WeatherExtensions Doku

Author: RobbiTheFox

Inhalt

1 Allgemein

Die *WeatherExtensions* bieten eine leistungsstarke Oberfläche für Mapper in *"Die Siedler - Das Erbe der Könige"*, mit denen vielschichtige, dynamische Wetterlagen erzeugt werden können.
In Kombination mit den *WeatherCallbacks* ist es zudem möglich, gezielt Aktionen zu bestimmten Wetterlagen ausführen zu lassen.

HINWEIS:
Diese Doku setzt Grundkenntnise zum Wetter in *"Die Siedler - Das Erbe der Könige"* voraus. Darunter z.B. die korrekte verwendung von `Logic.AddWeatherElement` und das erstellen und verwenden eigener gfx sets über die gfx id von 3 hinaus.

2 Begriffserklärung

Siedler
Wird Siedler als Präfix verwendet, so bezieht sich der nachfolgende Begriff auf die Originalmechanik im Spiel.
Beispiel:
Siedler gfx sets - hier sind die gfx sets im C code von Siedler gemeint
gfx sets - ohne Präfix sind die Daten in den *WeatherExtensions* gemeint

Ebene
Eine Ebene (auch *Dimension)* ist für einen bestimmten Aspekt des Wetters zuständig. Welche Ebene für welchen Aspekt zuständig sein soll, muss der Mapper selbst festlegen. Hierbei sind ihm in seiner kreativen Freiheit keine Grenzen gesetzt.
Ebene1 wird standardmäßig als Siedler Wetter interpretiert, kann jedoch auch anders verwendet werden.
Denkbare Aspekte sind z.B. Temperatur, Luftfeuchtigkeit, Niederschlag, Tageszeit usw.

Element
Elemente sind Teil der Ebenen und können mit den Siedler Wetterelementen verglichen werden, die mit `Logic.AddWeatherElement` erstellt werden. Genau wie diese, bestehen Elemente aus einer weather id, einer gfx id, Dauer, Vorlauf- und Übergangszeit.
Die Element id, Dauer, Vorlauf- und Übergangszeit verhalten sich genau wie bei Siedler Wetterelementen. Nur die gfx id macht hier einen Unterschied. Das tatsächlich verwendete gfx set setzt sich aus der Summe der gfx ids des jeweils aktiven Elements aller Ebenen zusammen (siehe *GfxSets erstellen*).
Eine Ebene kann theoretisch aus beliebig vielen Elementen bestehen.

Periode
Eine Periode besteht aus allen Elementen einer Ebene und verhält sich wie das periodische Siedler Wetter.

GfxSet
Ein *GfxSet* (auch *gfx set*) beinhaltet alle grafischen Daten für eine bestimmte Wetterlage. Die *WeatherExtensions* überschreiben die Funktionsweise der Siedler gfx sets und übernehmen selbst die Berechnung der Wetterübergänge.

3 Abhängigkeiten

Die *WeatherExtensions* benötigen lediglich den *TriggerFix* aus der *S5CommunityLib un*d können auch im normalen Siedler (ohne Serverfunktionen) verwendet werden.

4 Grundlagen

Die *WeatherExtensions* basieren auf einem System aus mehreren Ebenen, die unabhängig voneinander ihre jeweils eigene Periode durchlaufen.
Am einfachsten lässt sich das System am Beispielskript `weatherextensions_2dexample.lua` erklären.

Ebene 1 besteht aus dem bekannten Siedler Wetter mit Sommer, Regen und Winter.
Ebene 2 ist ein Tag-Nacht Zyklus bestehend aus Tag, Dämmerung und Nacht.

Mit den *WeatherExtensions* ist es möglich z.B. einen festen Tag-Nacht Zyklus zu erstellen, während das Wetter zwar seiner Periode folgt, jedoch auch auf Eingriffe mit dem Wetterturm oder andere Events auf der Karte korrekt reagiert, ohne dass sich der Mapper um die aktuelle Tageszeit kümmern muss.

Es müssen gfx sets für jede mögliche Kombination aus Elementen verschiedener Ebenen in der richtigen Reihenfolge angelegt werden (siehe *GfxSets erstellen*).

Je nach Komplexität des Wettersystems macht es Sinn, Comfort Funktionen zu schreiben wie im Beispiel oben `AddPeriodicDay`, `StartNight` usw. die am Ende auf `WeatherExtensions.AddElementToDimension( ... )` zurück greifen
(siehe *Elemente hinzufügen)*.

