Zellomat3D

Design Prototypen

Projekt:	3D Cellular Automata Simulator – Diplomarbeit – SS/2005	
Auftraggeber:	Hochschule Rapperswil HSR	
Betreuer:	Eduard Glatz – Prof. Dipl. Ing. ETH	eglatz@hsr.ch
Mitarbeiter:	Michael Florin	loop@loop.li
	Andreas Weinmann	a.weinmann@gmx.ch
Ablage:	DesignPrototypen - 21032005.doc	

Inhaltsverzeichnis

1. Einführung

Zweck

Dieses Dokument beschreibt die Architektur und das Design der evolutionären Prototypen. Es erleichtert einem Entwickler den Einstieg in den Programmcode zu erleichtern, damit dieser die die Funktionsweisen der Prototypen nachvollziehen kann. Des Weiteren ist daraus ersichtlich, warum ein solches Design und eine solche Architektur verwendet wurden.
Zu Beginn wird die Architektur der evolutionären Prototypen beschrieben, danach folgen das Klassendiagramm und zum Schluss einige Sequenz- und Kollaborationsdiagramme, die den dynamischen Aspekt der Prototypen dokumentieren.

Gültigkeitsbereich

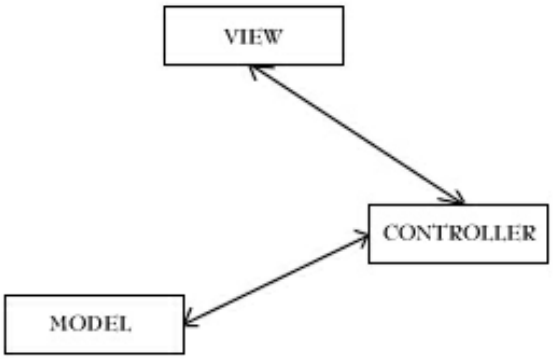
Dieses Dokument gilt für die Diplomarbeit "Zellomat3D", welche im SS/2005 an der Hochschule Rapperswil HSR durchgeführt wurde.

2. Architekturmodell

Architektur

Die zu entwickelnden evolutionären Prototypen sind nach dem Prinzip des Model-View-Controller-Pattern aufgebaut. Das heisst, diese werden schon nach dem Aspekt des MVC-Patterns entwickelt, so dass ein späterer Zusammenbau bzw. eine Weiterentwicklung zur fertigen Applikation keinen grösseren Zusatzaufwand mehr benötigt

MCV-Pattern



Grund

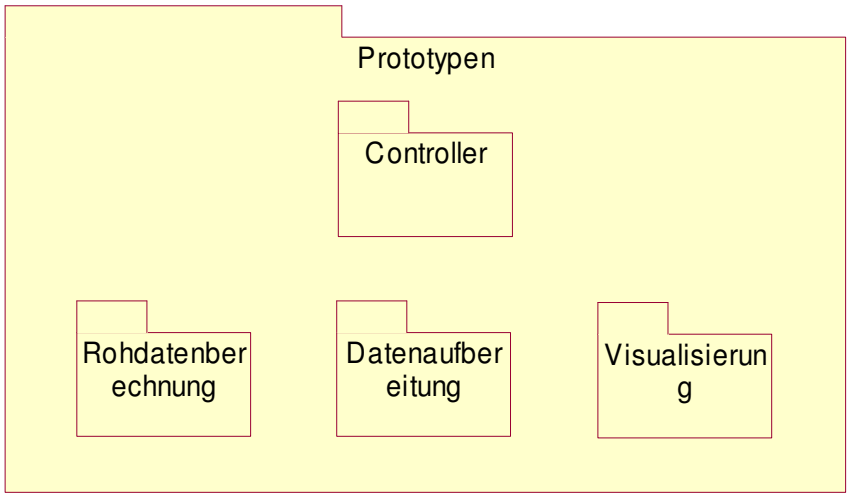
Dieses Pattern scheint ideal für die Realisierung des Projektes, denn Ziel des Modells ist es, ein flexibles Programmdesign zu ermöglichen, um eine spätere Änderung oder Erweiterung einfach zu halten und die Wiederverwendbarkeit der einzelnen Module zu ermöglichen. Außerdem sorgt das Modell für eine gewisse Übersicht und Ordnung in der Anwendung, denn durch das saubere Trennen der einzelnen Module können diese getrennt realisiert werden, ohne irgendwelche Abhängigkeiten zu anderen Modulen, ausser dem Controller, zu haben.

