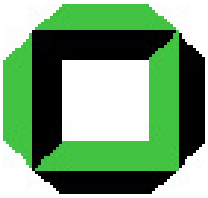

UNIVERSITÄT KARLSRUHE (TH)
Fakultät für Informatik
System Architecture Group
Frank Bellosa, Gerd Liefländer, Philipp Kupferschmied
Dominik Bruhn, Atanas Dimitrov,
Jonathan Dimond, Johannes Weiß

Basispraktikum Systemarchitektur - WS 2008/2009

Verteiltes Schiffe Versenken

1. Thematik

Im Folgenden ist eine lauffähige Variante des allgemein bekannten Spieles *Schiffe versenken* zu erstellen. Besonders Augemerk soll hierbei zum einen auf der Programmierung einer graphischen Oberfläche zur Bedienung und Visualisierung des Spielverlaufs liegen, zum anderen auf der Implementation einer Netzwerkschnittstelle, die das gemeinsame Spielen auf vernetzten Computern mittels TCP/IP ermöglicht.
Der Spielverlauf soll dahingehend vom bekannten verändert werden, daß das Spiel nicht rundenbasiert sondern parallel durchgeführt wird. Es gilt dabei, daß die Anzahl der von den einzelnen Spielern abgefeuerten Schüsse nur maximal um einen konstanten einzustellenden Wert differieren darf, so dass man nicht beliebig viele Felder hintereinander befeuern kann, ohne dass der Kontrahent agiert. Einige Schüsse nacheinander sollen somit allerdings möglich sein.

2 Implementierung und Visualisierung

2.1 Anforderungen

- Das Spiel soll von zwei Spielern an verschiedenen Rechnern mit- bzw. gegeneinander spielbar sein.
- Aus Mistrauen gegenüber der anderen Seite, soll jeder Spieler die Informationen über sein eigenes Spielfeld (die von ihm gesetzten Schiffe) für sich behalten und nur auf reguläre Nachfrage (wenn z.B. auf ein Feld gefeuert wurde) preisgeben.
- Die Steuerung des Spieles soll so weit wie möglich mittels Mauseingaben gelöst werden!

2.2 Vorüberlegungen

Im Vorfeld zur Implementation gilt es, einige Schlüsselfragen zu stellen, um gezieltes Vorgehen zu ermöglichen:

- In welche sinnvollen Teilobjekte läßt sich das gesamte Projekt zerlegen?
- Welche Möglichkeiten bietet Java zur Kommunikation über ein bestehendes Netzwerk?
- Welche Art von (unterschiedlichen) Daten soll über das Netzwerk übertragen werden? Wie können diese codiert werden?
- Welche Teilbereiche müssen (sollen) eigenständig arbeiten und daher mit Hilfe von Threads implementiert werden?

- Zu welchen Zeitpunkten ist es notwendig, den Spielablauf zu synchronisieren und wie ist dies zu tun?
- Gibt es zu synchronisierende Ressourcen?

2.3 Implementierung

Die Implementation soll das gesamte Programm in sinnvolle Teilobjekte zerlegen. Die vorliegende Dokumentation[1] einer Lösung versucht sich weitgehend an die so genannte *Model-View-Controller* Architektur zu halten. *Model* hält hierbei stets die für den Ablauf relevanten Daten und stellt benötigte Zugriffsmethoden zur Verfügung, *View* zeichnet eine visuelle Repräsentation der gehaltenen Daten und *Controller* kümmert sich um Events und den allgemeinen Ablauf.
Zu erstellen sind die Klassen *Game* (Model für das Spiel, hält Daten über das Spielfeld, Spielernamen, Munition usw.), *View* (zeichnet in der dokumentierten Lösung mit Hilfe der Klassen Board, BoardField und ScoreBoard das Feld), der *Controller* und die Klasse *NetDaemon*, welche sich um die Kommunikation zwischen den verteilten Instanzen des Spiels kümmert.
Desweiteren sollte ein *MouseFieldAdapter* geschrieben werden, welcher das Interface MouseListener implementiert und zur Mauseingabe benötigte Sensibilität bietet.
Als Einstiegspunkt in die Implementation steht *OptionScreen* zur Verfügung, welcher im Controller die Methoden *setAction(...)* und *setInfo(...)* aufruft.

Hinweis: Mehr zum Thema Custom Networking finden Sie im gleichnamigen Java Tutorial[2]

3 Bemerkungen

Die beigefügte Dokumentation[3] einer Lösung ist gedacht, um ein besseres Verständnis der geforderten Aufteilung der einzelnen Komponenten und deren Einbettung in ein laufendes System zu gewährleisten. Sicherlich gibt es im Zuge dieses Ablaufs weitere (bessere?!) Lösungen. Eigene Lösungen, die alles Wesentliche berücksichtigen und sich an die gegebenen Punkte halten, sind willkommen.

4 Extras und Erweiterungen[4]

- Das Spiel könnte auf mehrere Mitspieler erweitert werden. Welche Anpassungen sind hierfür nötig?
- Die Ressource Munition könnte an einzelne Schiffe gebunden sein, so dass man im Vorfeld zu einem Schuss auswählen muß, von welchem Schiff dieser stammt. Getroffene Schiffe können nicht mehr schießen, was den Munitionsvorrat verringert.
- Schiffe könnten beweglich sein. Dies heißt unter anderem, dass Felder mehrfach beschossen werden müssen.

---

[1] Javadoc erstellt aus Musterlösung; im Anhang zu diesem Blatt
[2] Online only
[3] Siehe oben.
[4] Feat. G. Liefländer