Im dritten Teil unseres Tutorials zum Karten Editor von DIE SIEDLER®: Das Erbe der Könige™ möchten wir euch einen Leitfaden zum Scripten mit Lua an die Hand geben, mit dem es euch auch gelingt, komplizierte Karten, sowie Quests zu erstellen. Dieses Tutorial geht nicht näher auf die Funktionsweise der Programmiersprache Lua ein, da dies den Rahmen sprengen würde, sondern zeigt nur die Verwendung entsprechender Code Zeilen. Zu diesem Thema gibt es im Internet kostenloses Material z.B. unter: www.lua.org

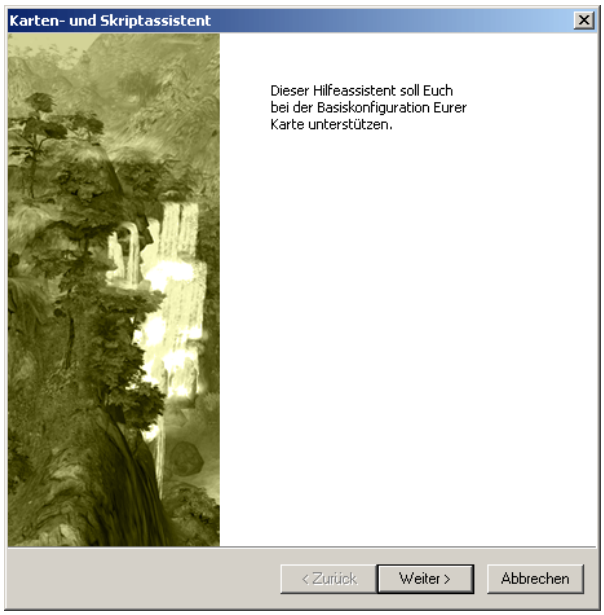
Werft auch einen Blick auf die ScriptingReference (zu finden in eurem DIE SIEDLER: DEdK Verzeichnis unter extra1\manual\ScriptingReference.html), um die in diesem Dokument verwendeten Skriptbefehle nachzuschlagen.

1 Eine einfache Mission erstellen

1.1 Aller Anfang ist schwer

Auch eine scheinbar simple Mission will gründlich vorbereitet sein. Welche Gebäude und Nichtspielercharaktere (NSCs) werden benötigt? Welche Aktionen sollen wann und unter welchen Voraussetzungen durchgeführt werden?

Am Beispiel einer einfachen Mission werdet ihr einige grundlegende Funktionen der Skriptsprache kennen lernen. Startet zuerst den Editor und erstellt ein neues Projekt. Klickt dazu auf die entsprechende Schaltfläche „Neu“ (das weiße Blatt Papier oben links).

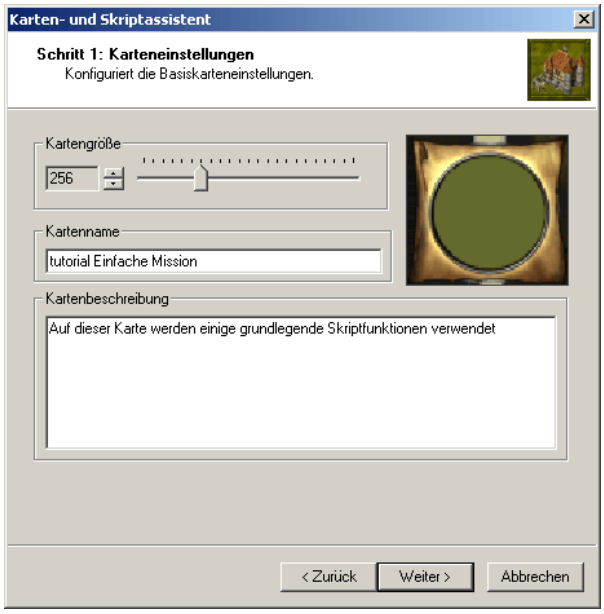


Nun startet der Karten- und Skriptassistent, der ein Basisskript aufgrund einiger Parameter erstellt.

Dazu sei gesagt, dass dieses Basisskript in seiner Funktionalität eingeschränkt ist. Wie es der Name vermuten lässt, enthält das Skript nur die nötigsten Funktionen. Trotzdem wird nun kurz auf den Karten- und Skriptassistenten eingegangen, aber am Ende werdet ihr ein eigenes Skript erstellen, das euch die gesamte Skriptsprache verdeutlichen wird.

