

KickOffTUG Diagnosis

Weiterentwicklung des bestehenden Diagnosesystems zur Verbesserung der Usability

Máté Wolfram

March 22, 2009

Contents

1	Ausgangssituation	1
1.1	Altes Diagnose-GUI	2
2	Ziele	2
3	Projektschwerpunkte	2
3.1	Verwaltung von Datenquellen	2
3.1.1	ContextEntry	3
3.1.2	ContextDataSource	3
3.2	Integration in bestehendes GUI	3
3.2.1	DatasourcesTreePanel	3
3.2.2	DatasourcesPopupMenu	3
3.2.3	Tabs zur parallelen Verwaltung mehrerer Spiele	4
4	Ergebnisse	4

1 Ausgangssituation

Der erschwerte Zugang zu Diagnoseinformationen von simulierten Matches war neben der Forderung nach ausgefeilteren Diagnosemöglichkeiten wohl der Hauptmotivator für diese Projektarbeit. Aufgrund des Einsatzes von Simulationsservern waren Matchdaten und zugehörige Debuginformationen fast ausschließlich online verfügbar und mussten manuell heruntergeladen werden. Durch diesen langwierigen Vorgang ging neben wertvoller Entwicklungszeit auch der Bezug zum Originalmatch verloren, was auch die interne Weiterleitung entdeckter Fehler behinderte.

1.1 Altes Diagnose-GUI

Das Diagnose-GUI konnte vor seiner Überarbeitung lediglich mit lokalen Dateien umgehen. Eine Ausweitung auf andere Arten von Ressourcen war nicht geplant, was auch die Integration einer neuen Sichtweise auf Daten und deren Quellen (beschrieben in 3.1) erschwerte. Die einzelnen Anzeigeelemente (Frames für verschiedene Diagnoseinformationen, Gesamlayout der Anwendung) boten sich aber für eine Wiederverwendung an.

2 Ziele

Hauptziel dieses Projekts war es, den Vorgang der Überprüfung simulierter Spiele zu beschleunigen und vor allem angenehmer zu gestalten, um die Motivation zur Verwendung des Diagnosesystems zu erhöhen. Durch die Aufrechterhaltung des Bezugs zwischen lokal betrachtetem und zentral auf einem der Simulationsserver liegendem Spiel sollte die teaminterne Kommunikation entdeckter Fehler erleichtert werden. Geplant war auch die Einführung einer Art URL für Spiele, die es ermöglichen sollte, ein Teammitglied auf eine bestimmte Szene zu verweisen.

Die Datenquellenverwaltung (beschrieben in 3.1) sollte möglichst erweiterbar bleiben. Es muss in Zukunft die Interpretation verschiedenster Quellen möglich sein, sowohl bezogen auf Verbindungsprotokoll als auch auf zugrundeliegende Struktur der Daten.

Auch waren neue, ausgefeiltere Diagnosemöglichkeiten erwünscht, die es z.B. auch ermöglichen sollten Algorithmen dynamisch zu überprüfen. Hierbei sollte die Berechnungsgrundlage graphisch dargestellt und bei jeder Veränderung des WorldState neu berechnet werden.

Trotz all der Anforderungen war es wichtig, bereits vor der German Open 2009 eine funktionierende Lösung bieten zu können.

3 Projektschwerpunkte

Das Gesamtprojekt soll im Folgenden in zwei Schwerpunkte gegliedert werden. Die Datenquellenverwaltung (Kapitel 3.1) befasst sich mit dem eigenständigen Modul, das sich um die Beschaffung, Verwaltung und Bereitstellung von Ressourcen sowie um die Interpretation von Kontexten (in unserem Fall simulierten Spielen) kümmert. In Kapitel 3.2 - Integration in bestehendes GUI wird der Einbau der neuen Datenquellenverwaltung in das bestehende Diagnose-GUI erläutert.

3.1 Verwaltung von Datenquellen

Das grundlegende Konzept dieses Systems ist eine baumartige Struktur aus ContextEntries (3.1.1) und ContextDataSources (3.1.2), wobei erstere die Blätter und letztere die Knoten bilden. Diese baumartige Struktur hat es auch ermöglicht, das Design des bestehenden Dateibrowsers aus dem GUI zu übernehmen.

