PROGRAMMENTWICKLUNG FÜR MOBILE GERÄTE PROF. MEIXNER

MOBILE EXPERIENCE: HIT HANDY INFORMATION TERMINAL PROF. JOHN

SS 2006

FRANK GRUBER
NEPOMUK KARBACHER
PHILIPP MÜHLBAUER
CORNELIA-ANCA PAULNICI

INHALT

SPIELESTEUERUNG AUF EINEM PUBLIC-SCREEN

ÜBER BLUETOOTHHANDYS

Im Rahmen des Studienprojekts Programmentwicklung für mobile Geräte bzw. mobile experience: HIT/Handy Information Terminal hat unsere Gruppe die Aufgabe übernommen ein Spiel zu entwickeln und prototypisch umzusetzen. Dabei war die Voraussetzung, dass das Spiel Mobiltelefone über Bluetooth mit einem Public-Screen verbindet. Davon ausgehend haben wir verschiedene Ideen entwickelt. Im Laufe des Semesters wurden Team übergreifend Entscheidungen getroffen, die einen neuen Ansatz erforderlich gemacht haben. Dies hat zu dem Konzept geführt, das wir am Ende als einen spielbaren Prototypen realisiert haben.

IDEEN

Im folgenden Abschnitt sind einige Ideen beschrieben, die uns zur Aufgabenstellung eingefallen sind. Davon sind einige nur in Teilaspekten interessant oder schwer umsetzbar.

PONGCRACY

(PONG - DEMOCRACY - (CRAZY))

Umsetzung des Pong-Spiels in einer Multiplayer Variante. Die einzelnen Mitspieler werden möglichst gleichgewichtig auf zwei Teams aufgeteilt. Die Bewegung der Schläger ergibt sich demokratisch aus dem Mittel der einzelne Spielereingaben einer Mannschaft. Optional gibt es anspruchsvollere Spielvariationen durch Modifikation der Eingabelogik. So ändert sich z.B. die Zuordnung der Befehle <RAUF> und <RUNTER> bei jedem Schlag. Nachdem der Ball vom gegnerischen Schläger abgeprallt ist wird die neue Belegung auf den Displays der Mitspieler angezeigt. Die Mitspielerzahl ist theoretisch unbegrenzt.

SHOOT'N'RUN

Auf dem Public-Screen wird zunächst nur eine Spielfigur, ein Start- und ein Endpunkt sowie verschiedene Wegmarkierungen und Hot-Spots angezeigt. Es existieren jedoch keine Verbindungen zwischen diesen Punkten. Die Figur kann so den Ausgang nicht erreichen ohne in den Abgrund zu fallen. Durch geschicktes fotografieren von Formen und Strukturen mit starken Hell-Dunkel Kontrasten kann das Spielfeld ergänzt werden. Die Bilder vom Handy werden dabei analysiert (Kontrast, Farben, Formen) und in die Wege, Stege, Leitern, usw. übersetzt. Somit wird der Weg für die Spielfigur frei und der Ausgang kann erreicht werden.

BILD-ASSOZIATIONEN

Vom Handy aus lassen sich Bilder und Texte auf den Public Screen übertragen. Passanten haben die Möglichkeit eigene Beiträge hochzuladen und können sich dabei z.B. auf vorherige Beiträge anderer Nutzer beziehen.

BLUENOTE

Am Bluetooth Hotspot wird willigen Spielern ein Takt per Klicks im Gerätehörer übermittelt. Dem Benutzer ist es nun möglich eine Aufnahme zu starten. Jetzt kann er zum Takt Geräusche machen oder singen. Nach dem Beenden der Aufnahme wird diese an den Hotspot geschickt. Aus dem Bestand aller eingesendeten Aufnahmen wird eine Geräusch-Landschaft komponiert, die im Bereich um den Hotspot zu hören ist. Ein Download des Stücks als Klingelton ist ebenfalls möglich.

