitig werden Banditentruppen südlich der Straße erschaffen, die den Karren angreifen sollen. Durch eine zeitliche Verzögerung erscheinen die Truppen erst, wenn der Karren bereits auf dem Weg ist. Die Verzögerung wird über einen SimpleJob, sowie eine Zeitangabe und eine Zahl für den Erscheinungsort bewältigt

```

function CreateUeberfallBanditen(_location,_zeit)
    if Counter.Tick2("BanditenspawnLocation".._location) then
        local _spawnpos = GetPosition("Banditen"..location)
        local troopID
        if location ~= 3 then
            troopID=AI.Entity_CreateFormation(3,Entities.PU_BanditLeaderSword1,
                nil,8, _spawnpos.X, _spawnpos.Y, 0, 1, 0, 8
        else
            troopID = AI.Entity_CreateFormation(3, Entities.PU_LeaderBow3,
                nil, 8, _spawnpos.X, _spawnpos.Y, 0, 1, 0, 8)
        end
        local _karrenpos = GetPosition("KarrenMitWaren")
        Logic.GroupAttackMove(troopID, _karrenpos.X, _karrenpos.Y, 305)
        return true
    end
    return false
end

```

Neben Banditentruppen werden dem Spieler auch einige Truppen zur Verfügung gestellt. Damit der Spieler leicht gegen die Banditen gewinnen kann, bekommt er 4 Bastardschwertkämpfer- und 2 Arbellestensbogenschützentruppen bereitgestellt. Sie werden im Briefing erschaffen, da sie im Briefing auch gezeigt werden.

```
AI.Entity_CreateFormation(1,Entities.PU_LeaderSword4, ,nil,  
                         8,GetPosition("innererRing").X,GetPosition("innererRing").Y,0,1,0,8)  
AI.Entity_CreateFormation(1,Entities.PU_LeaderSword4, ,nil,  
                         8,GetPosition("innererRing").X,GetPosition("innererRing").Y,0,1,0,8)  
AI.Entity_CreateFormation(1,Entities.PU_LeaderSword4, ,nil,  
                         8,GetPosition("innererRing").X,GetPosition("innererRing").Y,0,1,0,8)  
AI.Entity_CreateFormation(1,Entities.PU_LeaderSword4, ,nil,  
                         8,GetPosition("innererRing").X,GetPosition("innererRing").Y,0,1,0,8)  
  
AI.Entity_CreateFormation(1,Entities.PU_LeaderBow4, ,nil,  
                         8,GetPosition("Waypoint8").X,GetPosition("Waypoint8").Y,0,1,0,8)  
AI.Entity_CreateFormation(1,Entities.PU_LeaderBow4, ,nil,  
                         8,GetPosition("Waypoint8").X,GetPosition("Waypoint8").Y,0,1,0,8)
```

## Kapitel 5 Die Offenbarung der Prinzessin:

Als Pilgrim der Prinzessin geholfen hat, indem er mit den Truppen von ihr den Karren mit Waren auf seinem Weg in die Stadt beschützt hat, soll er sich ausruhen. In echt wird das Setting auf Nacht umgestellt und ein längeres Briefing wird gestartet, in dem sich Pilgrim und die Prinzessin in der Gartenanlage treffen und dort miteinander reden; sie dabei völlig überrascht von ihm.

Um den Wechsel von Tag auf Nacht zu simulieren wird eine Schwarzblende benutzt. Die Schwarzblende wird mithilfe einiger Widgets im GUI-Baum erstellt, indem deren Farbe auf schwarz und mit jedem Tick die Transparenz weiter abgedunkelt und nach einiger Zeit wieder aufgehellt wird. Zusätzlich kann eine Funktion übergeben werden, die zum Zeitpunkt der vollständigen Schwärze aufgerufen wird.

```
function StartSchwarzBlende(_func)
    XGUIEng.ShowWidget("PauseScreen", 1)
    XGUIEng.ShowWidget("PauseScreenBG", 1)
    XGUIEng.ShowWidget("PauseScreen_Message", 0)
    for i = 1,5,1 do
        XGUIEng.SetMaterialColor("PauseScreenBG", i, 0, 0, 0, 0)
    end
    counter = 1
    assert(TriggerFix,"TriggerFix wird benötigt")
    StartSimpleHiResJob("ControllSchwarzBlende",_func)
end
```

