Neue Modder braucht das Land

Workshop – Einführung in LUA

Manuel Leithner aka Face
SFM-Modding
Kontakt: sfm_f@web.de
Homepage: http://forum.landwirtschafts-simulator.de

Gliederung

1. Grundlagen des Game-Development
2. Bäume, Bäume, Bäume aber kein Wald in Sicht
3. Grundlagen XML
4. LUA-Scripting – Einführung
5. LUA-Scripting – Fortgeschrittenes
6. Grundlagen der GIANTS – Engine
7. LUA-Scripting – Landwirtschafts-Simulator 2009
8. Zusammenfassung / Feedback / Fragen

Gliederung

2. Bäume, Bäume, Bäume aber kein Wald in Sicht

   2.1. Datenstrukturen in der Informatik

   2.2. Baum-Strukturen

Gliederung

Datenstrukturen in der Informatik

- Was ist eine Datenstruktur

  „Mathematisches Objekt zur Speicherung von Daten. Es handelt sich um eine Struktur, weil Daten in einer bestimmten Art und Weise angeordnet und verknüpft werden, um den Zugriff auf sie und ihre Verwaltung geeignet zu ermöglichen. Datenstrukturen sind daher nicht nur durch ihre beinhalteten Daten charakterisiert, sondern vor allem durch die Operationen auf diesen Daten, welche Zugriff und Verwaltung realisieren“

- Warum benötigt man Datenstrukturen:
  - Zusammenfassung logisch zusammengehöriger Daten
  - Aufwandsminimierung (siehe nächste Folie)
  - Möglichkeit der Wiederverwendbarkeit

Aufwandsoptimierung

- Was ist Aufwand?:
  - Kosten (nicht unbedingt = Zeit) die benötigt werden um ein Element anzusteuern

- Beispiel: Papierstapel (eine reale Datenstruktur ☺)
  - Es liegt ein Papierstapel mit 50 Rechnungen vor dir
  - Du suchst nun nach der letzten Rechnung deiner Amazon Bestellung
    Fatalerweise sind die Rechnungen nicht sortiert! ☹
  - Der Aufwand dieser Suchoperation:
    - BestCase: Die Rechnung liegt GANZ oben -> Aufwand von 1 Vergleich
    - WorstCase: Die Rechnung liegt GANZ unten -> Aufwand von 50 Vergleichen
    - Ø: Im Schnitt benötigt man 25 Vergleiche
  - Nun haben wir eine neue Rechnung bekommen und wollen diese hinzufügen:
    - BestCase: Wir legen die Rechnung oben drauf -> Aufwand 1 Operation
    - WorstCase: Wir legen die Rechnung oben drauf -> Aufwand 1 Operation
    - Ø: Im Schnitt benötigen wir 1 Operation
  - Das Entfernen ist eine Kombination von Suchen und Herausnehmen

Aufwandsoptimierung

- Aufwandoptimierung:
  - Das Suchen in unserem Papierstapel ist abhängig von der Anzahl der Blätter ☹
    - Was machen wir wenn unser Papierstapel statt 50, 1 Million Blätter hat?
    - Wir benötigen eine bessere Datenstruktur -> Aufwandsoptimierung
    - Welche Möglichkeiten hätten wir?

Aufwandsoptimierung

- Beispiel: Papierstapel
  - Wir Sortieren den Papierstapel (z.B. nach Datum)
  - Der Aufwand dieser Suchoperation:
    - BestCase: Die Rechnung liegt GANZ oben -> Aufwand von 1 Vergleich
    - WorstCase: Die Rechnung liegt GANZ unten -> Aufwand von 50 Vergleichen
    - Ø: Im Schnitt benötigt man 25 Vergleiche (eher < 25)
  - Nun haben wir eine neue Rechnung bekommen und wollen diese hinzufügen:
    - BestCase: Es ist die neuste Rechnung -> Aufwand 1 Operation (Oben drauflegen)
    - WorstCase: Die Rechnung ist schon älter
      Wir müssen den Papierstapel durchsuchen um das letzt kleinere Datum zu finden
      -> evtl. müssen wir den ganzen Stapel durchsuchen
    - Ø: Im Schnitt benötigen wir 25 Vergleiche/Operation
  - Das Entfernen ist eine Kombination von Suchen und Herausnehmen
  - Die Einführung einer Sortierung hat nicht viel gebracht? Was nun?

Aufwandsoptimierung

- Beispiel: Papierstapel
  - Wir bauen aus unserem Papierstapel einen „Baum“
  - Wir fügen neue Registerkarten ein z.B. Rechnungen vom März 2010

  - Der Aufwand dieser Suchoperation: (Wir wissen das Datum der Rechnung)
    - BestCase: Wir schnappen uns die Registerkarte des Monats und durchsuchen nur die darin enthaltenen Rechnungen
      Die Rechnung ist an erster Stelle der Registerkarte -> Aufwand 1
    - WorstCase: Die Rechnung liegt in der Registerkarte GANZ unten
      -> Aufwand von Anzahl der Märzrechnungen

  - Das Entfernen und Einfügen ist eine Kombination mit dem Suchen.

  - Wie man sieht muss man nun nicht mehr den ganzen Stapel durchsuchen. Wir haben also den Aufwand optimiert.

