GeoBashing Postmortem

Projektübersicht

GeoBashing ist ein Spiel mit den folgenden zwei Leitmotive: Sport (bzw. Betätigung im Freien) und Interaktion mit anderen Spielern. Es wird passiv oder aktiv gespielt. Der Spieler soll ermutigt werden zu kommunizieren und mit anderen zu interagieren. Die Entscheidungsfreiheit des Spielers soll möglichst groß sein. Es soll das Gefühl entstehen, Teil des Spiels zu sein. Um dem Spieler zusätzliche Motivation zu bieten werden Rollenspielelemente in Challenges mit eingebunden. Das Spielfeld für das Platzieren und Lösen von Challenges ist keine „virtuelle“ sondern die „echte“ Welt. Ein positionsabhängiges Kampfsystem mit physischer Flucht bringt zusätzliche Dynamik in das Spielgeschehen bzw. das alltägliche Leben.

Umgesetzt wurde das Spiel als Server-Client-Architektur. Serverseitig wurden REST-Webservices implementiert, welche vom Client über HTTP angesprochen werden. Die Serverkomponente basiert auf der freien Webserver-Implementierung XSP2 die unter Linux betrieben wird. Um die Applikationsdaten zu visualisieren, ist zusätzlich eine Webapplikation entwickelt worden. Clientseitig haben wir uns für JAVA ME als Basis entschieden. Um auf GPS-Daten zugreifen zu können, wurde die Java Location API benutzt. Die Referenzplattform der aktuellen Clientversion ist Nokia S60v3.

Das Projektteam hat aus drei Entwicklern bestanden, wobei die Aufgabenteilung wie folgt aussieht:

- Dieber Bernhard: Client-GUI, Controller
- Grassauer Thomas: Backend, Web-Applikation
- Mayring Jakob: Client-Netzwerklayer, Server-Konfiguration

Alle drei Entwickler sind Vollzeit berufstätig. In einer Kernentwicklungsphase von einer Woche haben wir die erste lauffähige Version des Servers und des Clients entwickelt. Danach gab es Treffen in unregelmäßigen Abständen. Bei diesen Treffen waren nicht immer alle Teammitglieder anwesend. Um dennoch effektiv arbeiten zu können, haben wir von Anfang an Subversion zur Source-Code- und Dokument-Verwaltung eingesetzt. Gegen Ende des Projekts schlossen wir das Projekt in mehreren intensiven Arbeitstagen (und -nächten) ab. Der Gesamtentwicklungszeitraum betrug etwas mehr als einen Monat mit einem Gesamtstundenaufwand von ca. 300 Stunden.

Was hat funktioniert?

- Subversion
- Entwicklungstools
- Zusammenarbeit im Team
- Flexibilität im Team
- Einsatz von C# im Backend
- Geplante Features (fast) komplett realisiert
- Gute Aufgabenteilung
- Ausgezeichnete Unterstützung durch die UNI Klagenfurt

Eine gute Entscheidung war es, von Beginn an auf die Client-Server-Architektur zu setzen. Das Beschränken auf nur eine Referenzplattform hat sich als großer Vorteil herausgestellt. Vor Projektstart war ein zusätzlicher Client für Windows Mobile geplant. Die drastische Reduktion der umgesetzten Features war gut abgeschätzt. Ein größeres Featureset, hätte zu einer deutlich höheren Entwicklungszeit geführt. Durch die Reduktion konnten sowohl die Deadline eingehalten, als auch die funktionalen Vorgaben erfüllt werden. Soweit wie möglich auf bereits vertraute Technologien zurückzugreifen, hat einen schnellen Einstieg in das Projekt ermöglicht. Der Einsatz von Versionskontrolle unterstützte das verteilte Arbeiten. Vertrautheit im Team sowie zeitliche Flexibilität der Mitglieder war ein essentieller Erfolgsfaktor. Ebenso hat die Aufgabenteilung nach Kompetenzen den Zeitverlust durch Einarbeitung vermindert. Feedback zum Projektvorhaben sowie die zur Verfügung gestellte Hardware durch die Universität waren eine große Hilfe.

Was hat nicht funktioniert?

- Java ME
- Plattformunabhängigkeit von Java
- Fehlende Debugging-Möglichkeiten
- Fehlende Designarbeit vor Projektstart
- Regelmäßiges Arbeiten
- Arbeitsaufwand falsch eingeschätzt
- Projektmanagement auf Micro-Management-Ebene

Java ME stellte uns vor Schwierigkeiten bezüglich Designbarkeit von Anwendungen, sowohl visuell als auch softwaretechnisch. Fehlerhafte Implementierung der JVM auf Symbian OS führte zu hohem Energieverbrauch. Nur Nokia hat eine vollständig Implementierung der Java ME API, wodurch eine Portierung auf Windows Mobile nicht möglich war. Nichtdeterministisches Verhalten von Java ME und nicht dokumentierte, kryptische Fehlermeldungen der Java Implementierung bedingten einen hohen Aufwand in der Fehlerfindung und –korrektur. Der fehlende Zugriff auf Standard-Out sowie fehlende Debugging-Unterstützung verstärkten diesen Effekt. Aus zeitlichen Gründen konnte kein vollständiges Applikationsdesign erstellt werden. Dies führte im weiteren Verlauf zu wiederkehrendem, erhöhtem Korrekturaufwand. Aufgrund der zeitlichen Einschränkungen der Teammitglieder war ein geordnetes Arbeiten nicht möglich. Die geplanten 150 Stunden wurden um 100 Prozent überschritten. Die sehr knappe Deadline ließ uns den Fehler, Projektmanagement nicht zu betreiben, begehen.

Schlussfolgerung

Wir haben mit sehr viel Leidenschaft und Einsatz ein Produkt entwickelt, hinter dem wir zu 100 Prozent stehen und dessen Weiterentwicklung bereits beschlossen ist. Die gelockerten zeitlichen Einschränkungen werden uns helfen, gemachte Fehler nicht zu wiederholen.