Einige Widgets müssen dafür ausgeschaltet werden, da der Pause-Hintergrund für die Schwarzblende benutzt wird. Texte und weitere Container-Widgets müssen dabei entfernt und später wieder hergestellt werden. Als Funktion, die übergeben wird, dient eine Funktion, die das GFX-Set zu Nacht ändert.

```
function SetNightGFX()
    Display.GfxSetSetSkyBox(9, 0.0, 1.0, "YSkyBox09")
    Display.GfxSetSetRainEffectStatus(9, 0.0, 1.0, 0)
    Display.GfxSetSetSnowStatus(9, 0, 1.0, 0)
    Display.GfxSetSetSnowEffectStatus(9, 0.0, 0.8, 0)
    Display.GfxSetSetFogParams(9, 0.0, 1.0, 1, 52,82,92, 3500,32000)
    Display.GfxSetSetLightParams(9, 0.0, 1.0, 40, -15, -50, 80,90,80, 1,1,1)

    CreateBriefingInGarden()
end
```

```

function ControllSchwarzBlende(_func)
    if not testswitch then
        if counter <= 51 then
            for i = 1,5,1 do
                XGUIEng.SetMaterialColor("PauseScreenBG", i, 0, 0, 0, counter * 5)
            end
            counter = counter + 1
        else
            testswitch = not testswitch
            if _func then
                _func()
            end
        end
    else
        if waitingdone or Counter.Tick2("schwarzbild",30) then
            if not waitingdone then
                waitingdone = true
                counter = counter - 1
            end
            if counter >= 1 then
                for i = 1,5,1 do
                    XGUIEng.SetMaterialColor("PauseScreenBG", i, 0, 0, 0, counter * 5)
                end
                counter = counter - 1
            else
                XGUIEng.ShowWidget("PauseScreen_Message", 1)
                XGUIEng.ShowWidget("PauseScreen", 0)
                XGUIEng.ShowWidget("PauseScreenBG", 1)
                for i = 1,5,1 do
                    XGUIEng.SetMaterialColor("PauseScreenBG", i, 0, 0, 0, 160)
                end
                counter = 1
                waitingdone = nil
                testswitch = nil
                return true
            end
        end
    end
    return false
end

```

Damit das Briefing und die Aktionen in dem Briefing zeitlich passen, müssen sie mit einer gewissen Verzögerung abgespielt werden um nicht bereits während des schwarzen Bildes zu laufen. Außerdem müssen bereits zuvor Position der Kamera und die Möglichkeit sie zu bewegen eingeschränkt werden. Ansonsten findet das Briefing an einer anderen Stelle statt und sieht weder Pilgrim noch die Prinzessin, während diese reden.

Um den Spieler bei dem Briefing mit dabei zu haben wird eine Multiple-Choice-Seite erstellt. Dann kann der Spieler zwischen zwei möglichen Antworten auswählen und eine andere Antwort von der Prinzessin erhalten.

```
page = page + 1
BothInGardenBriefing[page] = {}
BothInGardenBriefing[page].mc          ={}
BothInGardenBriefing[page].mc.title= "Prinzessin"
BothInGardenBriefing[page].mc.text="Ich... Ich..."
BothInGardenBriefing[page].mc.firstText="Ihr braucht nicht antworten, wenn ihr nicht wollt."
BothInGardenBriefing[page].mc.secondText="Vielleicht ist es am Besten, wenn ihr euch mir öffnet."
BothInGardenBriefing[page].mc.firstSelected=4
BothInGardenBriefing[page].mc.secondSelected=5
```

Die firstSelected und secondSlected Attribute dienen dabei Angabe zur nächsten Seite und damit zum nächsten Text der Prinzessin.

Nachdem das komplette Briefing abgelaufen ist, wird die Ausgangssituation wieder hergestellt. Das GFX-Set wird wieder rekonstruiert, Pilgrim und die Prinzessin werden im Garten wieder gelöscht und das Kapitel beginnt mit einem Briefing über einen Angriff auf die Banditen.

## Kapitel 6 Die Banditen vertreiben:

Pilgrim hat mit der Prinzessin im Garten gesprochen und sie möchte nun von ihm, dass er mit einer Armee bestehend aus von ihr bereitgestellten Truppen die Banditen vertreibt.

Die Truppen werden mithilfe einer AI-Funktion erstellt.

```
local _swordSpawnPos = GetPosition("SwordSpawn")
local _cannonSpawnPos = GetPosition("CannonSpawn")
local _bowSpawnPos = GetPosition("BowSpawn")
local _sharpSpawnPos = GetPosition("SharpSpawn")
local _poleSpawnPos = GetPosition("PoleSpawn")
AI.Entity_CreateFormation(1, Entities.PU_LeaderSword4, nil, 8, _swordSpawnPos.X, _swordSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderSword4, nil, 8, _swordSpawnPos.X, _swordSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderSword4, nil, 8, _swordSpawnPos.X, _swordSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderSword4, nil, 8, _swordSpawnPos.X, _swordSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderPoleArm4, nil, 8, _poleSpawnPos.X, _poleSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderPoleArm4, nil, 8, _poleSpawnPos.X, _poleSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderPoleArm4, nil, 8, _poleSpawnPos.X, _poleSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderPoleArm4, nil, 8, _poleSpawnPos.X, _poleSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderBow4, nil, 8, _bowSpawnPos.X, _bowSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderBow4, nil, 8, _bowSpawnPos.X, _bowSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderBow4, nil, 8, _bowSpawnPos.X, _bowSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderBow4, nil, 8, _bowSpawnPos.X, _bowSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderRifle1, nil, 8, _sharpSpawnPos.X, _sharpSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderRifle1, nil, 8, _sharpSpawnPos.X, _sharpSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PU_LeaderRifle1, nil, 8, _sharpSpawnPos.X, _sharpSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon3, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon3, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon4, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon4, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon4, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
AI.Entity_CreateFormation(1, Entities.PV_Cannon4, nil, 8, _cannonSpawnPos.X, _cannonSpawnPos.Y, 0, 1, 3, 8)
```

