Projektbewertung

Inhalt

Projektbeschreibung

Das SpaceBrowsing-System basiert auf einem Server-System, einem Client sowie einer eigens entwickelten 3D-Beschreibungssprache (SBL), basierend auf XML.
Der in C++entwickelte Client erhält vom Server per HTTP ein SBL-Dokument, welches clientseitig geparst und als 3D-Raum dargestellt wird. Über eine entsprechende Server-Komponente kann der Client Informationen über die Position des Avatars an alle anderen am Server angemeldeten Clients verschicken. Somit sieht der Benutzer die anderen Clients in Form ihrer Avatare.
Die Avatatar-Informationen werden per TCP über einen gesonderten Socket an den Server und wiederum an die anderen Clients übertragen.

Team-Mitglieder

Sebastian Durzynski
Oliver Elias
Christian Fuchs
Oliver Jacob

Bewertung der Programmiersprachen

- **C++**
  Für den Client wurde die Programmiersprache C++ genutzt. Dies ist deutlich unkomfortabler als die Entwicklung mit z.B. Java, da Hilfsmittel wie Refactoring-Werkzeuge in der verwendeten IDE quasi nicht vorhanden waren.
  Mit der Verwendung von „shared pointers“, opaquen Pointern und weiteren Richtlinien haben wir versucht, möglichst professionell in C++ zu entwickeln, was nach einer längeren Einarbeitungszeit viel Spaß gemacht hat.
  Die Performance des fertigen Produkts bestätigt uns in der Entscheidung, auf diese Sprache zu setzen.

- **Java**
  Bei der Entwicklung des Servers in Java wurde uns bewusst, wie viel Zusatzarbeit wir uns durch die Wahl von C++ beim Client gemacht haben. Bei Java sind uns keine Überraschungen begegnet, bis auf einige kleinere Probleme bei der Arbeit mit Threads.

Bewertung der IDEs

- **Eclipse CDT**
  Wie schon im Abschnitt über C++ erwähnt, gab es für die Arbeit mit Eclipse CDT (C++) kaum Hilfsmittel, wie zum Beispiel Refactoring. Weitere Probleme waren die unvollständige Code-Completion bei Verwendung von opaquen Pointern, was die Arbeitsgeschwindigkeit etwas verlangsamte. Insgesamt war die Umgebung stets zuverlässig.

- **XCode (Entwicklung unter MacOS X)**
  Im Gegensatz zu Eclipse CDT war die Arbeit mit XCode insgesamt angenehmer. Der Debug-Modus lief von Anfang an ohne Konfigurationsaufwand und die Geschwindigkeit der Entwicklungsumgebung selbst ist deutlich schneller.

Bewertung peripherer Technologien

- **SVN**
  Ohne SVN wäre die Zusammenarbeit im Team kaum möglich gewesen. Durch die Einrichtung von entsprechenden „hook-Skripten“ konnten wir den Server, auf dem das SVN lief so konfigurieren, dass die Dokumentation (des C++-Projekts) automatisch erzeugt wird.

- **Trac**
  Über das Projektmanagementsystem „Trac“ konnten wir offene Arbeitspakete übersichtlich verwalten und hatten so stets einen Überblick über den aktuellen Projektstatus.
  Positiv zu erwähnen ist ebenfalls die Einbindung von SVN im Trac-System.

Bewertung verwendeter Frameworks und Bibliotheken

- **Qt**
  Qt ist ein sehr umfangreiches Framework, welches nicht nur für Oberflächenprogrammierung geeignet ist. Das Projekt ist sehr gut dokumentiert und stabil, weshalb keine Probleme bei der Verwendung von Qt aufgetreten sind. Das Signal-Slot-Prinzip hat uns bei der Event-gesteuerten Entwicklung in C++ stark unterstützt.
- **AspectJ**
  AspectJ wurde erst zu einem sehr späten Zeitpunkt von uns verwendet um eine zentralisierte Ausnahmebehandlung in der Serverkomponente zu erreichen. Auch hier gab es keine Probleme bei der Verwendung.
- **Boost**

Im Laufe des Projekts ist die Arbeit mit Rohzeigen zu mühselig geworden, sodass wir uns für die Verwendung von „shared pointers“ entschieden haben. Die Boost-Bibliothek wird in diesem Zusammenhang sehr häufig verwendet und hat uns gute Dienste geleistet.

- **OpenGL**
  Für die Darstellung der 3D-Räume im Client haben wir OpenGL genutzt. OpenGL ist nicht objektorientiert und recht einfach zu verwenden. Schwierigkeiten gab es nur bei der Planung des Szenengraphen mit den entsprechenden mathematischen Transformationen.

Bewertung verwendeter Vorgehensmodelle

- **Extreme Programming**
  Wir wollten von Anfang an vermeiden, dass das Projekt komponentenweise aufgeteilt wird, um zu gewährleisten, dass sich jedes Team-Mitglied mit dem Gesamtprojekt identifizieren kann und einen Überblick über alle Komponenten hat.
  Deshalb haben wir möglichst oft zusammen entwickelt, was am Anfang zwar zu einem recht langsamen Vorankommen führte, aber im Laufe der Zeit ein effektives Entwickeln ermöglichte und vor allem gut durchdachten Code zur Folge hatte.
  Nachteilig an dieser Vorgehensweise ist, dass sehr oft gemeinsame Termine gefunden werden mussten.
  Es wurde stets darauf Wert gelegt, dass das System lauffähig ist, was sich positiv auf die Motivation der einzelnen Team-Mitglieder auswirkte.

Sonstiges

- **Entwicklung einer eigenen 3D-Beschreibungssprache (SBL)**
  Die Entwicklung der eigenen 3D-Beschreibungssprache samt Parser war durch das Entwickeln eigener 3D-Räume mit selbst erdachten Elementen sehr spannend; durch die Vielzahl der Elemente aber auch sehr aufwändig. Deshalb haben wir uns entschlossen für die Abgabeversion nur einen Teil der ursprünglich entwickelten Raum-Elemente zu implementieren.