HEISSE LUFT

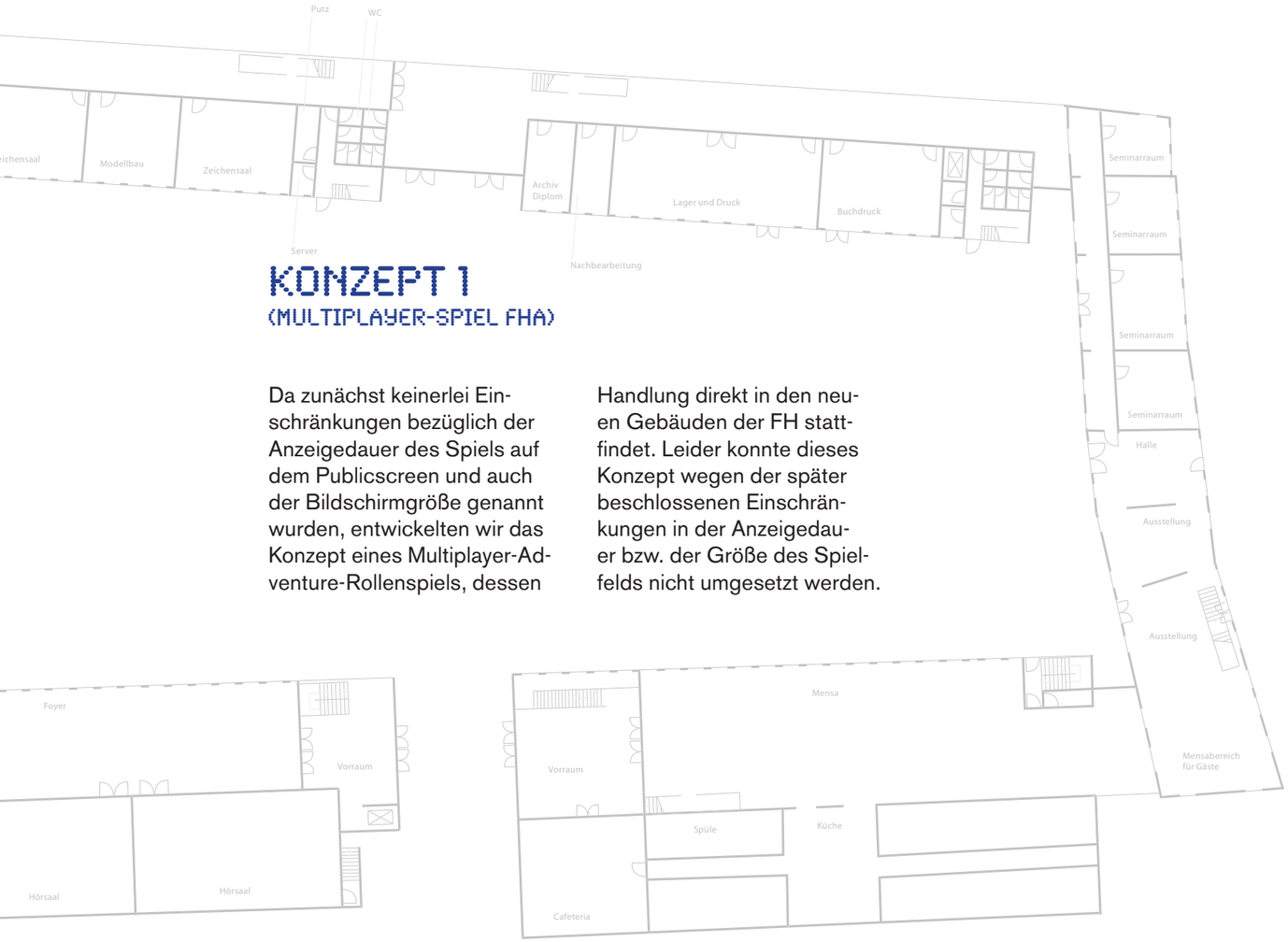
Der Spieler steuert ein Spiel indem er die Handy-Kamera auf eine bestimmte geometrische Form richtet, die an der Wand angebracht ist. Dies kann z.B. eine horizontale Linie sein. Durch Rotieren der Kamera ändert sich die Neigung der Linie im Bild und steuert so z.B. ein Flugzeug.

BLUTTOOTH

In einer Spielarena treten beliebig viele Mitspieler gegeneinander an. Nach einem einfachen Prinzip können sich die Spieler angreifen (hauen) bzw. Angriffe abwehren. Dies führt zu jeweiligem Punktabzug oder Gewinn. Die Arena wird dabei von oben betrachtet um die Übersicht zu behalten (ähnlich dem Spiel Bomberman). Die Spielfiguren können durch den Spieler individualisiert werden. Im Prinzip kämpft jeder gegen jeden. Durch jeweilige Telefonbucheinträge können sich Mannschaften formieren (oder das Punktesystem kann auf soziale Netzwerke über Telefonbucheinträge reagieren). Über das Triggern von Hot-Spots in der Arena oder andere Ereignisse (Zeit usw...) wird der Spielverlauf unterbrochen. Hier beginnt die alternative Spielphase. Auf dem Großdisplay wird den Spielern eine Aufgabe gestellt. Erst wenn sie diese absolviert haben dürfen sich die einzelnen Spieler wieder bewegen. Sie können dann noch gelähmte Spieler angreifen.

Möglichkeiten für Aufgaben:
- Ein Bild mit 50% Blau und 50% Rot fotografieren
- Möglichst schnelle einen Text eingeben
- Einen Ton in bestimmter Höhe in das Mikrophon singen
- Unter den Mitspielern jemanden finden der das gleiche Symbol wie man selbst auf dem Handydisplay angezeigt bekommt.

Die Lösung der Aufgaben wird noch durch zusätzliche Punkte belohnt.



KONZEPT 1
(MULTIPLAYER-SPIEL FHA)

Da zunächst keinerlei Einschränkungen bezüglich der Anzeigedauer des Spiels auf dem Publicscreen und auch der Bildschirmgröße genannt wurden, entwickelten wir das Konzept eines Multiplayer-Adventure-Rollenspiels, dessen Handlung direkt in den neuen Gebäuden der FH stattfindet. Leider konnte dieses Konzept wegen der später beschlossenen Einschränkungen in der Anzeigedauer bzw. der Größe des Spielfelds nicht umgesetzt werden.

FUNKTIONSWEISE

Auf dem Publicscreen wird eine Karte angezeigt, welche die Gebäude des Fachbereichs aus der Vogelperspektive zeigt.
Das Spiel wird auf dem Publicscreen und auf den Handydisplays der Benutzer gespielt.
Das Handy dient zur Steuerung der Figur auf dem Publicscreen.
Auf dem Handydisplay werden Informationen über den eigenen Charakter, Shops, Kämpfe, Interaktionen mit anderen Benutzern etc. dargestellt. Der Publicscreen gibt einen Überblick. Auf dem Handy wird alles dargestellt, was nur den eigenen Charakter betrifft.