Im gleichen Zug werden die Aussichtstürme im Banditenlager aufgewertet zu Balistatürmen.

```
function UpgradeBanditTowers()
    Logic.DEBUG_UpgradeBuilding(GetID("BanditenDef1"))
    Logic.DEBUG_UpgradeBuilding(GetID("BanditenDef2"))
end
```

Die Banditentruppen werden nicht verändert. Ihr Verhalten ist bereits in Kapitel 1 gesetzt und wird dementsprechend vorerst nicht angepasst. Ebenso werden keine weiteren Truppen erstellt, um es dem Spieler nicht noch schwerer zu machen, da es sich um eine Karte handeln soll, die von jedem Spieler bewältigbar ist, egal ob Anfänger oder Profi.

Die Bedingung für den Sieg über die Banditen wird über einen SimpleJob abgefragt, der jede Sekunde nach Spielerentitäten des Spielers abfragt. Die Funktion `Logic.GetPlayerEntities` gibt dabei die Anzahl und ggf. eine maximale Anzahl von 16 EntityIDs zurück, wodurch sich feststellen lässt, ob ein gewisser Spieler zwischen 0 und 16 Entitäten besitzt.

Sobald der Spieler der Banditen keine Entitäten mehr auf der Karte hat, gilt der Angriff als erfolgreich und ein Briefing startet, welches das 7. Kapitel einleitet.

```
function IsPlayer3Dead()
    local _P3Entities = {Logic.GetPlayerEntities(3,nil,16)}
    if _P3Entities[1] <= 0 then
        CreateBriefingBanditsGone()
        return true
    end
    return false
end
```

Damit Skriptentitäten nicht für Spieler 3 zählen, werden sie auf einen anderen Spieler oder am Anfang vom Skript alle auf Spieler 0 gesetzt.

## Kapitel 7 Ende gut, alles gut:

Als Pilgrim die Banditen besiegt hat, möchte er ein Fest veranstalten, zu dem alle aus der Region eingeladen sind. Dafür werden aus jedem Bezirk und jedem Gebiet jeweils eine Person herangezogen, die dann auf den Marktplatz geht.

--NPC Tabellen sind davor für jeden NPC erstellt worden.

```
CreateNPC(HermitNPCEnd)
CreateNPC(LeonardoNPCEnd)
CreateNPC(SawmillNPCEnd)
CreateNPC(MonkNPCEnd)
CreateNPC(SmelterNPCEnd)
CreateNPC(GeldmeisterNPCEnd)
CreateNPC(CaptainNPCEnd)
CreateNPC(SteinmeisterNPCEnd)
CreateNPC(MasterBuilderNPCEnd)
CreateNPC(GardenDudeNPCEnd)
```

Damit bei jeder Interaktion ein anderer Text erscheint, wurde eine Index-Value-Tabelle erstellt. Bei jeder Interaktion bzw. jedem Briefing wird ein Zähler um eins erhöht und jedes Mal wird auf die Tabelle über den Zähler als Index zugegriffen, sodass bei zehn Einträgen sich mit jeder Interaktion der nächste Eintrag geholt wird.

--Aus Gründen der Länge wurde der Text mit „...“ ersetzt

```
NPCGoToMarketBriefingAnswerTable = {}
NPCGoToMarketBriefingAnswerTable[1] = "..."
NPCGoToMarketBriefingAnswerTable[2] = "..."
NPCGoToMarketBriefingAnswerTable[3] = "..."
NPCGoToMarketBriefingAnswerTable[4] = "..."
NPCGoToMarketBriefingAnswerTable[5] = "..."
NPCGoToMarketBriefingAnswerTable[6] = "..."
NPCGoToMarketBriefingAnswerTable[7] = "..."
NPCGoToMarketBriefingAnswerTable[8] = "..."
NPCGoToMarketBriefingAnswerTable[9] = "..."
NPCGoToMarketBriefingAnswerTable[10] = "..."
NPCGoToMarketBriefingAnswerTableCounter = 0
```

Sobald in der Finished-Funktion des Briefings der Zähler den Wert zehn erreicht hat, wird ein SimpleJob gestartet, der überprüft, ob sich Pilgrim auf dem Marktplatz befindet.

```

function WaitForPilgrimAtMarket()
    if GetDistance("Pilgrim", "Fest3", 1000) then
        VictoryCounter = 0
        CreateEndBriefing()
        StartSimpleJob("MarktplatzFirework")
        return true
    end
    return false
end