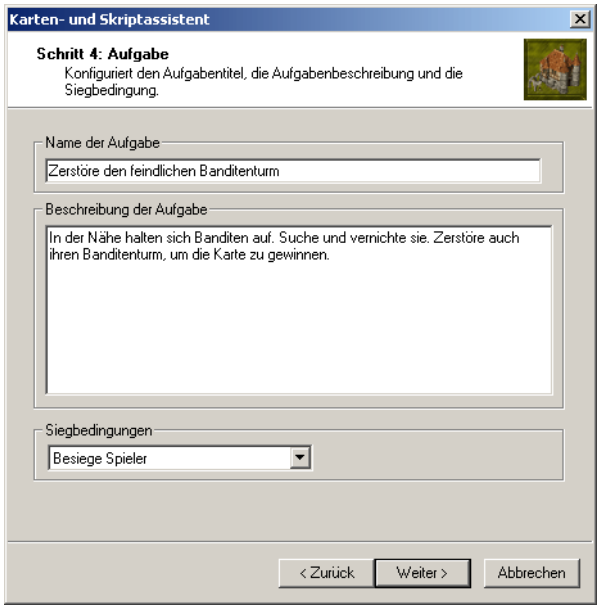
Zurück zum Skriptassistenten:

Drückt auf „Weiter“ um zu Schritt 1 zu gelangen, wo ihr die Kartengröße festlegt. 256 ist für dieses Beispiel mehr als genug. Benennt die Karte „Tutorial Einfache Mission“. In der Kartenbeschreibung könnt ihr z.B. eintragen: „Auf dieser Karte werden einige grundlegende Skriptfunktionen verwendet“.



Nun auf „Weiter“ klicken und den Schritt 2 ignorieren, weil diese Karte eine Einzelspieler-Karte sein wird. Also gleich wieder auf „Weiter“ klicken“.

Im Schritt 3 werden die Rohstoffe festgelegt, die der Spieler zu Beginn erhält.
In dieser Mission wird das nicht von Bedeutung sein, weil keine Siedlung aufgebaut und Truppen gekauft werden sollen. Der Spieler wird mit den Ressourcen also nicht viel anfangen können. Ansonsten hat sich ein Wert von **1000 für jede Ressource** in vielen Karten als Standardwert bewährt.



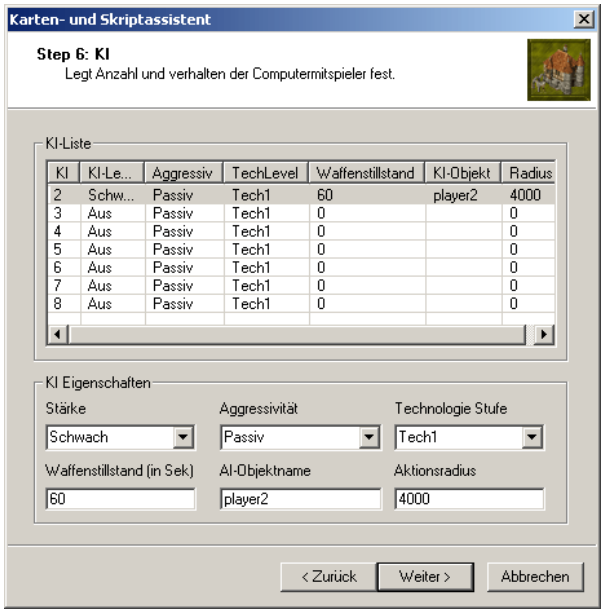
Klicke auf „Weiter“, um im Schritt 4 die Aufgabe zu definieren. Trage hier als Name der Aufgabe: „Zerstöre den feindlichen Banditenturm“ ein und als Beschreibung: „In der Nähe halten sich Banditen auf. Suche und vernichte sie. Zerstöre auch ihren Banditenturm, um die Mission zu gewinnen.“ Die Siegbedingungen werden auf „Besiege Spieler“ eingestellt.

Mit einem Klick auf „Weiter“ werdet ihr nun gefragt, welcher Spieler besiegt werden soll, um die Karte zu gewinnen. Hier muss der Wert „2“ stehen (Voreinstellung).

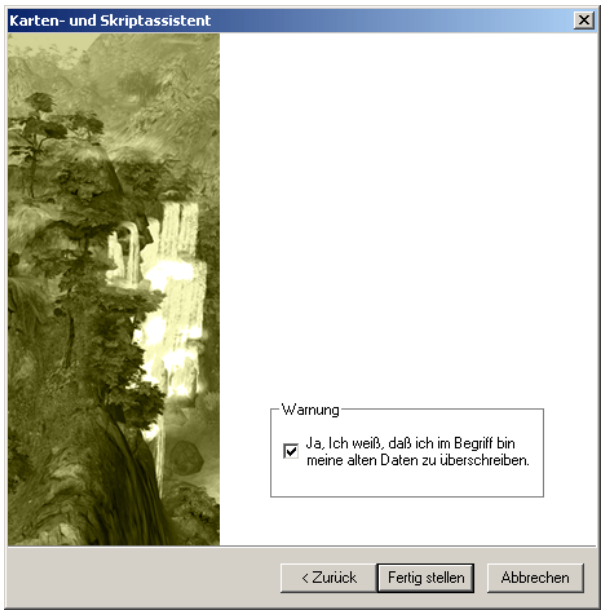
Der nächste Schritt fragt euch nun noch nach einigen Werten des gegnerischen KI Spielers (KI = Künstliche Intelligenz).

