Einführung

Mod

Regeln von Computerspielen werden klassischerweise im Quellcode festgelegt.

Manche Spiele lesen allerdings Teile der Regeln aus Konfigurationsdateien.

Damit kann man das Spiel ändern, ohne den Hauptquellcode zu ändern
$\Rightarrow$ Modifikation des Spiels, kurz Mod

Historische Beispiele: Änderung der Bewegungsreichweite

Civilisation (1991)
Binär editierbar

Colonization (1994)
ASCII-Format editierbar

Mod, historische Entwicklung

- Mit schnelleren Computern wanderten immer mehr Regeln in Konfigurationsdateien: Effizientere Spielentwicklung
- Steigende Anzahl inoffizieller Mods durch Spieler
- Seit dem Half-Life (1998)-Mod Counterstrike (1999) sind Mods auch in direkter Aufmerksamkeit der Spieleentwickler
- Neue Spiele kommen immer öfter mit der expliziten Möglichkeit komplexe Mods zu integrieren

Von Konfigurationsänderungen zu vollen Mods

Alte Spiele erlauben oft nur kleine Änderungen durch geänderte Konfigurationsdateien: Geschwindigkeit von Einheiten, Masse von Gegenständen, Levellayout, ...

Steigende Moddinganforderungen haben in einigen Spielen zum "versehentlichen" Entwickeln von Programmiersprachen geführt: Stückweise mehr hinzugefügt, bis beliebige Programme geschrieben werden konnten (Beispiel: StarCraft).

Lösung 1: Beliebige Programme können als Konfiguration hinzugefügt werden

Nachteil: Mangelnde Portabilität, Einfachheit der Konfiguration geht oft wieder verloren

Lösung 2: Einfache Programmiersprache entwickeln, die sich für eine Vielzahl von Projekten eignet und von Grund auf dafür durchgedacht ist. Beispiele: Lua, Squirrel

Lua

Lua (portugiesisch: Mond) ist eine Skriptsprache.

- Optimiert auf hohe Ausführungsgeschwindigkeit und kleinen Interpreter (120 kB)
- Eignet sich gut, um in andere Projekte eingebaut zu werden
- Wird vielfältig für verschiedene Softwareprojekte eingesetzt
  - Spiele: Minetest, The Battle for Wesnoth, Civilization VI, SimCity 4, Factorio, . . . (191 Einträge in der englischen Wikipedia)
  - Sonstige Software: Adobe Photoshop Lightroom, Apache HTTP Server, TeamSpeak, Wireshark, . . . (77 Einträge in der englischen Wikipedia)
  - Embedded: Olivetti drucker, Kindle Touch, . . .

Lua – Grundlegende Syntax

Lua lässt an mehreren Stellen syntaktische Freiheiten, Beispiel Hello World-Programm:

```
print("Hello World!")
print 'Hello World!'
```

Kommentare beginnen mit **--**, mehrzeilige Kommentare

```
--[[
  schreibt man in doppelten eckigen Klammern
  mit vorangestelltem --
  ]]
```

Lua – Grundlegende Semantik

Lua ist schwach typisiert

```
local x = 42
local y = x + 5
print ("Auch eine Antwort: " .. y)
y = nil -- undefiniert, null
y = { "U", "V", "W", "X" } -- y ist jetzt ein Array
y = {a = 10, b = {1, 2, 3}} -- y ist jetzt eine Table
```

Table in Lua entspricht dem Konzept eines assoziativen Arrays, auch bekannt als

- Map: C++, Java, go, Scala
- Dictionary: C#, Julia, PostScript, Python
- Hash: Perl, Ruby

Variablen sind in Lua global, alle Variablennamen existent; vorangestelltes `local` macht diese Variable lokal.

Lua – Kontrollfluss

Gängige Kontrollflusskonstrukte: if-then-else, for, while

```
if num < 90 then
  ...
elseif num > 9000 then
  ...
else
  ...
end

for i = 1, n do
  -- 1,2,3,...
end

while i < 137 do
  ...
end
```

Lua – Funktionen

```
function malzwei(x)
  return 2 * x
end
```

Funktionen werden als First-Class-Funktion betrachtet: `malzwei` ist eine Variable, die als Inhalt diese Funktion bekommen hat. Diese Definition ist äquivalent zu

```
malzwei = function(x)
  return 2 * x
end
```

In beiden Fällen können wir danach schreiben

```
timestwo = malzwei

print timestwo(21) -- 42
```

Minetest

Minetest ist 2010 entstanden und ebenso wie Minecraft (2011) inspiriert durch das Spiel Infiniminer (2009).

Verbindendes Konzept: Es gibt eine große Spielewelt aus einem rechtwinkligem 3-D-Gitter aufgebaut

Infiniminer hatte relativ viel vorgegebene Handlung; kam nicht gut an, Entwicklung wurde frühzeitig eingestellt.

$\Rightarrow$ Minetest und Minecraft haben die vorgegebene Handlung weggelassen

Minecraft ist wesentlich umfangreicher und verbreiteter als Minetest, dementsprechend hat Minetest im Laufe der Zeit einiges von Minecraft übernommen, aber auch eigene Inhalte

Minetest – Spielewelt

Minetest ist aus einem rechtwinkligen 3-D-Gitter von biszu $61\,840 \times 61\,840 \times 61\,840$ Nodes aufgebaut.

Würde man nur 1 Byte pro Node brauchen (meist verwendet das Spiel mehr), so bräuchte man über 200 TB Speicherplatz.

$\Rightarrow$ Es wird immer nur ein kleiner Bereich berechnet, der gerade im Spiel gebraucht wird

Minetest – Modding API

Eine mehr oder weniger vollständige und aktuelle Dokumentation mit allen Fähigkeiten der Modding-API findet sich unter https://github.com/minetest/minetest/blob/master/doc/lua_api.txt

Beispiel: Node setzen

```
minetest.add_node(pos, node)
```

Wobei node eine table ist, die zumindestens den Namen der node enthält.

Minetest – Mods installieren

Mods sind immer ein Verzeichnis und die darin liegenden Dateien und Verzeichnisse. Zum Installieren in `mintestverzeichnis/mods/modname` kopieren.

Einfacher Mod auf der Vorlesungsseite verfügbar.

Um sich mit Mods vertraut zu machen gerne einige verschiedene Mods installieren und ausprobieren.

Praktikumsserver

Es gibt einen Server für das Praktikum: Leeres Ubuntu-System in einer virtuellen Maschine.

Zugangsdaten werden jetzt bekannt gegeben.

Ports 30000-30020 sind für Minetest-Server freigeschaltet.

Veröffentlichung am Praktikumsende – WTFPL

Am Ende des Praktikums werden alle erfolgreichen Projekte unter der WTFPL veröffentlicht

Die WTFPL (Do What the Fuck You Want To Public License)

- ist eine open-source-Lizenz; es ist strittig, ob die WTFPL tatsächlich eine open-source-Lizenz ist, oder nur die Weiterverwendung als open-source ermöglicht
- ist im allgemeinen nicht zu empfehlen
- ist im Minetest-Umfeld extrem verbreitet, darum für Minetest-Mods zu empfehlen