```

Wenn Pilgrim auf dem Marktplatz angekommen ist, wird ein weiterer SimpleJob und ein letztes Mal ein Briefing gestartet. Das Briefing ist der Abschluss der Karte. Während eines Briefings läuft das Triggersystem weiter. Das heißt, dass im letzten SimpleJob Feuerwerks-Effekte platziert werden um dem Fest einen gebührenden Abschluss zu geben.

```

function MarktplatzFirework()
    VictoryCounter = VictoryCounter + 1
    if VictoryCounter == 3 then
        local _firework1Pos = GetPosition("Fest3")
        Logic.CreateEffect(GGL_Effects.FXYukiFireworksJoy , _firework1Pos.X, _firework1Pos.Y,0)
    end
    if VictoryCounter == 5 then
        local _firework2Pos = GetPosition("Waypoint6")
        Logic.CreateEffect(GGL_Effects.FXYukiFireworksJoy , _firework2Pos.X, _firework2Pos.Y,0)
        return true
    end
    return false
end

```

Am Ende des Briefings wird die Victory-Funktion aufgerufen, wodurch die Karte als gewonnen gilt und man das Spiel mit einer Siegesmeldung verlassen kann.

## Testphase:

Die Testphase wird in zwei Teile unterteilt. Zuerst wird die Karte auf ihr technischen Begebenheiten überprüft. Darunter fällt die Funktionsfähigkeit von Routinen im Script, aber auch erwartete Ausgänge bei gewissen Aktionen, wie der Banditenarmee. Nachdem die Karte auf ihre Funktionalitäten geprüft wurde, wird die Karte auf ihre Spielweise auf die Probe gesetzt. Hierfür gibt es keine wirklichen Kriterien, da hierbei das Spielerlebnis im Vordergrund steht. Eine gute Herangehensweise ist es einige Personen auszusuchen und diese die Karte von Anfang bis Ende spielen zu lassen. Es ist wichtig, dass sie nicht gewisse Vorgehensweisen vom Kartenersteller zu testen haben, sondern sie auf eigenen Wegen die Karte erkunden.

## Aufsetzen der Testumgebung:

Es gibt 2 Wege, wie man die LuaDebugger-Umgebung hervorrufen kann.

- Es wird eine Verknüpfung der Executable-Datei angelegt und beim Dateipfad -debugscript ergänzt. Sobald ein Luafehler auftritt, wird das Spiel pausiert und eine Fehlermeldung mit dem Fehler erscheint. Wirklich Debuggen mit einer Konsole und Stacktrace geht hiermit allerdings nicht.
- Mithilfe des S5Updaters kann, sofern der einfache Modus deaktiviert ist, man den LuaDebugger installieren lassen und ihn bei Spielstart mit starten lassen.

Alternativ kann man den LuaDebugger auch manuell installieren: [S5 LuaDebugger](#)

Er bietet neben der Ansicht welche Skripte geladen sind, auch eine Ansicht auf ihren Inhalt, sowie einen Output und eine Eingabe am unteren Rand. Zusätzlich wird bei einem auftretenden Fehler das Spiel pausiert und der Fehler wird mit samt Stacktrace angezeigt und kann nachverfolgt, sowie auch Variablen im Kontext abgefragt und geändert werden.

- Lua Error -		
[string "Data\Script\MapTools\Comfort.lua"]:1682: assertion failed!		
Function in Call Stack	Source	Line
Game Engine (direct call)	unavailable	0
global CreateNPC()	Data\Script\MapTools\Comfort.lua	1682
global CreateNPCDogAway()	maps\user\contestmap\Chapter 3\npcbriefings_ch3.lua	14
global StartQuestHelpOthers()	maps\user\contestmap\Chapter 3\helpothers_ch3.lua	10

Das Einzige, was damit nicht debugbar ist, sind Fehler auf der C-Ebene. Solche entstehen, indem invalide Werte an C-Funktionen übergeben werden. So führt z.B. Entities.PB\_Haus5 innerhalb der Funktion Logic.CreateEntity zu einem Fehler, woraufhin das Spiel crasht. Auf solche Sachen sind dementsprechend zu achten.

## **Technische Testphase:**

In der technischen Testphase fallen vor allem Fehler auf, wie fehlender Code um Briefings zu starten oder Rechtschreibfehler bei Funktionen, wie GetPosition → GetPositoin.

Neben solchen Fehlern ist vor allem einer aufgetreten beim Erscheinen der Banditen, die den Karren angreifen sollen. An den Trigger wurde eine Tabelle übergeben und die benutze Funktion wurde mit 2 Variablen definiert. Allerdings werden durch die Übergabe der Tabelle nicht zwei Variablen übergeben, sondern eine, sodass die Funktion in der Definition auch nur einen Parameter haben soll. Anschließend wird innerhalb der Funktion auf die Tabelle zugegriffen.