Klickt in der KI-Liste auf die erste Zeile (KI 2) und tragt bei den KI-Eigenschaften folgende Werte ein:

Stärke:	Schwach
Aggressivität:	Passiv (der KI Spieler greift erst an, nachdem er angegriffen wird)
Technologie Stufe:	Tech1 (Der KI Spieler startet mit den Starttechnologien)
Waffenstillstand (in Sek):	60 (erst nach 60 Sekunden greift der KI Spieler an)
AI-Objektname:	player2 (Die KI hat diesen Namen im Diplomatiefenster)
Aktionsradius (in cm):	4000 (innerhalb von diesem Radius sucht die KI nach Gegnern)



Auf „Weiter“ geklickt erscheint nun das gesamte Basisskript. Das soll jetzt nicht weiter interessieren, also gleich noch mal auf „Weiter“ klicken und ihr werdet gefragt, ob ihr wirklich eine neue Karte mit den eben gewählten Werten erschaffen wollt. Diese Aktion wird eine bereits offene Karte ersetzen. Wenn ihr also noch eine Karte im Viewport geöffnet habt, speichert diese vor Erstellen einer Karte stets ab. Setzt ein Häkchen bei der Warnung und klickt auf „Fertig stellen“.



Nun liegt die noch nackte Karte vor euch, aber schon bald werdet ihr eure erste Mission erstellt haben.

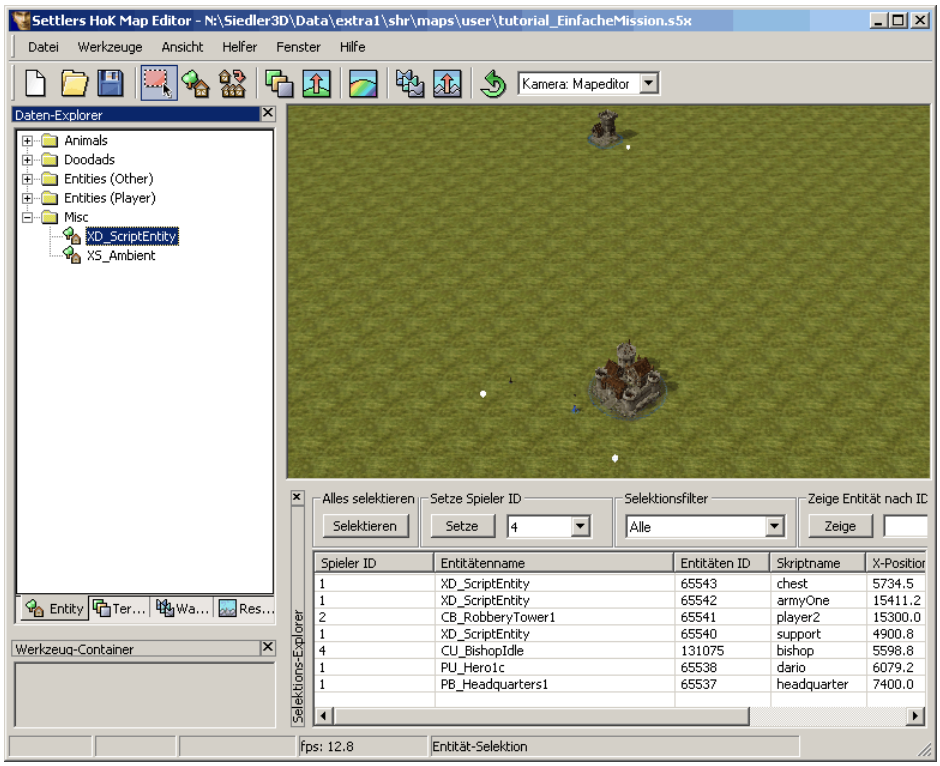
Es soll eine sehr einfache Mission werden, deshalb wird das Missionsziel sein, einen Banditenturm zu zerstören. Dieser soll bewacht werden von einigen Truppen Banditen, die sich aber passiv verhalten und den Spieler erst dann angreifen, wenn sich dieser ihnen nähert.

Der Spieler bekommt zur Bewältigung dieser Aufgabe den Helden Dario zur Seite gestellt, eine Burg als Haupthaus und einen NSC, den er ansprechen kann und der ihm daraufhin ein paar Soldaten zur Verfügung stellt. Natürlich könnt ihr die Karte editieren und ausschmücken, für unseren Zweck ist dies aber nicht notwendig.