Ausnahmen

Einzig die Steuerung des Moduls “Visualisierung“ erfolgt direkt, ohne Umwege über den Controller. Dies, weil dieses Module ein eigenes GUI besitzt und es aus Performancegründen Sinnvoll ist, die Steuerung direkt mit dem Modul zu koppeln. Denn die Steuerung des Users muss so schnell wie möglich in die Drehung und Bewegung des Bildes umgesetzt werden.

3. Klassendiagramm

Struktur der Klassenkategorien



Hinweise

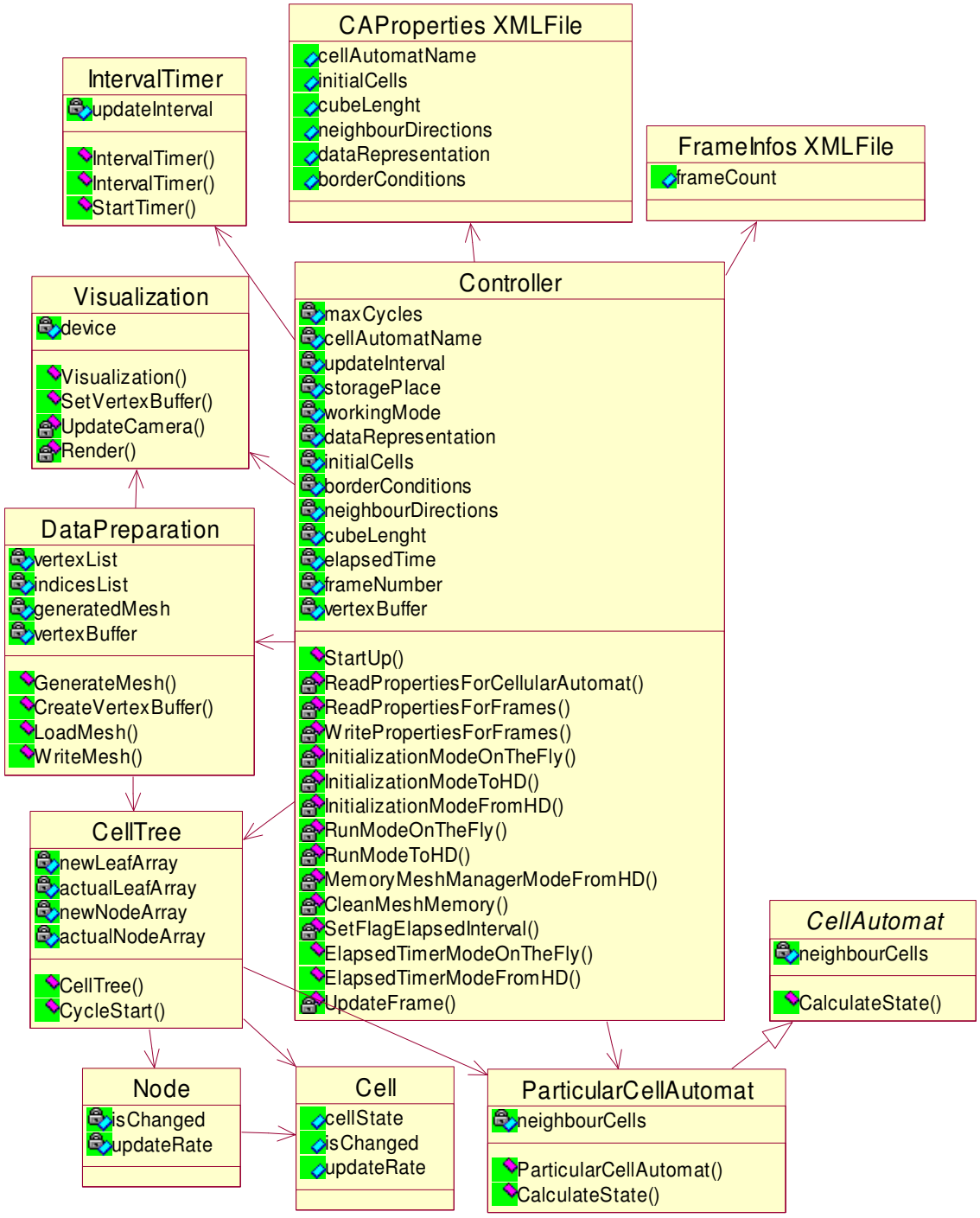
Die Prototypen haben je einen eigenen Namespace. Der Controller wird erst in der 2. Iteration entwickelt, dies weil zuerst mittels den Prototypen ermittelt werden muss, wie die Applikation später im Detail aufzubauen ist und welche Vorstellungen zu realisierbar sind.