Durch die falsche Handhabung der Variablen kommt es zur versuchten Verarbeitung mit Nil/Null-Werten und in Folge dessen zu Lua-Fehlern.

Ansonsten sind überwiegend Rechtschreibfehler und syntaktische Fehler aufgetreten, welche sich über Erweiterungen mit Lua-Interpretern vermeiden lassen.

Das Ziel der technischen Testphase ist, dass das Spiel von Anfang bis Ende den gewünschten Lauf einnimmt und die Ereignisse, wie erwartet, auftreten und ablaufen.

Nach einem vollständigen Durchlauf und der Überprüfung wird die Karte für weitere, ausgesuchte Testpersonen zur Verfügung gestellt. Damit ist die technische Testphase vorbei.

## **Spielerische Testphase:**

In der spielerischen Testphase geht es darum, wie die Spieler ihren Weg durch die Karte finden. Dabei werden oftmals auch Sachen gemacht, die der Kartenersteller nicht berücksichtigt hat, streng nach dem Motto „Die menschliche Dummheit ist unendlich.“  
\*Zitat von Albert Einstein\*

Bei einigen Läufen kamen Spieler auf die Idee, statt dem Wohnhaus für den Einsiedler, Bauernhöfe und Universitäten zu bauen. Auch wenn diese nicht bewohnbar waren aufgrund eines fehlenden Dorfzentrums, sind diese Gebäude trotzdem nicht gewollt. Eine Lösung dafür ist das Bauen dieser Gebäude von Anfang an zu sperren.

```
ForbidTechnology(Technologies.B_Farm)
ForbidTechnology(Technologies.B_Village)
ForbidTechnology(Technologies.B_University)
ForbidTechnology(Technologies.B_Claymine)
```

Ein weiterer Gedanke, der bei manchen Spielern besteht ist: „Wie breche ich die Karte/das Level so hart wie möglich?“. In anderen Worten bedeutet das was für Sachen sie außerhalb von den eigentlichen Tätigkeiten machen können. Sie neigen dazu die Grenzen des Machbaren auszureißen. So auch die Banditen.

Die Banditen haben ein einziges Erscheinen in ihrem Camp, sodass theoretisch ein vorläufiges Töten von ihnen dafür sorgt, dass man das Lager ohne Widerstand zerstören kann. Dadurch wird das sechste Kapitel, in dem das Banditencamp zerstört wird, übersprungen. Ein Fix hierfür ist ein sogenannter Spawngenerator. Die UnlimitedArmy kann einen Spawngenerator angehängt bekommen, der dann, je nach Bedingung Truppen erscheinen lassen und sie der Armee hinzufügen kann. Jener wird ab dem sechsten Kapitel dann deaktiviert. Dadurch bleiben bis zum sechsten Kapitel dauerhaft Truppen im Banditenlager und der Spieler kann dieses nicht vorläufig vernichten.

```
BanditenSpawner = UnlimitedArmySpawnGenerator:New(Bandits,
    Position = GetPosition("BanditCampSpawn"),
    ArmySize = 6,
    SpawnCounter = 5,
    SpawnLeaders = 6,
    LeaderDesc = {
        {LeaderType = Entities.CU_BanditLeaderSword1, SoldierNum = 8, SpawnNum = 1,
         Looped = true, Experience = 0},
        {LeaderType = Entities.PU_LeaderBow3, SoldierNum = 8, SpawnNum = 1,
         Looped = true, Experience = 0}
    },
    RefillSoldiers = true
}

-- Zum Start von Kapitel 6
BanditenSpawner:Remove()
```

Gerade die Aufgabe die Steine zu sprengen in Kapitel 3 geben dem Spieler die Möglichkeit die Banditen leicht auszulöschen, da man dann sowohl die Bomben, als auch die stationären 4-Schuss-Kanonen von Pilgrim nutzen kann.

Nebenbei ist noch aufgefallen, dass manche Arbeiter im mittleren Ring trotzdem Feuerstellen legen. Dabei handelt es sich um den Fall, dass es zu wenig Essensplätze gibt. Um dies zu verhindern wird ein Trick angewandt. In die selben Essenshäuser werden weitere Essenshäuser mit dem selben Blocking und kleiner gesetzt. Die Modelgröße dieser zusätzlichen Gebäude wird auf 0.01 gesetzt, sodass sie kaum bis gar nicht vom Spieler auswählbar sind. Die Eingänge überschneiden sich mit der Aufwertung in der sie stehen, sodass alle Arbeiter visuell trotzdem in das selbe Gebäude reingehen.