Platziert zuerst Darios Burg (PB_Headquarters1) im Süden der Karte. Stellt nun neben die Burg den Helden Dario (PU_Hero1c). Nicht weit von Dario entfernt soll nun ein NSC stehen, der für Dario die dringend benötigten Truppen bereithält.
Für diesen Zweck nehmen wir einen Bischof (CU_BishopIdle). Die Truppen brauchen nun noch eine eindeutige Position, an der sie auf der Karte erscheinen sollen. Nehmt dafür das Objekt „XD_ScriptEntity“, welches ihr im „Misc“-Verzeichnis findet. Die „XD_ScriptEntity“ ist ein Platzhalter, der im Spiel nicht und im Editor als weißer Quader zu sehen ist und wichtige Dienste bei den verschiedensten Missionen leistet. In diesem Fall soll an der Position dieses Objekts die Verstärkung für Dario erscheinen.
Jetzt braucht ihr noch einen Feind. Setzt einen Banditenturm (CB_RobberyTower1) in den Norden der Karte. Auch hier wird nun mittels einer nah am Turn platzierten „XD_ScriptEntity“ die Position festgelegt, an der die feindlichen Banditen erschaffen werden sollen.

Nun sind alle benötigten Objekte auf der Karte verteilt. Damit das Skript jedoch mit diesen kommunizieren kann, müssen diese Objekte noch den Spielern zugewiesen und benannt werden:

Standardmäßig besitzen alle Objekte die Spieler ID „1“ und gehören damit dem menschlichen Spieler.
Der Banditenturm soll jedoch dem KI Spieler gehören und bekommt deshalb die Spieler ID „2“. Außerdem soll er den Namen „player2“ bekommen.
Die in der Nähe des Banditenturms gesetzte XD_ScriptEntity kann die Spieler ID „1“ behalten, obwohl hier feindliche Einheiten erschaffen werden sollen, aber das Skript braucht dafür nur einen Namen – also tragt beim Namen „armyOne“ ein.
Der Bischof soll eine neutrale Figur sein und bekommt die Spieler ID „4“. Tragt für ihn als Namen „bishop“ ein.
Die neben ihm gesetzte „XD_ScriptEntity“ kann ebenfalls die Spieler ID „1“ behalten und bekommt den Namen „support“.



Euch wird aufgefallen sein, dass die Objekte englische Namen bekommen. Das ist nicht unbedingt nötig, erhöht aber die Übersichtlichkeit, weil die Skriptsprache auch in Englisch verfasst ist.

Der Held Dario bekommt den Namen „dario“ und seine Burg „headquarter“.

Auf dem Screenshot befindet sich zusätzlich noch die XD_ScriptEntity „chest“, auf die im Abschnitt 2 eingegangen wird. Sie soll im Moment ignoriert werden.

Nachdem jetzt auch die richtigen Spieler IDs und Namen gesetzt sind, kommt jetzt der wirklich spannende Teil: das Editieren des Skripts.

Klickt auf „Datei“ in der oberen Menüleiste und wählt „Skript bearbeiten“. Das Basisskript erscheint, welches nun etwas genauer betrachtet werden soll. Die grün eingefärbten Texte dienen nur zur Information und haben keinerlei Auswirkungen auf das eigentliche Skript. Sie sind dennoch sehr hilfreich, um die Übersichtlichkeit zu erhöhen und um **Kommentare** einzufügen.
Der restliche Text ist hauptsächlich schwarz eingefärbt, einige Schlüsselwörter **rot**.

Übrigens könnt ihr jederzeit Teile des Skripts auskommentieren, so dass diese im Spiel nicht mehr ausgeführt werden. Dazu müsst Ihr lediglich zwei Bindestriche vor die betreffende Zeile setzen.

Ganz oben im Skript könnt ihr vermerken, wie diese Karte heißt und wer sie erstellt hat. Danach werden verschiedene Funktionen eingelesen und initialisiert. So wird z.B. festgelegt, welches Wetter auf der Karte herrschen soll. Um das Skript effektiv editieren zu können, ist der eingebaute Skript-Editor jedoch nur begrenzt geeignet. Er dient eher als Schnittstelle zwischen Karte und Skript, so dass euer erster Schritt sein wird, das Basisskript der Karte zu exportieren, um es mit einem vollfunktionalen Editor weiter zu bearbeiten.
Klickt dazu im Skript-Editor auf der oberen Menüleiste auf „Skript“ und dann auf „Exportiere Skript“. Ein Fenster geht auf und ihr werdet gefragt, unter welchem Namen und wo das exportierte Skript abgelegt werden soll.
Gebt als Namen „tutorial_EinfacheMission“ ein und klickt auf „Speichern“. Das Skript ist jetzt im Ordner eurer Wahl als LUA-Datei abgelegt, die ihr mit einem herkömmlichen Editor wieder öffnen könnt. Es gibt eine Vielzahl von Editoren: UltraEdit ist eine Möglichkeit, es gibt aber auch viele freeware-Editoren, die fast den gleichen Leistungsumfang bieten. Nach dem Editieren und Abspeichern des Skripts im Editor, könnt ihr es dann mit dem Skript-Editor wieder importieren, die Karte speichern und testen.