5 Verwendung

Um die *WeatherExtensions* verwenden zu können, muss
vor der Initialisierung der gfx sets und des Wetters
zunächst das Skript `weatherextensions.lua` geladen werden.
Es werden u.A. einige C und Comfort Funktionen überschrieben.

Hiernach sollten alle nötigen gfx sets definiert und Elemente den einzelnen Ebenen hinzugefügt werden. Ist ein gfx set nicht definiert, werden Standardwerte angenommen und es erscheint eine Warnung im *LuaDebugger*.
Ebenen müssen nicht zwangsläufig Elemente enthalten. Diese Ebenen haben dann keinen Einfluss auf das verwendete gfx set und es wird Element Id = 1 angenommen. Das hat den praktischen Hintergrund, einzelne Ebenen nur zeitweise verwenden zu können.

Gfx sets werden wie gewoht über die `Display.GfxSetSetXXX` Funktionen definiert oder über die Funktion `WeatherExtensions.GfxSetSetAll( ... )` (siehe *GfxSets erstellen*).
Das Wetter kann entweder mit den bekannten Funktionen `AddPeriodicXXX` und `StartXXX` oder über `WeatherExtensions.AddElementToDimension( ... )` erstellt werden (siehe E*lemente hinzufügen*). Letztere ist für komplexere Wettersysteme unabdingbar.

Als nächstes kann noch der Aufruf von
`WeatherExtensions.AppendDisplayGfxSet(_Id)`
folgen, wobei `_Id` sich auf die gfxsets bezieht. Diese Funktion wendet das übergebene gfxset statisch an. Das macht z.B. dann Sinn, wenn der Wetterzyklus erst zu einem späteren Zeitpunkt aktiviert werden soll, z.B. in EMS Karten nach Ablauf des 10s Timers.

Um den Wetterzyklus zu aktivieren ruft man
`WeatherExtensions.Start( _arg )`

auf. `_arg` ist optional und kann eine Reihe von Zahlen sein. Diese Geben den Index in der Periode der jeweiligen Ebene an, an dem gestartet werden soll.

Beispiel mit 2 Ebenen:
`WeatherExtensions.Start( 2, 3 )`

Ebene 1 startet mit dem zweiten Element und Ebene 2 mit dem Dritten.
Gäbe es noch eine dritte Ebene, würde diese mit dem ersten Element starten, da für sie kein Wert übergeben wurde.

6 GfxSets erstellen

Der wohl wichtigste Punkt beim Erstellen der gfx sets ist die Reihenfolge. Hierbei spielt sowohl die Anzahl der Ebenen als auch die Anzahl der verschiedenen Elemente pro Ebene eine Rolle.
Der einfachste Weg den Überblick zu behalten ist jedem Element einen individuellen Namen zu vergeben und eine Liste zu erstellen.

Alle Ebenen starten bei Element 1. Die kleinste Ebene zählt jeweils um 1 hoch.
Ist ihre maximale Anzahl an Elementen überschritten, wird sie auf 1 zurück gesetzt und die nächst höhere Ebene zählt um 1 hoch.
Hat die höchste Ebene ihre maximale Anzahl an Elementen überschritten, ist die Liste fertig.

Beispiel mit 2 Ebenen:

Ebene1, bestehend aus 3 Elementen (Sommer, Regen, Winter)
Ebene2, bestehend aus 2 Elementen (Tag, Nacht)

```
1    (1) Sommer  (1) Tag     Ebene1: 1 + 1 = 2
2    (2) Regen   (1) Tag     Ebene1: 2 + 1 = 3
3    (3) Winter, (1) Tag     Ebene1: 3 + 1 = 4
```

das maximum von 3 in Ebene 1 wurde überschritten
Ebene1: = 1, Ebene2: 1 + 1 = 2

```
4    (1) Sommer  (2) Nacht   Ebene1: 1 + 1 = 2
5    (2) Regen   (2) Nacht   Ebene1: 2 + 1 = 3
6    (3) Winter, (2) Nacht   Ebene1: 3 + 1 = 4
```

das maximum von 3 in Ebene 1 wurde überschritten
Ebene1: = 1, Ebene2: 2 + 1 = 3

das maximum von 2 in Ebene 2 wurde überschritten
Die Liste ist fertig.