  - Weitere Optimierung möglich durch:
    - weiteres Einfügen von Registerkarten (z.B. Wochen)
    - andere Datenstrukturen

Wichtig

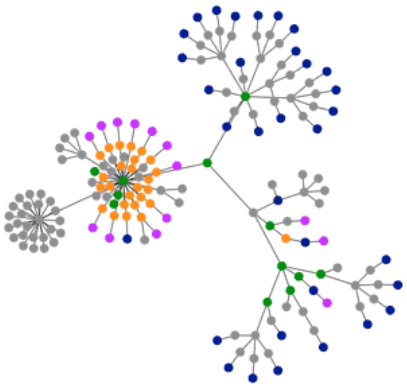
- Am Beispiel kann man erkennen das eine Datenstruktur Aufwände minimieren kann
- ABER: Nur möglich durch weitere „Verschwendung“ von Speicherplatz (Hier: Registerkarten)
- Folge: Schnellere Berechnung benötigt mehr Speicher!

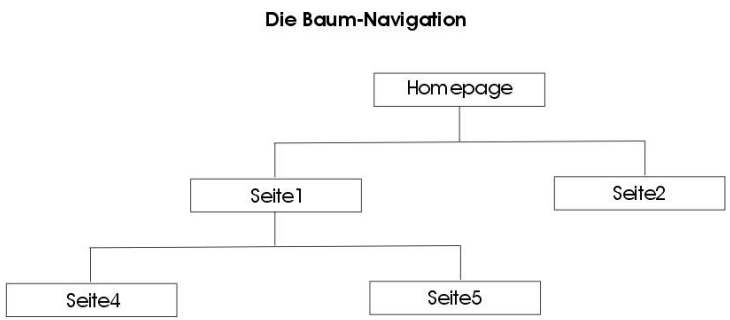
Gliederung

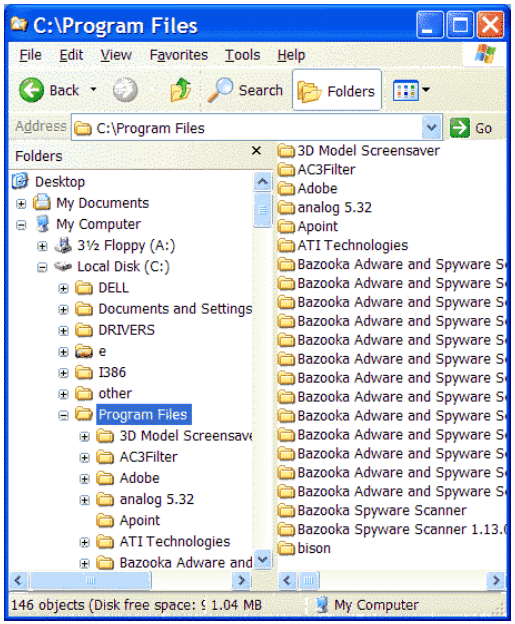
Baum-Strukturen

- Wie sieht ein Baum aus:
  - Wurzelknoten (Root) (Im Beispiel: z.B. die Ablageschale)
  - 1-n Kindknoten (Im Beispiel: z.B. die Registerkarten)
  - Diese können weitere Kindknoten haben (Im Beispiel: Registerkarten oder Rechnung)

- Beispiele für Bäume:







Kritische Beurteilung von Bäumen

- Vorteile:
  - Aufwandsoptimierung
  - Klare Hierarchien durch Parent-Child-(Vater-Kind)-Verknüpfungen
    - Jedes Kind kann nur 1 Vater haben!
  - Übersichtliche Darstellung

- Nachteile:
  - Es wird mehr Speicher verbraucht

Verwendung bei LS2009

- Baumstrukturen finden sich überall
  - Grundaufbau der Engine
  - Aufbau der I3D
  - XML ist ebenfalls ein Baum
  - Anordnung des Mod-Ordners im Explorer
- Mehr dazu in den nächsten Veranstaltungen

Fragen / Feedback

- Gibt es noch Fragen?
- Was sollte besser gemacht/geändert werden?
- Welche Inhalte würden euch interessieren oder fehlen noch?

Übungen bis zur nächsten Veranstaltung

- Aufgabe 1 (Recherche):
  - Suche min. 2 weitere Datenstrukturen
    - Nenne Vorteile, Nachteile und Erläutere kurz den Aufbau

- Aufgabe 2 (Recherche):
  - Problemstellung: Gegeben sei der Punkt (x=3, y=4) im Kartesischen Koordinatensystem
    Es soll nun berechnet werden, wo sich der Punkt befindet, wenn man das Koordinatensystem um den Nullpunkt um PI dreht
  - Aufgabe:
    - Finde einen Rechenweg!
    - Erläutere kurz deine Vorgehensweise bei der Suche nach einer Lösung
    - Berechne die neuen Koordinaten
    - Wichtig: Das stupide eingeben in einen Mehrfunktionalen Taschenrechner ist nicht ratsam, da ein Script die Funktionen eines solchen Taschenrechners nicht unterstützt

- Einsendeschluss: Samstag, 6.März 2010 – 24 Uhr

- Nächster Termin: Sonntag, 7.März 2010 – 20 Uhr