Doch soweit ist es noch nicht. Im Moment habt ihr das Basisskript in einem beliebigen Editor vor euch liegen und nun ist es endlich an der Zeit, die einzelnen Punkte Schritt für Schritt der Mission anzupassen, die der Spieler zu bewältigen hat.

Wie schon erwähnt, dienen die ersten Zeilen lediglich der Bestimmung des Kartennamens und des Autors.

1.2 Hauptfunktionen

Danach werden die Hauptfunktionen geladen:

```
Script.Load( Folders.MapTools.."Main.lua" )
IncludeGlobals("MapEditorTools")
```

Ohne diese Informationen würde das Skript nicht funktionieren, daher solltet ihr hier nichts verändern.

1.3 DIPLOMATIE

Der nächste Punkt ist dagegen sehr interessant:

```
function InitDiplomacy()
end
```

Hier wird festgelegt, welche Spieler untereinander freundlich, neutral oder feindlich sind.

Für unsere Mission muss hier also definiert werden, dass die Banditen feindlich gegenüber dem menschlichen Spieler eingestellt sind.

Dies geschieht mit dem Befehl:

```
SetHostile(_playerId1,_playerId2)
```

Der menschliche Spieler hat die Spieler ID „1“, die Banditen die Spieler ID „2“.

Der Befehl müsste also folgendermaßen aussehen:

```
SetHostile(1,2)
```

Um ein neutrales Verhältnis zwischen dem Bischof (besitzt die Spieler ID „4“) und dem menschlichen Spieler zu erzeugen, benutzt ihr einen ähnlichen Befehl:

```
SetNeutral(1,4)
```

Schließlich könntet ihr auch Spielern untereinander eine freundliche Gesinnung geben:

```
SetFriendly(_playerId1,_playerId2)
```

Die gesamte Funktion sieht nun so aus:

```
function InitDiplomacy()
  SetHostile(1,2)
  SetNeutral(1,4)
end
```

1.4 Startressourcen

Die nächste Funktion gibt an, wie viele Startressourcen der menschliche Spieler bekommt:

```
function InitResources()
  AddGold  (1000)
  AddSulfur(1000)
  AddIron  (1000)
  AddWood  (1000)
  AddStone (1000)
  AddClay  (1000)
end
```

In diesem Fall bekommt der Spieler also von jeder verfügbaren Ressource 1000 Einheiten zu Spielbeginn.

1.5 Technologien

Weiter geht es mit den Technologien.

```
function InitTechnologies()
end
```

Normalerweise kann man diesen Punkt vernachlässigen. Wird nichts eingetragen, kann der Spieler jede Technologie, die das Spiel bietet, erforschen.

Um dem Spieler jedoch die Erforschung gewisser Technologien zu verbieten, zu erlauben oder automatisch zur Verfügung zu stellen, gibt es drei Befehle:

```
AllowTechnology(_technology, _playerID)
ForbidTechnology(_technology, _playerID)
ResearchTechnology(_technology, _playerID)
```

Der Parameter _playerID ist optional und standardmäßig auf den menschlichen Spieler (Spieler ID „1“) eingestellt.

Eine Liste aller globalen Technologien (GT):

GT_Mercenaries	→ Wehrpflicht
GT_StandingArmy	→ stehendes Heer
GT_Tactics	→ Taktiken
GT_Strategies	→ Pferdezucht
GT_Construction	→ Konstruktion
GT_GearWheel	→ Zahnräder
GT_ChainBlock	→ Flaschenzug
GT_Architecture	→ Architektur

GT_Alchemy	→ Alchimie
GT_Alloying	→ Legierungen
GT_Metallurgy	→ Metallurgie
GT_Chemistry	→ Chemie
GT_Literacy	→ Bildung
GT_Trading	→ Handelswesen
GT_Printing	→ Buchdruck
GT_Library	→ Büchereien
GT_Mathematics	→ Mathematik
GT_Binocular	→ Ferngläser
GT_Matchlock	→ Luntenschloss
GT_PulledBarrel	→ gezogener Lauf

Möchtet ihr z.B. dem Spieler die Technologie „Luntenschloss“ verbieten, ihm dafür aber die Technologie „Mathematik“ von Beginn an zur Verfügung stellen, sähe die komplette Funktion so aus:

```
function InitTechnologies()
  ForbidTechnology(Technologies.GT_Matchlock)
  ResearchTechnology(Technologies.GT_Mathematics)
end
```

Achtet aber gerade beim Verbieten von Technologien darauf, daß Ihr auch die Folgetechnologien verbietet.