Gfx sets können wie gewohnt mit den bekannten `Display.GfxSetSetXXX` Funktionen erstellt werden, oder mit einer etwas vereinfachten Funktion für alle Werte gleichzeitig.

```
WeatherExtensions.GfxSetSetAll
(_Id, _Rain, _Snow, _Ice, _Fog, _Dist, _Shad, _Amb, _Dif, _Sky)
```

Beispiel für Sommer aus dem Normal set

```
WeatherExtensions.GfxSetSetAll (
     1, 0,0,0,
     Color(147,177,187),   -- Nebelfarbe
     Range(9000,28000),    -- Nebelentfernung
     Position(40,-15,-50), -- Schatten Koodinaten
     Color(120, 110, 110), -- Amient Licht
     Color(205,204,189),   -- Diffuse Licht
     "YSkyBox02"           -- SkyBox )
```

7 Elemente hinzufügen

Einer Ebene kann man neue Elemente mit
`WeatherExtensions.AddElementToDimension`
`( _Dim, _Id, _Dur, _Per, _Gfx, _Fore, _Tran, _Index )`
hinzufügen.

`_Dim` gibt die Ebene an, in die das Element hinzugefügt werden soll. Ist die Ebene noch nicht vorhanden, wird diese neu erstellt.

`_Index` ist optional und nur für periodische Elemente nutzbar. Dieser gibt die Position in der Periode an, an der das Element eingefügt werden soll. Nachfolgende Elemente rutschen um eine Position nach hinten. Wird `_Index` nicht angegeben, wird das Element am Ende der Periode angefügt.

`_Id`, `_Dur`, `_Per`, `_Gfx`, `_Fore` und `_Tran` werden genau wie in Logic.AddWeatherElement verwendet, mit dem einzigen Unterschied, dass `_Gfx` nicht absolut ist, sondern additiv.

In Ebene1 sollte `_Gfx` immer gleich `_Id` sein.
Für alle weiteren Ebenen muss etwas gerechnet werden:
`_Gfx` Element(1) ist immer gleich 0
`_Gfx` Element(`_Id`) ist (`_Id` -1) * dem Produkt der Anzahl an Elementen der vorherigen Ebenen. Klingt kompliziert, ist es aber nicht.

Beispiel mit 3 Ebenen

Ebene1, bestehend aus 3 Elementen
Ebene2, bestehend aus 2 Elementen
Ebene3. bestehend aus 3 Elementen

Ebene1:
Element1 = 1
Element2 = 2
Element3 = 3

Ebene2:
Element1 = (1 - 1) * (nEbene1) = 0 * 3 = 0
Element2 = (2 - 1) * (nEbene1) = 1 * 3 = 3

Ebene3:
Element1 = (1 - 1) * (nEbene1) * (nEbene2) = 0 * 3 * 2 = 0
Element2 = (2 - 1) * (nEbene1) * (nEbene2) = 1 * 3 * 2 = 6
Element3 = (3 - 1) * (nEbene1) * (nEbene2) = 2 * 3 * 2 = 12

Um die genaue Verwendung noch besser nachvollziehen zu können, können die Comforts aus `weatherextensions_3dexample.lua` herangezogen werden.

8 Templates nutzen

8.1 Essentielle Templates

Essentielle Templates müssen immer einen Wert zurückgeben.

`WeatherExtensions.Template_GetNextWeatherState()`

Gibt standardmäßig die Id des nachfolgenden Elements von Ebene1 zurück.
Diese Funktion ist wichtig für die Wettervorhersage. Bei komplexeren Wettersystemen, in der der Siedler weather state von mehreren Ebenen beeinflusst wird, muss an dieser Stelle eine manuelle Berechnung durch den mapper erfolgen
(siehe `weatherextensions_3dexample.lua`).

`WeatherExtensions.Template_GetTimeToNextWeatherState()`

Gibt standardmäßig die Zeit bis zum nachfolgenden Element von Ebene1 zurück.
Dieser Wert ist ebenso wichtig für die Wettervorhersage und muss wie `WeatherExtensions.Template_GetNextWeatherState` ggf. auch manuell vom mapper berechnet werden.

`WeatherExtensions.Template_GetWeatherStateAndTransition()`

Gibt standardmäßig die Id und transition Zeiten des aktuellen Elements von Ebene1 zurück. Diese Funktion ist essentiel wichtig für das aktuelle Wetter und muss wie die vorherigen Funktionen bei komplexeren Systemen vom mapper manuell berechnet werden.

`WeatherExtensions.Template_GetElementDefault(_Dim, _Id, _Dur, _Gfx, _Fore, _Tran)`

Aus Kompatibilitätsgründen wird standardmäßig das System von Ubi verwendet, welches die transition Vorlaufzeit zur duration hinzu addiert.
Wenn die duration nicht verändert werden soll, muss die return Zeile ersetzt werden:
`return WeatherExtensions.GetElementDefault(_Dim, _Id, _Dur, _Gfx, _Fore, _Tran)`
An dieser Stelle können auch andere element defaults definiert werden.
z.B um eigene defaults für verschiedene Ebenen zu definieren.