Das Modul des Controllers wird hier aber trotzdem in das Design des Klassendiagrammes miteinbezogen, dies weil die Definitionen der Schnittstellen unabdingbar für ein späteres Kombinieren der Module sind.

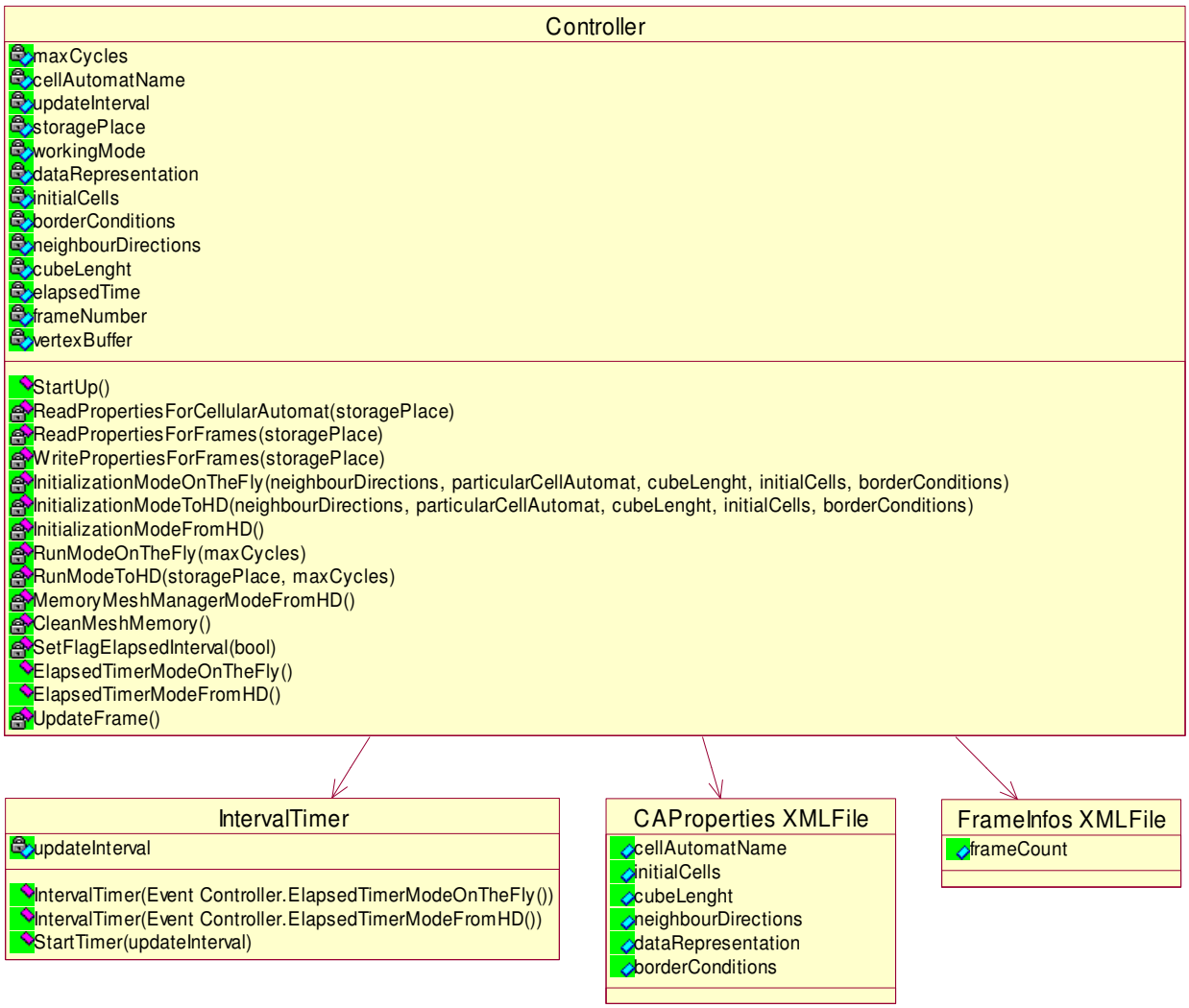
Der Grad der Detailgenauigkeit der einzelnen Module, ausser dem Controller, ist gering gehalten, weil bewusst noch nicht genau bekannt ist, was später aus den gewonnenen Erkenntnissen des Prototypenbaus weiterentwickelt werden kann.