1.6 Wetter

Das Wetter wird von zwei Funktionen beeinflusst:

Zum einen gibt es verschiedene Grundstimmungen, welche Auswirkungen auf die Farbe von Licht und Nebel haben:

```
function InitWeatherGfxSets()
end
```

Zur Auswahl stehen vier Grundstimmungen:

```
SetupNormalWeatherGfxSet()
SetupHighlandWeatherGfxSet()
SetupSteppeWeatherGfxSet()
SetupMoorWeatherGfxSet()
SetupEvelanceWeatherGfxSet()
```

Eine normale Grundstimmung würdet ihr z.B. so erhalten:

```
function InitWeatherGfxSets()
  SetupNormalWeatherGfxSet()
end
```

Zum anderen könnt ihr das Wetter auch direkt editieren. Ob Schnee, Regen, Sonne oder eine Abfolge von allem bleibt euch überlassen. Diese Funktion:

```
function InitWeather()
end
```

läßt sich mit folgenden Befehlen bestücken:

```
AddPeriodicSummer(_seconds)
AddPeriodicRain(_seconds)
AddPeriodicWinter(_seconds)
```

Es soll 120 Sekunden lang die Sonne scheinen, dann 60 Sekunden regnen und 60 Sekunden schneien? Dann sähe die Funktion so aus:

```
function InitWeather()
  AddPeriodicSummer(120)
  AddPeriodicRain(60)
  AddPeriodicWinter(60)
end
```

Sobald die Abfolge durchgelaufen ist, fängt sie wieder von vorne.

1.7 Startaktionen

Eine der interessantesten Funktionen ist die

```
function FirstMapAction()
end
```

Von hier aus werden weitere Funktionen beim Start der Karte aufgerufen. Für diese Mission ist das sehr wichtig, denn es sollen gleich mehrere Funktionen gestartet werden:

Zuerst soll der **Bischof** ein Ausrufezeichen über dem Kopf bekommen, damit ihn der Spieler ansprechen kann. Dazu wird ein Briefing benötigt, welches noch später angelegt wird.
Dann muss ein **KI Spieler** erzeugt werden, denn beim Start der Karte existiert vorerst nur der menschliche Spieler (Spieler ID „1“).
Für den KI Spieler soll daraufhin eine **Armee** erzeugt werden, die ebenfalls noch nicht existiert.

Zum Schluss muss noch eine Funktion aufgerufen werden, die permanent nachschaut, ob der **Banditenturm** vom KI Spieler noch existiert. Sobald dieser nämlich zerstört ist, hat der Spieler gewonnen.

Erweitert also die Startaktionen um folgende Einträge:

```
function FirstMapAction()
  createBriefingBishop()
  CreatePlayer2()
  CreateArmyOne()
  StartSimpleJob("VictoryJob")
end
```

Das macht jetzt noch wenig Sinn, aber ihr werdet gleich Schritt für Schritt erfahren, welche Funktionen da aufgerufen werden und welchen Zweck sie erfüllen.

1.8 Briefing

Damit der Bischof etwas sagen kann, muss ihm ein Briefing zugewiesen werden. Wie schon anfangs erwähnt, ist das Basisskript hier am Ende. Löscht also zuerst alle Zeilen unterhalb von

`end` der
`function FirstMapAction()`

Nun kopiert folgendes Briefing, welches ihr ganz unten im Skript einfügt:

```
function createBriefingBishop()

 BriefingBishop                          =  {}
 BriefingBishop.restoreCamera            =  true
 -- call on end of biefing
 BriefingBishop.finished                 =  BriefingBishopFinished

 local page = 0

 -- page #1

    page = page + 1

    BriefingBishop[page]                 =  {}
    BriefingBishop[page].title           =  "Bishop"
    BriefingBishop[page].text            =  "Hello, nice to meet you!"
    BriefingBishop[page].position        =  GetPosition("bishop")
    BriefingBishop[page].dialogCamera    =  true

 -- page #2

    page = page + 1

    BriefingBishop[page]                 =  {}
    BriefingBishop[page].title           =  "Bishop"
    BriefingBishop[page].text            =  "Destroy this enemy robbery tower."
```

```
    BriefingBishop[page].position        =  GetPosition("player2")
    BriefingBishop[page].explore         =  BRIEFING_EXPLORATION_RANGE
    BriefingBishop[page].marker          =  ANIMATED_MARKER


  -- page #3

    page = page + 1

    BriefingBishop[page]                 =  {}
    BriefingBishop[page].title           =  "Bishop"
    BriefingBishop[page].text            =  "These troops will help you."
    BriefingBishop[page].position        =  GetPosition("support")

  -- setup table for npc

    local npcBishop                      =  {}
    npcBishop.name                       =  "bishop"
    npcBishop.briefing                   =  BriefingBishop

  -- create npc

    CreateNPC(npcBishop)

end
```

Auf den ersten Blick sieht das Briefing verwirrend aus, aber im Detail betrachtet wird klar, was genau passiert:

Es wird eine Tabelle mit dem Namen „BriefingBishop“ erzeugt. Diese Tabelle kann nun mit den verschiedensten Variablen und Funktionen gefüllt werden.