8.2 Optionale Templates

Optionale Templates können nach Bedarf mit Funktionalität bestückt werden.

```
WeatherExtensions.Template_GetShadows( _Gfx )
```

Wird jeden Tick aufgerufen und kann genutzt werden, um die Schatten manuell zu berechnen, z.B. für einen Tag-Nacht Zyklus.
Als Parameter steht das aktuell verwendete gfx set zur Verfügung.
Hinweis:
Die Funktion wird 2x aufgerufen. 1x mir dem aktuellen und 1x mit dem vorherigen gfx set. Das hängt damit zusammen, wie das Skript dem Spiel die tatsächlichen Wetterdaten übermittelt.

```
WeatherExtensions.Template_OnStateChanged(_Dim, _OldId, _NewId)
```

Alternative zu einem StateCallback (siehe *StateCallbacks*)

```
WeatherExtensions.Template_OnGfxChanged(_OldId, _NewId)
```

Alternative zu einem GfxCallback (siehe *WeatherCallbacks*)

```
WeatherExtensions.Template_EveryTick()
```

Wird in jedem game tick aufgerufen, solange die WeatherExtensions aktiv sind.

9 WeatherCallbacks

Die *WeatherCallbacks* sind ein unabhängiges Skript, welches auch ohne die *WeatherExtensions* verwendet werden kann. In diesem Fall sind diese jedoch von *CUtilMemory* abhängig. In jedem Fall benötigen aus sie den *Triggerfix*.

Initialisierung

1. Lade `weathercallbacks.lua`
1a (optional) Lade `weathercallbackextensions.lua`
2. Führe aus `WeatherCallbacks.Init()`

Hinweis:
Werden die *WeatherCallbackExtensions* nach der Initialisierung der *WeatherCallbacks* geladen, muss `WeatherCallbacks.Init()` erneut ausgeführt werden, da es sonst zu doppelaufrufen in den gfx callbacks kommen kann.

Die *WeatherCallbacks* bestehen aus 2 Komponenten. Eine Dritte kommt mit den *WeatherCallbackExtensions* hinzu, welche eine spezielle Schnittstelle zu den *WeatherExtensions* bereitstellt.

WeatherCallbacks - werden ausgeführt, wenn sich der weather state ändert.

`WeatherCallbacks.AddWeatherCallback(_Id, _CallbackStart, _CallbackEnd)`
Fügt einen weather callback hinzu.

`_Id` - der weather state, der das callback auslösen soll
`_CallbackStart` - Funktion die zu Beginn dieses weather states ausgeführt werden soll
`_CallbackEnd` - Funktion die zum Ende dieses weather states ausgeführt werden soll

Eine Mehrfachbelegung ist möglich.
In diesem Fall werden alle jemals übergebenen Funktionen ausgeführt.

`WeatherCallbacks.RemoveWeatherCallbacks(_Id)`
Entfernt alle Callbacks für den übergebenen weather state.

GfxCallbacks - werden ausgeführt, wenn sich das gfx set ändert.

`WeatherCallbacks.AddGfxCallback(_Id, _CallbackStart, _CallbackEnd)`
Fügt einen gfx callback hinzu.

`_Id` - die gfx Id, die das callback auslösen soll
`_CallbackStart` - Funktion zu Beginn
`_CallbackEnd` - Funktion zum Ende

Eine Mehrfachbelegung ist möglich.

`WeatherCallbacks.RemoveGfxCallbacks(_Id)`
Entfernt alle Callbacks für die übergebenen gfx Id.

10 WeatherCallbackExtensions

StateCallbacks - werden ausgeführt, wenn sich der state der übergebenen Ebene ändert (benötigt *WeatherExtensions*)

`WeatherCallbacks.AddStateCallback(_Dim, _Id, _CallbackStart, _CallbackEnd)`
Fügt einer Ebene einen state callback hinzu.

`_Dim` - die Ebene, aus der die Id gelesen werden soll
`_Id` - die Element Id, die das callback auslösen soll
`_CallbackStart` - Funktion zu Beginn
`_CallbackEnd` - Funktion zum Ende

Eine Mehrfachbelegung ist möglich.

`WeatherCallbacks.RemoveStateCallbacks(_Dim, _Id)`
Entfernt alle Callbacks für die übergebene Element Id in der übergebenen Ebene.