Es wird auf ein „low-coupling“ zwischen den einzelnen Modulen Wert gelegt. Alles kann aber aus Performancegründen nicht durch den Controller gesteuert werden. Ausser diesen wenigen Interaktionen zwischen den Modulen selbst regelt der Controller den Grossteil der Abläufe der Applikation.

Hier der Gesamtüberblick der zu entwickelnden evolutionären **Overall** Prototypen der einzelnen Module.



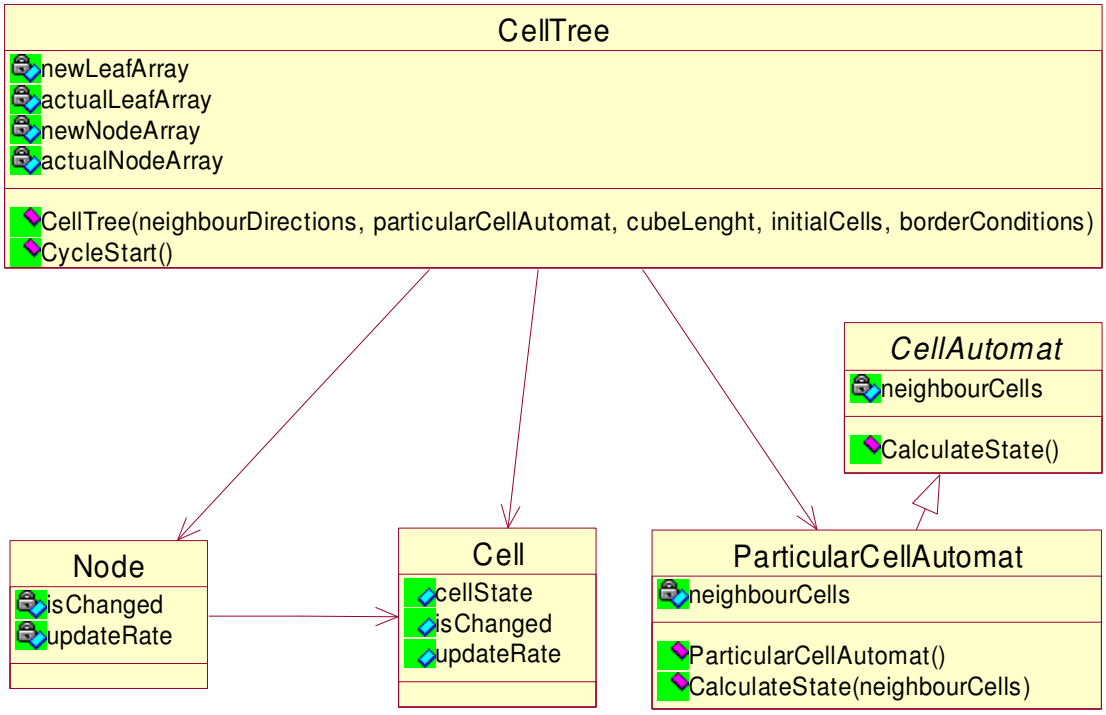
Modul Controller



Beschreibung

Wie hier ersichtlich ist, übernimmt der Controller eine zentrale Position in der Klassenhierarchie. Er steuert den Programmablauf, verarbeitet Benutzereingaben und regelt das Timing der Visualisierung.

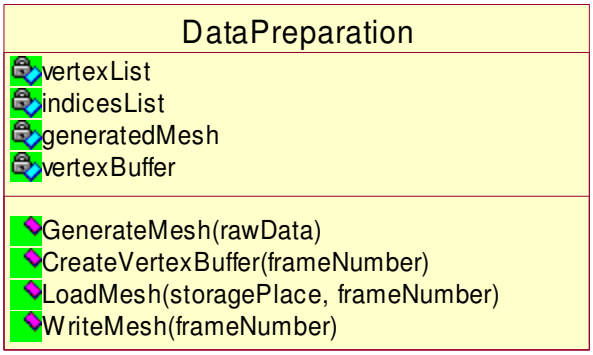
Modul Rohdaten berechnung



Beschreibung

Dieses Modul verfügt über eine Datenhaltungsstruktur, die jeweils das aktuelle, sowie das neue Frame eines Zyklus beinhaltet sind. Diese Struktur wird jeweils beim Erstellen des CellTrees automatisch initialisiert. Mit der Methode CycleStart() wird ein Berechnungsvorgang eines einzelnen Zyklus gestartet. Die Berechnungen der Zustände der Zellen werden durch den ParticualCellAutomat durchgeführt.

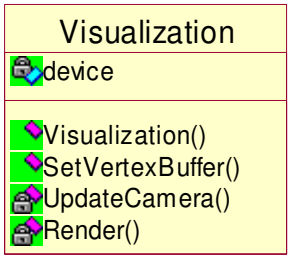
Modul Daten aufbereitung



Beschreibung

Die Datenaufbereitung übernimmt das aktuelle Frame der Datenhaltung und bereitet es für die grafische Visualisierung vor und leitet es an das Modul Visualisierung weiter.

Modul Visualisierung



Beschreibung

Dieses Modul ist für die Visualisierung der generierten Frames verantwortlich. Ausserdem übernimmt es die Steuerungen der Kameraführung, sowie einzelnen Parametrisierungen der Grafikkarte.