Zusätzlich wird die maximale Arbeiteranzahl in einer Mühle und einem Gutshof von 2 bzw. 3 auf 1 gesetzt, sodass das Gebäude trotzdem noch die anderen Arbeiter mit essen versorgen kann, allerdings keine weiteren Plätze wegnimmt und so die anderen Arbeiter dort essen können. Dadurch verschwinden auch die Feuerstellen auf den Straßen.

```
function GetAllEntitiesOfType(_player, _type)
    local n, first = Logic.GetPlayerEntities(_player, _type, 1);
    if n > 0 then
        local entity = first;
        repeat
            --here
            Logic.SetCurrentMaxNumWorkersInBuilding(entity, 1)
            entity = Logic.GetNextEntityOfType(entity);
        until entity == first;
    end;
end;

--In der FMA
GetAllEntitiesOfType(2, Entities.PB_Farm2)
GetAllEntitiesOfType(2, Entities.PB_Farm3)
```

## Karten-Info anpassen:

Mit der Beendigung der Testphase fehlt noch eine einzige Sache. Im Hauptmenü steht noch immer der falsche Titel, sowie eine falsche Beschreibung und ein falsche Kartenvorschau. Um dies zu berichtigen wird nun die info.xml editiert:

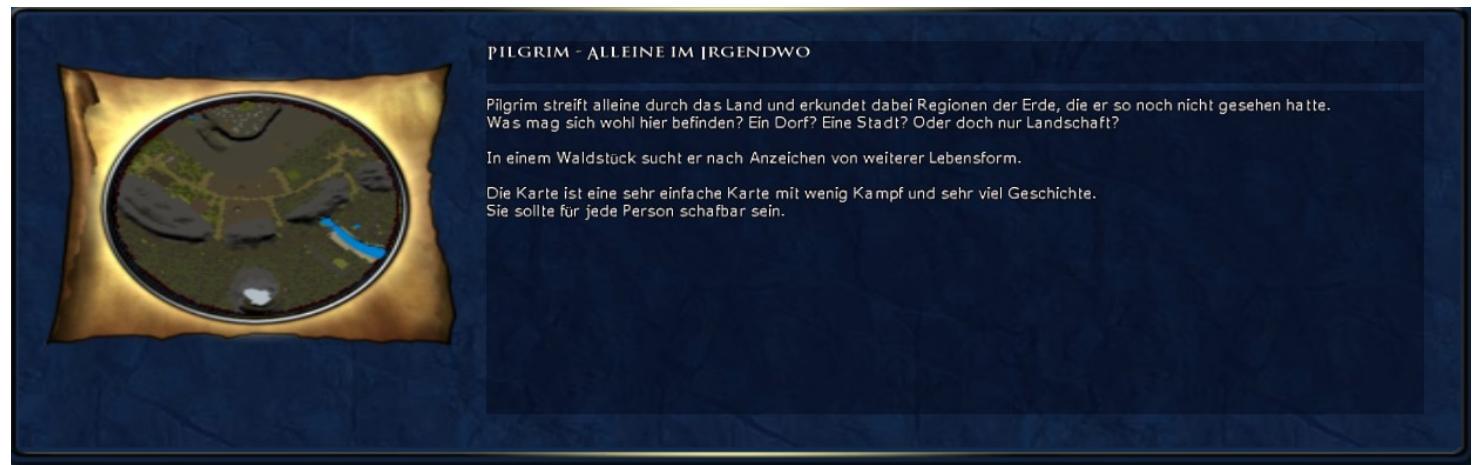
- Die Werte für NameKey und DescKey werden entfernt.
- Der Wert für Name wird auf einen passenden Titel gesetzt. In diesem Fall auf „Pilgrim – Alleine im Irgendwo“.
- Der Wert für Desc wird auf eine passende Beschreibung gesetzt:

„Pilgrim streift alleine durch das Land und erkundet dabei Regionen der Erde, die er so noch nicht gesehen hatte. @cr Was mag sich wohl hier befinden? Ein Dorf? Eine Stadt? Oder doch nur Landschaft?“

@cr In einem Waldstück sucht er nach Anzeichen von weiterer Lebensform. @cr @cr Die Karte ist eine sehr einfache Karte mit wenig Kampf und sehr viel Geschichte. @cr Sie sollte für jede Person schafbar sein.“

- Der Wert für MiniMapTextureName wird auf den Pfad zum Bild im Order gesetzt: maps\user\Pilgrim – Alleine im Irgendwo\mappic , wobei mappic das Bild der Kartenvorschau ist.

Nachdem diese Infos angepasst wurden, wird die Karte folgendermaßen bei der Kartenauswahl angezeigt.



## **Spielen der Karte:**

Damit die Karte gespielt werden kann, muss diese an einem öffentlichen Ort hochgeladen werden. Für solche Dateien bieten sich Internetseiten, wie

- [siedelwood-3000.de](http://siedelwood-3000.de)
- [siedler-maps.de](http://siedler-maps.de)

an. Sie hosten eine Vielzahl an Karten für unterschiedliche Siedler-Versionen und bieten neben Einzelspieler-Karten auch Kooperations-Karten, sowie Kampf und Geschichtskarten. Wichtig ist, dass die Siedler-Maps (SMS, bekannt als Siedler-Maps-Source) nur s5x Dateien annimmt. Diese entstehen, wenn Karten als Archiv gespeichert bzw. mithilfe des bba-Tools gepackt werden.