KONZEPT 1 – MULTIPLAYER-SPIEL FHA

HANDLUNG

Das Spielfeld besteht aus den neuen Gebäuden der FH in U-Form, sowie etwas Landfläche drumherum. Es gibt betretbare und nicht betretbare Felder (Mauern, Bäume, Objekte etc.). Eine zuvor erstellte Spielfigur bewegt sich auf dem Feld und löst Aufgaben. Die Aufgaben sind soweit möglich FH-bezogen.
Das Spiel ist Rundenbasiert. Man bekommt pro Tag eine bestimmte Zahl an Schritten zugewiesen. Wenn man einen Tag nicht spielt, addieren sich die Schritte von maximal einem Tag dazu. Das bedeutet, man kann beim nächsten einloggen die doppelte Anzahl an Schritten machen. Ältere Schritte verfallen.

Ziel ist es, möglichst viele Punkte zu verdienen. Diese können folgendermaßen erworben werden:

Finden – Auf der Karte unsichtbar wird Geld verteilt, wenn man auf ein Feld mit Punkten geht, bekommt man den Betrag auf dem Handy angezeigt, auf dem Public-Screen wird über dem User ein Symbol eingeblendet.

Verdienen durch Aufgaben – die Professoren stellen oder die man auf dem Weg oder bestimmten Feldern findet.

Kämpfen – gegen andere User, wobei die Punkte transferiert werden, was der Eine verliert gewinnt der Andere.

Zum Geburtstag gibt's Extra-Punkte und Extra-Schritte.

Da man nur eine limitierte Anzahl an Schritten zur Verfügung hat, kann man seine Reichweite durch den Kauf verschiedener Items erhöhen. Diese Kosten Punkte (ECTS-Punkte?),

CHARAKTEREIGENSCHAFTEN

Der eigene Charakter kann durch verschiedene Gegenstände erweitert und aufgerüstet werden, welche entweder gekauft, im Rahmen von Spielen vergeben oder gefunden werden. Ein Charakter hat folgende Eigenschaften:

Nickname (frei wählbar)
Energiepotenzial (abhängig von: Essen, trinken)
Kraftpotenzial (abgängig von: Training oder Kauf von Energieriegeln)
Geld/Punkte (verdienbar durch: finden, Aufgaben erledigen, Spiele, Gewinnen von Kämpfen, oder im realen Leben durch den erwerb/verdienst von Gutscheinen mit Code (z. B. vom Prof. bei pünktlichem erscheinen in der Vorlesung oder für soziales Engagement (Sommerfest etc.))

Dabei wird der Charakter, ähnlich dem Tamagotchi, wenn er nicht am Spiel teilnimmt, auf dem Handy unterwegs weiter trainiert. Z. B. mit Spielchen auf dem Handy: Workout (Zahlenfolge, die immer schneller angezeigt wird nachtippen. Prinzip: Körperliche Arbeit für Kraftgewinn des Charakters)

ITEMS

Um im Spiel Vorteile gegenüber anderen Spielern zu haben, können verschiedene Items in speziellen auf dem Spielfeld verteilten Shops zugekauft werden, um den Charakter aufzurüsten. Dabei gibt es Items, die dem Charakter erhalten bleiben, bis es durch ein neues ersetzt wird (Waffen, Schild) – und Items, die nach einmaligem Gebrauch aufgebraucht sind (Gummistiefel, Energieriegel etc.). Items können ebenso gefunden und als Belohnung für gelöste Aufgaben vergeben werden.

DAUERHAFTE ITEMS

Waffen
Schild

TEMPORÄRE ITEMS

Speedschuhe/Sprungstiefel
Moped
Rollstuhl
Inlines
Bombe (Um sich durch Mauern zu sprengen)
GOD-Mode (Wie bei Doom, unverwundbar)

Der Preis bestimmt hier, wie sehr ein Item die Reichweite erhöht. So sind Speedschuhen bei denen für 2 Schritt einer gezählt wird günstiger als ein Moped bei dem 5 Schritt für einen gehen.

Man kann auf dem Spielfeld auch Löcher setzen die vorher gekauft werden müssen und auf der Karte unsichtbar sind. Fällt ein Spieler in das Loch eines anderen Spielers, bekommt der Fallensteller einen Betrag gutgeschrieben, welcher dem Reingefallenen abgezogen wird.

Beispielhafte Spielfiguren entnommen aus Gunbound

Die aktuelle Wetterlage in Augsburg kann über das Internet abgefragt werden und die Umgebung des Spiels ändern. Z. B. könnte bei Schnee, Regen, Eis oder Hitze die Zahl der Schritte, die man aussen verbraucht, verdoppelt werden, wenn man nicht bestimmte Items gekauft hat, welche einem eine Anpassung an das Wetter ermöglichen.

WETTERABHÄNGIGE ITEMS

Regenschirm
Gummistiefel
Sonnenschirm
Schweissband
Ski
Snowboard
Schneefräse
Boot
Rettungsring
Schlittschuhe
Schneeschuhe

MÖGLICHKEITEN DER INTERAKTION

OFFLINE

Man stellt seine Spielfigur auf einem Feld ab und ist nach dem ausloggen unsichtbar. Wenn ein anderer Spieler jedoch auf exakt das selbe Feld geht, sieht er den abgestellten User auf seinem Handy.

Er kann dem anderen User nun eine Nachricht hinterlassen, vorgefertigte Beschimpfungen hinterlassen oder angreifen und mit ihm kämpfen.

Das Spielfeld ist so gestaltet, dass es eine vielzahl von versteckmöglichkeiten für den User gibt. Beispielsweise zwischen Bäumen, an schwer erreichbaren Stellen am Spielfeldrand, in Räumen der FH und so weiter.

