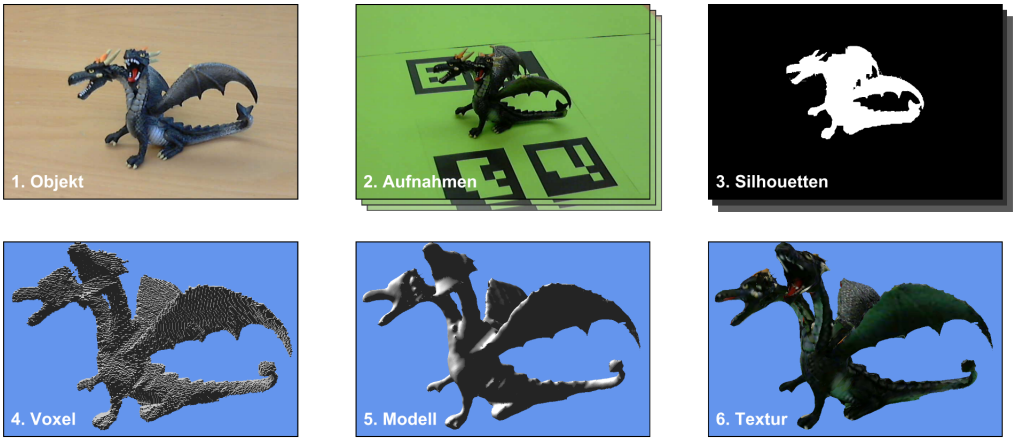
Bachelor-Thesis

Eclipse Entwicklungsumgebung für MicroCore

Benjamin Neukom

August 2015



Betreuer: Carlo Nicola

Inhaltsverzeichnis

In dieser Arbeit wird beschrieben, wie eine IDE für MicroCore mittels Eclipse implementiert werden kann.

1. Einleitung

Todo Copy from Forth MDT

2. Eclipse Platform

In diesem Kapitel wird gezeigt, was Eclipse für Möglichkeiten anbietet, um eine moderne Entwicklungsumgebung zu implementieren. Diese verschiedenen Möglichkeiten werden Verglichen und die Vor- und Nachteile aufgezeigt. Es werden auch die wichtigsten Eclipse Features, welche für das Entwickeln von Eclipse Rich Client Platform (RCP) Applikationen benötigt werden, beschrieben.

2.1. Eclipse als Platform

Eclipse RCP bietet eine Basis um beliebige (nicht zwingendermassen Entwicklungsumgebungen) Betriebssystem unabhängige Applikationen zu entwickeln. Es bietet Mechanismen, wie Plugins und Extension Points, um modulares programmieren zu unterstützen und vereinfachen. Auch bietet das Framework Features, wie das Konzept von Views und Editoren und vielem mehr, welche häufig in Applikationen gebraucht werden.

2.2. Plugins

Eclipse Applikationen nützen eine auf der OSGi Speizifikation basierten Runtime, eine Komponente in dieser Runtime ist ein Plugin. Eine Eclipse RCP Applikation besteht also aus einer Ansammlung von Plugins. Ein Eclipse Plugin ist ein Modul, welches gegen aussen ein API und Extension Points anbieten kann.

2.3. Extension Points

Um Plugins erweitern zu können, bietet Eclipse das Konzept von Extension Points an. Über Extension Points können Plugins ein bestimmte Funktionalität anbieten, welches von anderen Plugins aufgerufen werden kann.
So existiert zum Beispiel ein Plugin, welches einen Extension Point für Views definiert. Ein anderes Plugin kann über diesen Extension Point, deklarativ in einem XML File eine neue View erstellen. [1]

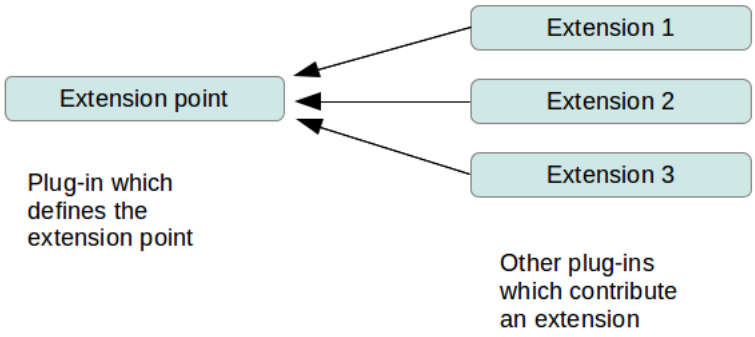


Abbildung 2.1.: Ein Plugin, welches einen Extension Point anbietet. Andere Plugins können

Es ist auch möglich eigene Extension Points zu definieren, falls ein eigens Plugin für andere Entwickler offen stehen soll für Erweiterungen.