3.1.1 ContextEntry

Das Interface ContextEntry fasst Informationen aus dem gleichen Kontext (in unserem Fall z.B. ein simuliertes Spiel) zu einer Einheit zusammen. Alle verfügbaren Ressourcen werden nach aussen hin als Ganzes präsentiert. ContextEntry kennt den lokalen Pfad zu jeder Ressource, hält eine Liste an Commands, die notwendig sind um die benötigte Ressource zu erhalten und kennt auch deren Bedeutung (z.B. Diagnoseinformation von Spieler 4, Format: XML). Wozu diese Ressource gebraucht werden kann, ist hier absichtlich nicht definiert.

3.1.2 ContextDataSource

Die abstrakte Klasse ContextDataSource wird verwendet, um neue ContextDataSources oder ContextEntries zu generieren. Dazu können ihre Subklassen Ordnerstrukturen auf entfernten Servern parsen, Konfigurationsdateien einlesen oder lokale Ordner durchstöbern. Aus so gewonnenen Informationen wird die bereits im Vorfeld erwähnte baumartige Struktur aufgebaut.

3.2 Integration in bestehendes GUI

Allgemein wären an dieser Stelle wohl zwei Haupterkenntnisse zu nennen. Einerseits war es etwas mühsam, das alte Konzept eines Dateibrowsers gegen die neue Datenquellenverwaltung auszutauschen, da hier doch grundlegende Unterschiede vorherrschen, die eine weitgehende Neuimplementierung der Navigation erforderten, andererseits hingegen funktionierte die Trennung zwischen Diagnoseframes und den Aktionen, die nötig waren um diese zu erstellen, erstaunlich gut. So mussten im Rahmen der Integration der neuen Datenquellenverwaltung lediglich die Aktionen angepasst werden. Es wurden im Laufe der Zeit zwar auch Frames angepasst, aber ausschließlich um neue Funktionalität hinzuzufügen.

3.2.1 DatasourcesTreePanel

Das DatasourcesTreePanel ersetzt das alte FileSystemTreePanel und bietet so die Möglichkeiten der Datenquellenverwaltung direkt im GUI an. ContextEntries bzw. ContextDataSources konnten dank ihrer baumartigen Struktur besonders intuitiv in einem Panel dieser Art dargestellt werden. Zur Erhöhung der Performance werden Knoten on-the-fly abgearbeitet, d.h. erst bei Öffnung eines Knotens erhält der zugehörige ContextDataSource den Befehl, die darunterliegende Struktur aufzubauen.

3.2.2 DatasourcesPopupMenu

Die Komplexität des Popup-Menüs stieg nach dem Wechsel von Datei- auf Datenquellenansicht unweigerlich. So entstand eine eigene Klasse, die dessen neue Aufgaben kapseln sollte. Besonders hervorzuheben ist der Menüpunkt "Open", der für alle Operationen zuständig ist, die Frames erzeugen und mit Daten aus dem ContextEntry versorgen. All diese Operationen wissen, welche

Ressourcen sie aufgrund ihrer Implementierung benötigen werden, so kann auch im Menüpunkt “Open” entschieden werden, welche Untermenüs bei einem bestimmten ContextEntry angeboten werden sollen und welche nicht, je nach Verfügbarkeit.

3.2.3 Tabs zur parallelen Verwaltung mehrerer Spiele

Um die parallele Betrachtung mehrerer Spiele zu ermöglichen, wurde eine Art Tabbed Browsing eingeführt. Ein Tab ist direkt assoziiert mit einem ContextEntry, ein DatasourcesPopupMenu ist auf der Desktopoberfläche innerhalb des Tabs verfügbar und bietet die selben Funktionen wie im DatasourcesTreePanel. Wenn ein neues Diagnoseelement angefordert wird, erkennt das System automatisch, ob es sich um ein bereits geöffnetes Spiel handelt. In diesem Fall wird das neue Diagnoseframe im passenden Tab eingefügt, sonst wird erst ein neuer erstellt und registriert.

4 Ergebnisse

Die besten Erfolgsindikatoren bei einem Projekt zur Verbesserung von Usability sind wohl die Reaktionen der User selbst. Da das Diagnose-GUI nun teamintern immer öfter genutzt wird, kann davon ausgegangen werden, dass das Projekt sein Ziel keineswegs verfehlt hat und es auch in Zukunft die Analyse simulierter Spiele und die effiziente Kommunikation unter Teammitgliedern fördern wird.