Kampf ist eine einmalige Entscheidung, der Kampf wird durch einen Algorithmus ausgeführt, der auf Kraft, Energie, Waffen und Schild beruht.

ONLINE

Beide Spieler sind online und spielen oder kämpfen miteinander.

Entweder wird Pong auf den Handys gegeneinander gespielt oder Zahlen werden auf dem Handy eingeblendet und müssen nachgetippt werden.

INTERAKTION ZWISCHEN ALLEN USERN

Ausserdem ist eine Interaktion unter allen Usern durch spezielle Aufgaben möglich. Das Spielgeschehen wird unterbrochen damit zum Beispiel auf jeweils zwei Handys ein Symbol und ein Code dargestellt werden. Man muss nun den Spieler im Raum finden, welcher das selbe Symbol auf dem Display hat und den Code dieses Spielers ins eigene Handy tippen. Die zwei Spieler, die das am schnellsten schaffen bekommen die meisten Punkte.

KONZEPT 1 – MULTIPLAYER-SPIEL FHA

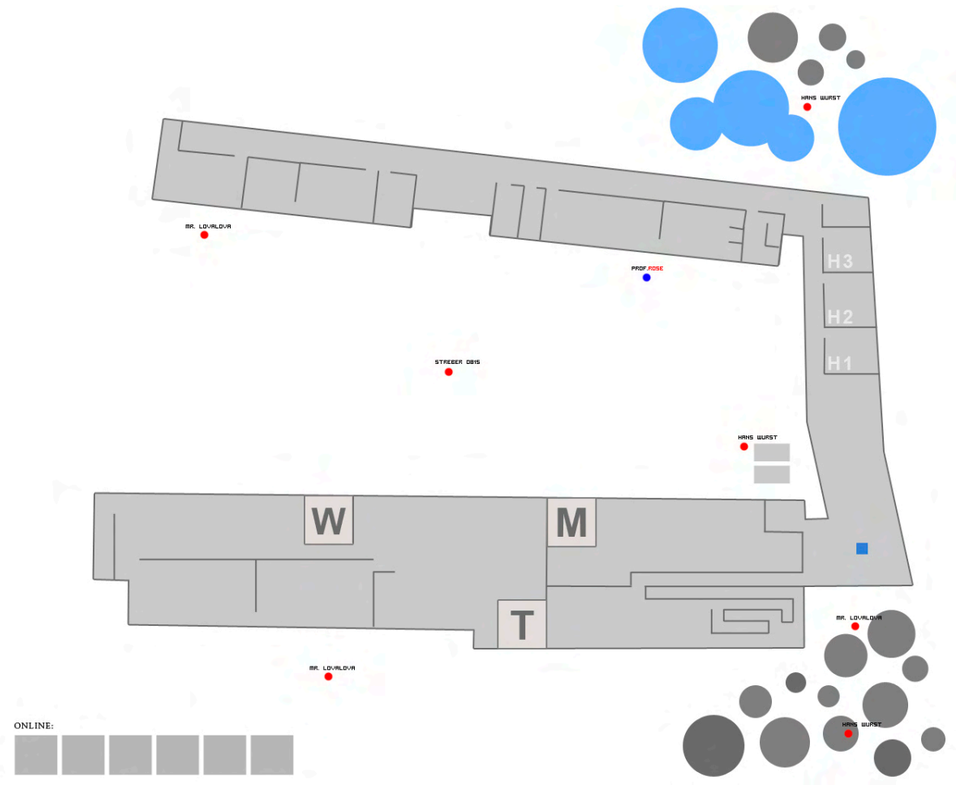
TECHNISCHE DATEN

Sicht auf dem Public-Screen ist die Vogelperspektive.

Auflösung 1280 x 1024 (hängt vom Budget der FH ab)
10 x 10 Pixel pro Feld

Der Grafikstil ist Reduziert, schlicht und passend zum neuen Gebäude.

Auf dem Beamer wird das gesamte Spielfeld gezeigt, auf den Handydisplay sieht man den eigenen Character inkl. Statusinformationen sowie Shops oder Kämpfe.



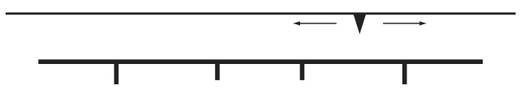
KONZEPT 2
(ARBEITSTITEL TUGWAR)

Da zwischenzeitlich die Anforderung aufkam alle Projekte möglichst gleichzeitig anzuzeigen, haben wir uns entschieden das Fullscreen-Konzept 1 aufzugeben. Nachdem keine Einigung bei der Aufteilung des Screens in Sicht war schlug unsere Gruppe eine horizontale Dreiteilung des Screens vor, bei der die Spielegruppe zugunsten des Fischeprojekts und der Infogruppe nur einen kleinen Streifen im unteren Bildbereich bekam. Die genauen Maße wurden später von der Info-Gruppe festgelegt.
Ausgehend von den Problemen des ersten Konzepts haben wir zunächst einige Vorgaben erarbeitet, die das neue Spiel erfüllen muss.
- fexibles Format
- schneller Ein- und Ausstieg
- rundenbasiert (Time-Slot kompatibel)
- u.U. beliebig viele Spieler
- kommunikativer Spielspaß

Basierend auf diesen Kriterien haben wir ein zweites Konzept entwickelt. Dabei basiert das Spielprinzip darauf, dass sich ein Wert auf einer Skala zwischen zwei festen Endwerten bewegt. Am Besten lässt sich dieses Konzept am Ablauf von Seilziehen erklären. Zwei Mannschaften treten gegeneinander an. Jede Runde dauert eine bestimmte Zeit. Am Seil ist ein Wimpel angebracht, der zu Beginn der Runde über einer Markierung hängt, die in der Mitte zwischen den beiden Teams liegt. Es gewinnt die Mannschaft, die den Wimpel des Seils über eine bestimmte Markierung gezogen hat oder diese am Ende der Runde auf der eigenen Seite hat. Dieses Prinzip lässt sich neben Seilziehen auf viele andere Szenarien übertragen.