In der Variante in dieser Dokumentation wird aber die Orderform behandelt, weshalb die SMS nicht zur Wahl steht (auch wenn beim Upload steht, dass sie Zip-Dateien nimmt).

Auf Siedelwood kann die Karte als Zip hochgeladen werden.

Damit die Karte im Spiel nachher von anderen Personen gespielt werden kann, müssen diese zuerst die Karte herunterladen. Nachdem sie die Karte herunter geladen haben, platzieren sie die Karte (in diesem Fall der Ordner) in da User-Verzeichnis der jeweiligen Spielversion( <base>\extra1\extra2>\shrlmaps\user\ ). Dabei steht base für das Basisspiel, für das es fast keine Benutzerkarten gibt, extra1 für Nebelreich und extra2 für Legende. Sie sollte entweder als s5x-Archiv oder als Ordner drinnen sein. Anschließend kann die Person die Karte im Spiel selbst spielen.

Die Karte kann unter folgendem Link heruntergeladen werden:

<https://siedelwood-3000.de/portfolio-item/pilgrim-alleine-im-irgendwo/>

## **Exkurs Erweitertes Mapping:**

Neben solchen Karten, wie diese hier, gibt es auch Karten mit weit aus mehr Inhalt und komplexeren Mechaniken und Stielen. Einzelne von diesen werden am Rande mal erwähnt.

## **Eigene Models/Entities/Animationen:**

Die Models von Siedler sind als DFF-Dateien hinterlegt. Es gibt von der S5Community eine erstellte Erweiterung für Blender, mit der man diese Modelle editieren und erstellen kann, sodass man sie danach später dem Spiel zur Laufzeit hinzufügen kann. Dies setzt natürlich voraus, dass man modellieren kann. Zusätzlich sollte auch die Fähigkeit bestehen Texturen zu erstellen, sowie ein Verständnis für den Aufbau der Dateien, womit diese initialisiert werden, vorhanden sein.

## **Eigenes Graphical User Interface:**

Das Graphical User Interface ist in der Regel eine XML-Datei, welche geladen wird. Mithilfe des [S5GUIEditor](#) kann neben dem Nachschauen vom Standardbaum auch ein eigener erzeugt werden, der dann anschließend in das Spiel geladen wird. Das ist praktisch, wenn man weitere Fenster neben den bereits Existierenden möchte, wie weitere Balken, Knöpfe und Textfelder.

## **Eigene Effekte:**

Die Effekte bestehen jeweils aus einem Vertex-Shader und einem Pixel-Shader. Der Vertex-Shader ist hierbei für die Anordnung der Verticies verantwortlich, womit sich auch kleine Animationen, wie ein Schaukeln eines Schiffes auf dem Wasser darstellen lassen. Der Pixel-Shader ist für die Platzierung der Textur auf dem Model verantwortlich. Durch ihn bekommt das Model Farbe und kann auch einzelne Effekte erzeugen, wie z.B. Wellen im Wasser.

## **Modloader:**

Um alle diese Sachen laden zu können, wird ein so genannter Modloader benötigt. Momentan gibt es für das Spiel 2 Modloader:

- [CppLogic](#) bzw. der alte [S5Hook](#) (CppLogic kann man über den S5Updater für die Gold Edition installieren)
- Kimichuras Modloader via MPUpdater

Sie bieten beide unterschiedliche Möglichkeiten neuen Inhalt in das Spiel einzubinden, wobei die Benutzung beider Modloader unterschiedlich ist. Für beide sind auch Dokumentationen vorhanden: [CppLogic Dokumentation](#) und [Kimichuras Modloader Dokumentation](#).

## **Eigene Mechaniken:**

Neben externen einzubindenden Inhalten können auch neue Mechaniken entwickelt werden. So wurden z.B. Giftreiter entwickelt, die eine Rauchwolke als Effekt auf dem Boden platzieren und jeder, der in diese Wolke tritt, bekommt Schaden über Zeit angerechnet; oder Bombenreiter, welche alle paar Sekunden eine Chance haben, eine Bombe fallen zu lassen. Auch bekannte Mechaniken von Kartenerstellern, sind Mörserfeuer, welche durch Effekte am Boden, eine Explosion und anschließendem Schaden simuliert werden; alles auf Lua-Ebene.

Auch wurde bereits übertragendes Feuer und relativer Realismus mit sterbender Bevölkerung bei fehlendem Nahrungs- und Schlafzugang erschaffen, wodurch sich Karten anders spielen als sonstige Karten und zusätzliche Schwierigkeiten herbei geführt wurden.

Der Kreativität ist dabei fast keine Grenzen gesetzt, soweit es die Basisfunktionen des Spiels zulassen. Die technische Grenzen, sowie die Performance sollten dabei bewahrt werden um dem Spieler keine negativen Einflüsse zu vermitteln.