Zuerst wird definiert, dass der Kamerablickwinkel nach Ende des Briefings wieder die alte Position einnimmt.

Außerdem soll nach Ende des Briefings die Funktion „BriefingBishopFinished“ aufgerufen werden (mittels dieser Funktion werden dann nach Ende des Briefings die Truppen für Dario generiert).

Nun werden die einzelnen Seiten des Briefings aufgeführt, in denen steht, wer in der jeweiligen Szene was spricht. Zusätzlich lassen sich die Seiten mit einigen nützlichen Funktionen ausstatten. So gibt es z.B. eine spezielle Dialog-Kamera-Sicht oder es können Bereiche aufgedeckt und markiert werden.

Wichtig sind die Positionen auf jeder Seite, die angeben, wo die Kamera jeweils hinschauen soll. Als Positionsnamen tauchen hier wieder die bekannten Namen auf, die vorher im Editor den einzelnen Objekten gegeben wurden.

Schließlich muss nach den einzelnen Seiten des Briefings noch der NSC selbst generiert werden, denn der Bischof ist momentan nur ein Objekt ohne jede Funktion, mit dem der Spieler nicht interagieren kann. Also wird ein NSC erzeugt, dem der Bischof und das Briefing zugewiesen werden.

1.9 Truppen für Dario

Hat Dario mit dem Bischof gesprochen, werden ihm Truppen zur Seite gestellt. Diese werden mit folgender Funktion erzeugt, welche ihr gleich nach der Briefing-Funktion einfügen könnt:

```
function BriefingBishopFinished()

 local troopDescription      = {

   minNumberOfSoldiers       = 0,
   maxNumberOfSoldiers       = 5,
   experiencePoints          = VERYHIGH_EXPERIENCE,
   leaderType                = Entities.PU_LeaderHeavyCavalry2,
   position                  = GetPosition("support")
 }

   local army                = {}
   local a1                  = CreateTroop(army,troopDescription)
   local a2                  = CreateTroop(army,troopDescription)

end
```

Mittels der troopDescription wird definiert, um welche Truppengattung es sich handelt, wo sie auf der Karte erscheinen soll, welche Minimal- und Maximal-Stärke und welche Erfahrung sie hat.
Danach werden zwei Truppen generiert, die jeweils auf die troopDescription zurückgreifen und dementsprechend aufgestellt werden.
Für fortgeschrittene Skripter ist es auch sehr einfach, diese Truppen zu steuern, denn sie haben einen eindeutigen Namen (a1, a2).

1.10 KI Spieler erzeugen

Nun ist endlich der KI Spieler an der Reihe. Er wird mit folgender Funktion erzeugt:

```
function CreatePlayer2()

 player2             = {}
 player2.id          = 2

 local description   = {serfLimit = 10}

 SetupPlayerAi(player2.id,description)

end
```

Hier wird im Grunde genommen nur eine weitere Funktion aufgerufen, die einen KI Spieler mit der ID “2” erzeugt. Ein kleiner, aber wichtiger Schritt.

Lest in der ScriptingReference (hier link einfügen) im detail nach, was diese Funktion tut.

1.11 EINE ARMEE FÜR DEN KI SPIELER

Weiter oben hat Dario ein paar Truppen bekommen. Diese Truppen sind einfach zu erzeugen, denn es sind keine KI-gesteuerten Truppen. Für den KI Spieler müsst ihr eine „richtige“ Armee kreieren, was etwas komplexer ist:

```
function CreateArmyOne()

 armyOne                = {}

 armyOne.player         = 2
 armyOne.id             = 1
 armyOne.strength       = 3
 armyOne.position       = GetPosition("armyOne")
 armyOne.rodeLength     = 500

 SetupArmy(armyOne)

 local troopDescription  = {

   maxNumberOfSoldiers   = 5,
   minNumberOfSoldiers   = 0,
   experiencePoints      = VERYLOW_EXPERIENCE,
 }

 troopDescription.leaderType = Entities.CU_BanditLeaderSword1

 EnlargeArmy(armyOne,troopDescription)
 EnlargeArmy(armyOne,troopDescription)
 EnlargeArmy(armyOne,troopDescription)

 Defend(armyOne)

end
```

Zuerst wird wieder eine Tabelle erzeugt (wie im Briefing). Diese wird nun mit Informationen zu der Armee gefüttert:

Sie gehört Spieler ID „2“

Sie hat die ID „1“ (nicht zu verwechseln mit der Spieler ID – Armeen unterscheiden sich untereinander durch diese ids)

Sie hat die Stärke 3. Das heißt, dass diese Armee maximal 3 Truppen aufnehmen kann.