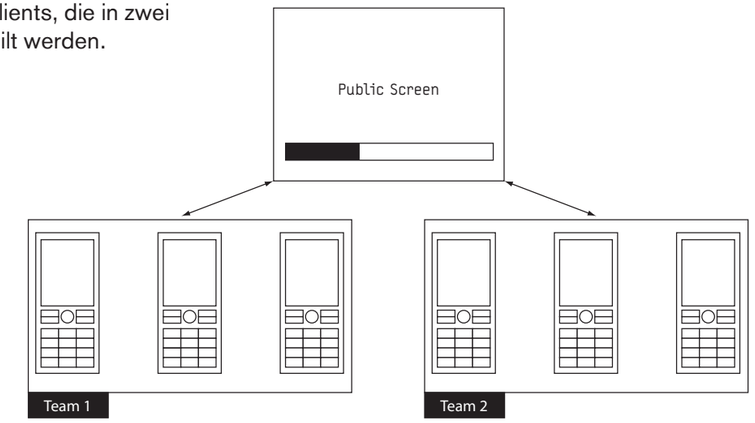
Start/Unentschieden

linke Mannschaft gewinnt
vor Ablauf der Zeit



rechte Mannschaft gewinnt
nach Ablauf der Zeit

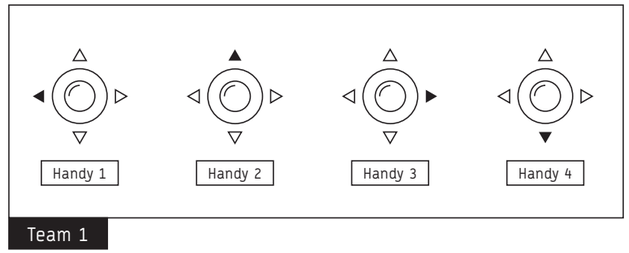
Der Aufbau für das Spiel besteht aus dem Public Screen und mehreren Clients, die in zwei verschiedene Teams aufgeteilt werden.



Die Zugkraft entsteht durch die Geschwindigkeit in der bestimmte Tasten der Handys in einer bestimmten Reihenfolge nacheinander von den Teammitgliedern gedrückt werden. Hierbei sind beliebige Regeln für die Tastenkombinationen möglich, von denen wir zwei Varianten ausgearbeitet haben.

1. STEUERUNGSVARIANTE

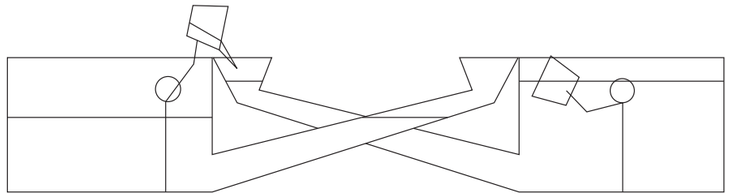
Jeder Spieler eines Teams drückt auf den Richtungstasten seines Handys in eine bestimmte Richtung, so dass die Eingaben zusammen eine Bewegungsfolge im Uhrzeigersinn ergeben. Jede vollständig eingegebene Runde löst eine Bewegung des Balance-Wertes zu Gunsten des entsprechenden Teams aus.



2. STEUERUNGSVARIANTE

Bei der zweiten Variante wird der Bewegungsimpuls durch die Eingabe des richtigen Ergebnisses einer Rechenaufgabe ausgelöst. Der erste Spieler gibt eine Zahl ein, die am Public Screen angezeigt wird. Daraufhin bekommt er in Verbindung mit dieser einen Operator und eine weitere Zahl als Operanden angezeigt. Der Operator kann dabei die Rechenmethode Addition, Subtraktion, Multiplikation oder Division sein. Das Ergebnis dieser Rechnung teilt er dem nächsten Spieler seines Teams mündlich mit. Der gibt das Ergebnis in sein Handy ein und bekommt wieder eine Rechenaufgabe gestellt usw. Die Teilnehmerzahl ist dabei theoretisch unbegrenzt und in unserem Fall nur durch die verwendete Technik limitiert. Der letzte Spieler in der Reihe schickt das Ergebnis seiner Aufgabe an den Server. Ist das Ergebnis richtig, wird ein Bewegungsimpuls für das Team registriert.