2.4. Eclipse basierte MCore Entwicklungsumgebung

Eclipse als Grundlage für eine Entwicklungsumgebung zu verwenden eignet sich besonders gut, da Eclipse schon einiges an Funktionalität für eine IDE zur Verfügung stellt und schon einige Entwicklungsumgebungen mit Eclipse RCP entwickelt wurden. Auch existieren einige Tools, auf welche ich noch genauer eingehen werde, wie Xtext und DLTK, welche das entwickeln einer Entwicklungsumgebung weiter vereinfachen.

2.4.1. JDT

Eine Möglichkeit die MCore Entwicklungsumgebung zu implementieren, wäre über normalen Features (Plugins und Extension Points) von dem Eclipse Java Development Tools (JDT). Dies bringt aber einige Nachteile mit sich. Es müssten sehr viel von Hand

2.4.2. Xtext

XText ist ein Framework, welches es erleichtert eine auf Eclipse basierte Entwicklungsumgebungen zu programmieren. Es ermöglicht auf schnelle Weise ein Grundgerüst einer IDE mit Features wie:

- Ein Editor mit Syntax Highligting
- Code Completion
- Compiler Integration
- Ein Java-basierter Debugger

zu generieren. [2] Es muss lediglich eine Grammatik für die Sprache definiert werden. Der grosse Nachteil ist, dass C, inklusive Preprozessor, zu parsen sehr schwierig ist und auch mit Xtext nicht einfach zu implementieren ist.

2.4.3. DLTK

Das Dynamic Language Toolkit ist ein weiteres Framework, welches Grundgerüste für Entwicklungsumgebungen generiert. Ursprünglicherweise war das Framework nur für dynamische Sprachen geignet, es können aber auch für statische Sprachen verwendet werden. Die D

2.4.4. Eclipse CDT

Das Eclipse C-Development Tools (CDT) ist eine Eclipse Distribution mit Sprach Unterstützung für C und C++. Das CDT bietet alle Features welche man von einer Entwicklungsumgebung erwartet und stellt Extension Points zur Verfügung um diese für eine eigene Entwicklungsumgebung zu gebrauchen. So kann man mit relativ wenig Aufwand einen neues Compiler Backend in die Entwicklungsumgebung einbinden, welche den C Code kompiliert.

2.4.5. Verwendung für MCore Eclipse

Ich habe mich dazu entschieden, das Eclipse CDT als Target Platform zu wählen. Ich kann somit alle Features welche das Eclipse CDT zur Verfügung stellt brauchen und einfach mögliche Anpassungen machen. Die Frameworks, welche ein Grundgerüst einer Entwicklungsumgebung generieren funktionieren leider für C nicht vollständig und sind somit keine guten Alternativen.

3. Compiler Integration

3.1. Integration mittels CDT

3.1.1. CDT Extension Points

3.2. Error Parsing

4. Forth Kommunikation

4.1. Process Kommunikation

4.1.1. GDB/MI-Commands

4.1.2. Direkte Kommunikation mit dem Target

4.2. Implementierungs Details

4.2.1. Kommunikation mittels Commands

4.2.2. Forth Output Parsing

4.2.3. Await auf Resultate

5. Debugger

5.1. Breakpoints in C

5.2. Konsolen basierter Debugger

5.3. Forth Debugger

5.3.1. Neue Debugger Aktionen

Jump Action

Over Action

5.3.2. Stack View

5.3.3. Memory View

5.4. C Debugger

6. Optimierungen

6.1. Optimierungen

6.1.1. Peephole Optimierung

A. Literaturverzeichnis

[1] Vogella. http://www.vogella.com/tutorials/EclipseExtensionPoint/article.html, 2013.

[2] Xtext. https://eclipse.org/Xtext/, 2013.

B. Abbildungsverzeichnis

C. Tabellenverzeichnis

D. Ehrlichkeitserklärung

Hiermit bestätigen die Autoren, diese Arbeit ohne fremde Hilfe und unter Einhaltung der gebotenen Regeln erstellt zu haben.

Benjamin Neukom

Ort, Datum Unterschrift