Sie wird bei der Position „armyOne“ auf die Karte gesetzt (zur Erinnerung: für die Positionsbestimmung wurde eine XD_ScriptEntity mit dem Namen „armyOne“ verwendet)

Schließlich wird noch der Radius festgelegt, innerhalb welchem diese Armee auf ihre Feinde losgeht. 500 ist ein sehr kleiner Wert (500cm bzw. 5 Meter im Siedler-Maßstab).

Jetzt wird die Armee mit den eigentlichen Truppen aufgefüllt (eine KI-Armee kann man sich als einen Behälter vorstellen, den man mit Truppen füllt – gesteuert wird dann aber nur der Behälter bzw. die Armee). Hier kommt wieder die bekannte troopDescription ins Spiel.

Die Armee bekommt die maximalen drei Truppen und wird abschließend in den Defensiv-Modus gesetzt. Das heißt, dass diese Armee passiv vor dem Banditenturm stehen bleibt und nur reagiert, wenn der menschliche Spieler sich ihr bis auf unter 5 Meter nähert.

1.12 Siegbedingung

Der menschliche Spieler hat gewonnen, sobald der feindliche Banditenturm zerstört wurde.
Das lässt sich mit einer einfachen Funktion abfragen:

```
function VictoryJob()

 if IsDead("player2") then
   Victory()
 end

end
```

Zur Erinnerung: diese Funktion wurde als sogenannter „SimpleJob“ in der

```
function FirstMapAction()
```

gestartet.

Das heißt, dass dieser Job jede Sekunde ausgeführt wird und sobald die Funktion feststellt, dass der Banditenturm („player2“) nicht mehr existiert, hat der menschliche Spieler die Mission gewonnen.
Neben den SimpleJobs gibt es auch noch andere Jobs für vielfältige Einsatzgebiete, die in der ScriptingReference (hier link einfügen) dokumentiert sind.

Die erste Mission ist damit fertig!

Ihr könnt sie testen, indem ihr das Spiel startet und eure Karte aus der List der Freispielkarten auswählt.
Ihr könnt die Karte außerdem hier herunterladen, um die Schritte nachzuvollziehen:

tutorial_einfacheMission.s5x

Auch wenn die Mission sehr einfach ist, sind vom Prinzip her alle Missionen der offiziellen Karten auf diese Weise geskriptet worden.

2 SCHATZTRUHEN

Interaktive Schatztruhen, die man öffnen kann, werden leicht verwechselt mit den Truhen-Objekten. Bis auf eine äußerliche Ähnlichkeit haben sie aber nichts miteinander zu tun.

Um eine „echte“ Schatztruhe zu generieren, muss wieder die XD_ScriptEntity herhalten, die im Editor an die Position gesetzt wird, wo später im Spiel die Schatztruhe stehen soll.

Setzt also eine XD_ScriptEntity mit dem Namen „chest“ auf die Karte „tutorial Einfache Mission“ in die Nähe von Dario. Nun müsst ihr im Skript folgende Funktion hinzufügen:

```
function createChest()

 CreateRandomGoldChest(GetPosition("chest"))
 CreateChestOpener("dario")
 StartChestQuest()

end
```

Die Funktion ist sehr übersichtlich und zeigt an, wo die Schatztruhe erzeugt werden soll und wer in der Lage ist, sie zu öffnen.

Es können auch weitere Helden dazu befähigt werden, diese Truhe zu öffnen, aber in diesem Beispiel ist mit Dario nur ein Held auf der Karte vertreten, so dass er als alleiniger Truhenöffner ausreicht.

Auch diese Funktion muss noch bei den Startaktionen aufgerufen werden, so dass diese Funktion dann erweitert so aussieht:

```
function FirstMapAction()

 createBriefingBishop()
 CreatePlayer2()
 CreateArmyOne()
 StartSimpleJob("VictoryJob")

 createChest()

end
```

Nun gibt es im Spiel eine Schatztruhe, die eine zufällige Menge an Gold enthält.

3 MUSIK

Damit die verschiedenen Landschafts-Settings (european, highland, mediteranean, darkmoor) richtig zur Geltung kommen, sollte auch die dazu passende Musik gespielt werden.

Dazu öffnet ihr wieder das Skript und fügt in der

```
function FirstMapAction()
```

eine der folgenden Zeilen ein:

```
LocalMusic.UseSet = EUROPEMUSIC
LocalMusic.UseSet = HIGHLANDMUSIC
LocalMusic.UseSet = MEDITERANEANMUSIC
LocalMusic.UseSet = DARKMOORMUSIC
LocalMusic.UseSet = EVELANCENUSIC
```

4 Kartenbeschreibung

Klickt auf „Datei“ in der oberen Menüleiste und öffnet die „Karteninfo“.
Jetzt könnt ihr den Kartennamen und die Kartenbeschreibung editieren. Mit einem Klick auf „OK“ werden die Änderungen übernommen. Die Kartengröße ist ausgegraut und nicht editierbar. Ihr könnt sie nur ändern, indem ihr eine neue Karte erstellt.