Neben Seilziehen sind viele andere Spielszenarien denkbar.
- Armdrücken
- Fingerhakln
- Erwürgen
- Ballspiele
- Wippen
- Ertränken

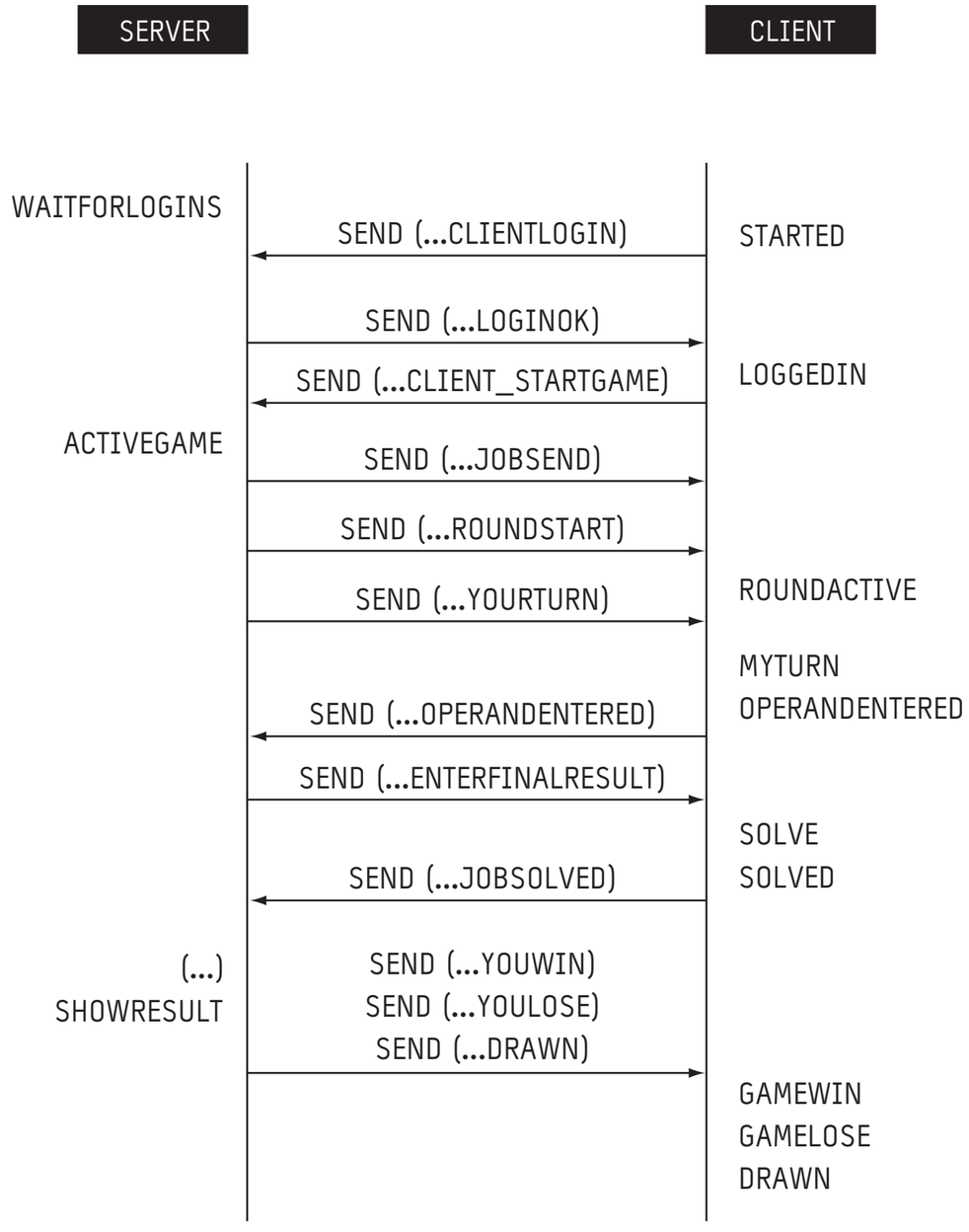


UMSETZUNG

Der entwickelte Prototyp baut auf dem technischen Rahmen auf, den eine andere Projektgruppe konzipiert und erstellt hat. Die vorgegebene Plug-In Struktur auf Client- und Server-Seite wurde auf Basis der zu Verfügung gestellten Beispiele implementiert. Für die Darstellung auf dem Public-Screen wurde eine Flash-Anwendung erstellt, die von dem Plug-In auf Server-Seite angesprochen und gesteuert wird. Durch die Konzentration aller Kommunikationsaufrufe zwischen Client- und Server auf wenige Methoden, wird die Funktionsweise des Anwendungs-Frameworks einfach verständlich und nachvollziehbar.

Client:
public void receive(ActionMessage message)
private void send(int actiontype, String[] param)
private void send(int actiontype)

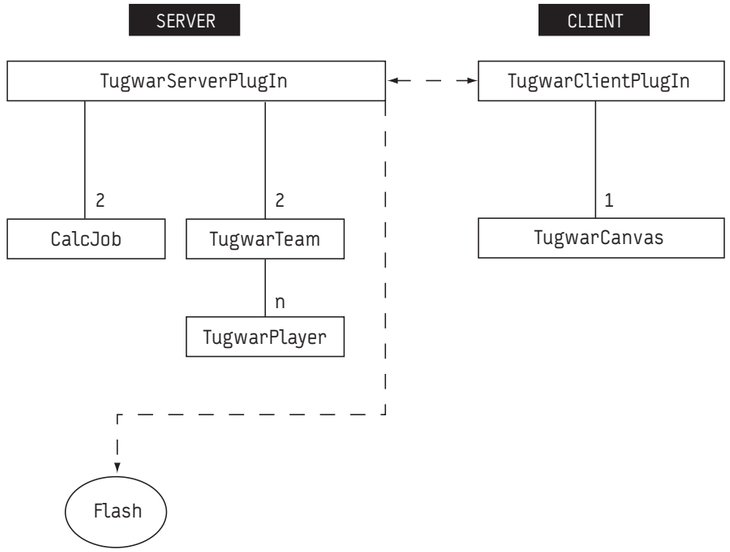
Server:
private boolean receive(ActionMessage message)
private void send(int action,String[] param,TugwarPlayer receiver)
private void send(int action,String[] param,EndPoint ep)
private void send(int action,TugwarPlayer receiver)
private void send(int action,EndPoint ep)