## Zusammenfassung:

Game-Engineering bzw. Level-Engineering ist eine Sache für sich, da es für jedes Spiel anders ist. In diesem Beispiel hier wurde ein Spiel herangezogen, was an sich zwar in das Genre der Echtzeit-Strategie-Spiele einordnet, aber durch seine Möglichkeiten an Briefings und geschichtlichen Interaktionen eine Ebene bietet, auf der aus einem reinen Strategiespiel plötzlich eine Vielfalt an Facetten wird.

Auch wenn die Installation von Entwicklungsumgebungen oftmals mehr Arbeit erscheint, als die eigentliche Entwicklung eines Levels, so hat das Level eine andere Herangehensweise.

Karten bzw Level sollten einen roten Faden beinhalten. Der rote Faden kann als Thema hierbei unterschiedlich sein. Mal beinhaltet er eine Geschichte, die es gilt nachzuspielen; mal eine oder mehrere Kampfhandlungen; mal eine Nacherzählung. Die Ausarbeitung ist hierbei von Bedeutung.

Um den roten Faden herum geht es schließlich an das Design der Karte und damit auch den Anfangszustand der Karte. Hierbei spielt nicht die Größe der Karte eine Rolle, sondern der Inhalt der Karte in Relation zur Größe. Es kann die größte Karte herangezogen werden, die allerdings wenig Inhalt bietet und dadurch mehr ein Lauf-Simulator anstatt einer guten Geschichte ist. Auf der anderen Seite kann eine Karte auch zu klein und dafür mit Inhalt vollgestopft sein, dass es bereits unübersichtlich und überfüllt ist. Basierend darauf wird die Karte dementsprechend der Idee dann designt.

Nach abgeschlossener Ausarbeitung der Karte wird der Verlauf der Karte angegangen. Hierbei wird in der jeweiligen Skriptsprache eine Art Checkpoint-System eingeführt. Nacheinander werden Trigger gesetzt, die nach jeder Aktion weitere Funktionen ausführen und so weitere Vorgänge in Kraft setzen; seien es Briefings, Checks auf Bedingungen, Trigger aus einem Triggersystem, Gebietsabfragen. Dies zieht sich durch die gesamte Karte und eröffnet verschiedene Möglichkeiten von Quests und Aufgaben auf der gesamten Karte. Neben der Gestaltung der Karte wird hierfür ein Großteil der Zeit investiert.

Sobald die Karte design- und skripttechnisch abgeschlossen sind, geht es an die Tests. Bei den Tests ist es wichtig, dass sowohl technische Tests auf ihre Funktionalität, als auch spielerische Tests auf ihre Spielart getätigten werden. Eine Karte kann komplett funktionieren, aber den Spieler nicht ansprechen und ihm nicht gefallen; ebenso das Gegenteil.

Mit der Beendigung der Tests fallen noch letzte Kleinigkeiten an, wie das Setzen der richtigen Informationen, wie Titel und Beschreibung, an. Sie müssen passend gesetzt werden um den Spieler anzusprechen und sollten keine Platzhalter enthalten.

Anschließend wird in einem finalen Test nochmal über das Erscheinungsbild geschaut und die Karte kann schließlich veröffentlicht werden.

## Quellen/Verweise:

- S5CommunityLib, Bibliothek mit Lua-Funktionen von der S5 MP Community, online, <https://github.com/mcb5637/s5LuaReference>
- S5Updater, Updater und Patcher für „Die Siedler – Das Erbe der Könige“ bezüglich Online-Karten und Funktionalitäten, online, <https://github.com/mcb5637/S5Updater>
- Kimichuras Discord für den MPUUpdater, online, <https://discord.gg/b28BsKz>
- Download-Link für den Patch 1.06, online, <https://www.siedler-maps.de/downloads.php?action=download&downloadid=20>
- S5LuaReference, online, <https://github.com/mcb5637/s5LuaReference>
- S5GUIEditor, online, <https://github.com/mcb5637/S5GUIEditor>
- Siedler-maps, Mögliche Internetseite für Karten, online, <https://www.siedler-maps.de/news.php>
- siedelwood, Mögliche Internetseite für Karten, online, <https://siedelwood-3000.de/>
- CppLogic, Der neue Hook und Ersatz für den alten S5Hook, online, <https://github.com/mcb5637/S5BinkHook>
- S5Hook, der alte Hook und Vorgänger von CppLogic, online, <https://github.com/mcb5637/S5Hook>
- CppLogic Dokumenation, Die Dokumentation bezüglich CppLogic, online, <https://github.com/mcb5637/S5BinkHook/tree/master/S5CppLogic/doc>
- Kimichuras Modloader Dokumentation, Die Dokumentation bezüglich seines MPUUpdaters mit den Funktionen innerhalb des Spiels, online, <https://github.com/mcb5637/s5LuaReference/tree/master/SpExtended>
- Viele individuelle Informationen von unterschiedlichen Personen sowie Diskussionen, online, <https://discord.gg/utBbZzh>