4. Sequenzdiagramme

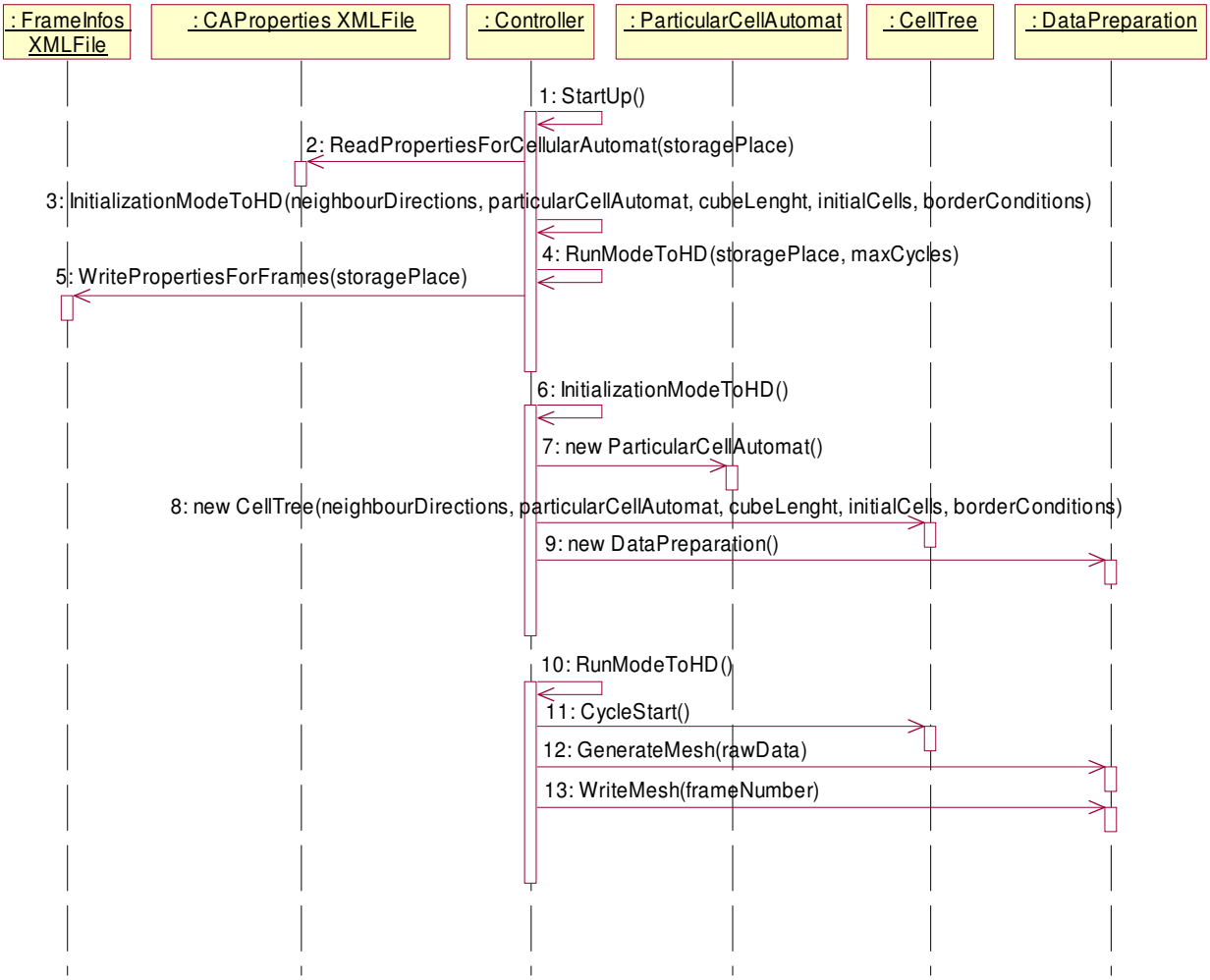
Hier werden die Rohdaten produziert, diese anschliessend **Modus „OnTheFly“** grafisch aufbereitet und danach visualisiert.

Bemerkung

Es muss darauf geachtet werden, das das Limit des Arbeitsspeichers nicht allzu schnell überschritten wird, denn durch den teilweise parallelen Ablauf der einzelnen Module kann es zu Speicherengpässen kommen.

Modus „SaveToHD“

In diesem Modus werden zuerst die Rohdaten berechnet, diese anschliessend für die grafische Ausgabe aufbereitet, jedoch dann nicht auf dem Bildschirm angezeigt, sondern für eine spätere Nutzung inklusive der relevanten Informationen auf der Harddisk abgespeichert.



Spezielles

Da es sich hier um ein „Vorproduzieren“ der Frames handelt, wird mehr Arbeitsspeicher und Rechenleistung zur Verfügung stehen, so dass grössere Lebensräume der zellulären Automaten berechenbar werden.

Hier werden die zu einem früheren Zeitpunkt vorproduzierten Frames von der Harddisk eingelesen und grafisch visualisiert.

Modus „LoadFromHD“