(die im folgenden verwendeten Klassennamen erklären sich durch den Arbeitstitel Tugwar)

CLIENT

Die Client Anwendung besteht aus der Plug-In Klasse TugwarClientPlugIn und der Klasse TugwarCanvas. TugwarClientPlugIn empfängt Nachrichten vom und sendet Nachrichten an den Server. Wir verwenden dabei ausschließlich Nachrichten vom Typ ActionMessage und spalten die weitere Verarbeitung anhand des ActionTypes auf. Die möglichen Nachrichtenarten sind in der Klasse TugwarActionTypes hinterlegt. In Abhängigkiet von verschiedenen Status, die die Client Anwendung haben kann, werden Nachrichten unterschiedlich verarbeitet. Vom aktuellen Status hängt auch ab, was auf dem Handy-Display zu sehen ist. Hierfür verwenden wir eine Klasse aus der Low-Level-API (Canvas) von der wir die Klasse TugwarCanvas ableiten. TugwarCanvas zeichnet die Screens und Eingabemasken auf den Bildschirm des mobilen Endgeräts und nimmt die Tastatureingaben entgegen.



SERVER

Auf Server-Seite umfasst unsere Anwendung die Klassen CalcJob, TugwarPlayer, TugwarServerPlugIn und TugwarTeam. Die Klasse CalcJob ist für die Generierung der benötigten Rechenaufgaben verantwortlich und überprüft auch deren Lösungsversuche. Um die Schwierigkeit der gestellten Aufgaben zu limitieren, werden nur Zahlen mit maximal zwei Stellen verwendet und die Rechenergebnisse der Teilaufgaben sind immer positiv und ganzzahlig. Bei der Lösung der Aufgaben wird während einer Spielrunde vermerkt, welche Teile der Aufgabe bereits bearbeitet und davon richtig gelöst wurden.
Die Klasse TugwarServerPlugin stellt die Hauptklasse des Spiels auf Server-Seite dar. Hier wird das Spielgeschehen verwaltet, Daten werden an die Clients verteilt und die Darstellung auf dem Public-Screen wird gesteuert. Genau wie die Clients nimmt auch das Server-Plug-In verschiedene Zustände (bzw. Status) an, die den aktuelle Spielverlauf abbilden bzw. steuern. Solange das Spiel nicht aktiv gestartet wurde, können sich Clients beim Spiel einloggen. Dann wird aus den Client-Informationen ein Objekt der Klasse TugwarPlayer erzeugt und einem der als TugwarTeam Array angelegten Teams (team[0],team[1]) zugeteilt. TugwarTeam stellt einen Container (ArrayList) zur Verfügung, zu dem komfortabel Spieler hinzugefügt oder entfernt werden können. Auch Methoden zur Abfrage welcher Spieler welchem Team angehört sind implementiert.
Während des Spiel durchlaufen beide Teams mehrere Spielrunden mit jeweils einer Rechenaufgabe. (Eine Rechenaufgabe wird durch den Konstruktor von CalcJob generiert bzw. durch den Aufruf von init()) Die Methode roundProgress sorgt dafür, dass ein Spieler nach dem anderen eine Rechenaufgabe zu lösen hat. Hierfür wird jeweils eine entsprechende Nachricht an den Client gesendet (send(TugwarActionTypes.YOURTURN,team[t].get(0));).
Um die Spieldauer zeitlich zu begrenzen, wird ein Timer verwendet, dem die innere Klasse TugTask übergeben wird. Nach dem Ablauf des Timers wird die Methode endGame() aufgerufen und damit das Spiel beendet. Nach dem Ablauf eines weiteren Timers der dafür sorgt, dass das Ergebnis des Spiels zu sehen ist, wird das Spiel über die Methode reset() neu gestartet.

FLASH

Die Anzeige des Spiels auf dem Public Screen erfolgt in einem Flash-Container. In diesem findet die Grafik-Steuerung statt. Die Methoden, die in der Klasse Game.as definiert sind, können im Java-Programm aufgerufen werden.

Für die Präsentation des Programms haben wir ein Fussballthema umgesetzt. Die Methoden können aber ohne Änderungen für andere Szenarien verwendet werden. Lediglich die Steuerung der Anzeige des Balancewertes im Flash muss unter Umständen an das Szenario angepasst werden. Im Fall des Fussballspiels gibt es dafür einen Movieclip in dem zwei Animationssequenzen liegen. Eine für einen Schuss des Balls nach links und eine für nach rechts. Bei einem Bewegungsimpuls wird auf das Anfangs-Frame der entsprechenden Sequenz gesprungen. Am Ende der Animation wird der Movieclip um die Strecke verschoben, um die der Ball sich bewegt hat und der erste Frame des Movieclips angesprungen. In diesem ist der Ball in Ausgangsposition zu sehen.

IM FOLGENDEN WERDEN DIE EINZELNEN METHODEN DER GAME-KLASSE ERLÄUTERT

```
public function Game(roots:MovieClip) {
    this.tugroot = roots;
}
```

Dieser Konstruktor benennt die Rootebene unseres Flash-Films von _root in _tugroot um. Durch den eindeutigen Namen werden Überschneidungen mit den anderen Flash-Filmen des Public Screens vermieden.

```
public function reset() {
    _root.gotoAndStop(1);
}
```

Um das Spiel auf den Anfangszustand zurückzusetzen kann diese Methode aufgerufen werden.
Gibt ein Spieler bei einer Rechnung ein falsches Ergebnis ein, kann durch den Aufruf von

```
public function offsideRed() {
    _root.offsideRed.gotoAndPlay(2);
}
```

oder

```
public function offsideBlue() {
    _root.offsideBlue.gotoAndPlay(2);
}
```

auf der Anzeige des entsprechenden Teams eine Fehlermeldung angezeigt werden.

```
public function startTimer(secs){
        _root.Timer.amount=secs;
        _root.Timer.gotoAndPlay(2);
    }
```

Hier wird der Timer mit einem Sekunden-Betrag als Startwert initialisert und gestartet. Er steuert lediglich die Zeitanzeige im Flash, das Ende des Spiels wird über einen Timer im Java ausgelöst. Beim Start des Timers wird über getTimer() die aktuelle Zeit in der Variable starttime gespeichert. Diese wird dann beim Ablaufen von der jeweils aktuellen Zeit abgezogen.

```
// Bewegung nach links
public function kickLeft(){
    if (_root.blocked) {
    _root.queue.push(„kickleft“);
    }
    else {
    _root.ball.gotoAndPlay(„kickleft“);
    }
}

// Bewegung nach rechts
public function kickRight(){
    if (_root.blocked) {
    _root.queue.push(„kickright“);
    }
    else {
    _root.ball.gotoAndPlay(„kickright“);
    }
}
```

In den Methoden kickLeft() und kickRight() werden die Bewegungsimpulse die von den Clients an den Server geschickt werden vom Java an das Flash weitergeleitet und in der Grafik visualisiert. Dazu werden in einem Movieclip bestimmte Bereiche angesteuert und abgespielt. Es kann vorkommen, dass zwei Impulse kurz nacheinander eintreffen. Da die Animation zuerst zu Ende gespielt werden soll, bevor eine neue Animation gestartet wird, gibt es eine Warteschlange in der Animationsaufrufe gespeichert werden können.

Solange eine Animation läuft wird eine boolsche Variable blocked auf true gesetzt. Falls eine neue Animation gestartet wird, prüft die Funktion zunächst ob blocked auf true gesetzt ist. Ist dies der Fall, wird der Aufruf in Form der Bewegungsrichtung in einem Array gespeichert. Nach Ablauf einer Animation werden die gespeicherten Animationsaufrufe nacheinander abgearbeitet.

```
public function numberLeft(number){
    _root.textLinks.text = number;
}

public function numberRight(number){
    _root.textRechts.text = number;
}
```

Diese Methoden ermöglichen die Übergabe und Anzeige einer Zahl auf den jeweiligen Anzeigebereichen der beiden Teams. Hierfür werden dynamische Textfelder verwendet. Da diese leicht rotiert sind, muss darauf geachtet werden die Zahlen der Schriftart einzubetten, da sie sonst nicht angezeigt werden.

```
public function whistle(){
    _root.gotoAndPlay(„whistle“);
}
```

Beim Start des Spiels wird hier eine kurze Startanimation abgespielt.

```
public function winBlue(){
    _root.gotoAndStop(4);
}

public function winRed(){
    _root.gotoAndStop(5);
}
```

Beim Sieg eines Teams wird über diese Methoden ein entsprechende Meldung angezeigt.


```
public function drawn(){
        _root.gotoAndStop(„drawn“);
    }
```

Bei Unentschieden wird über diese Methode ein entsprechende Meldung angezeigt.

```
// Spiel starten
public function startGame() {
        _root.gotoAndPlay(2);
    }
}
```

TEST

Für das Testen der Anwendung sind folgende Punkte zu beachten:
Server starten.
Starten der Anwendung und Auswahl des Balance-Plug-Ins auf mindestens zwei Geräten
Nach dem Starten erscheint ein grauer Startbildschirm mit einem Logo. Die Anwendung auf dem Client wurde erfolgreich gestartet.
Der Spieler ist aber noch nicht eingeloggt und noch keinem Team zugeordnet worden.
Durch Drücken der * Taste loggt man sich ein und wird im Anschluss einem Team zugeordnet.(rot oder blau).
Bevor das Spiel aktiviert wird müssen sich alle Mitspieler eingeloggt haben und einem Team zugeteilt sein.(Mindestens zwei Spieler)
Wenn alle Spieler eingeloggt sind kann einer der Spieler das Spiel durch erneutes Drücken der * taste starten.
Der erste Spieler jedes Teams muss zunächst die auf dem Public-Screen gezeigte Zahl abtippen.

```
private void flashsend(String func)
private void flashsend(String func,String[] param)
```

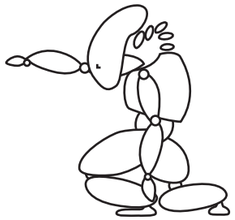
Für weitere Beschreibungen der Funktionsweise möchten wir an dieser Stelle auf die Javadoc-Dokumentation und die Dokumentation der technischen Rahmenanwendung verweisen.

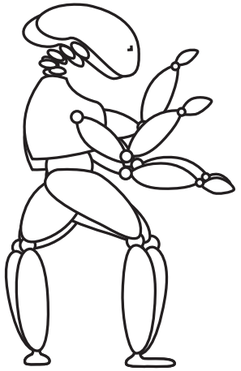
Der weitere Spielverlauf entspricht der Konzeptbeschreibung.
Die eingegebene Zahl kann man mit Drücken der # Taste bestätigen und zum Server senden.
Sollte man sich vertippt haben kann man durch weiteres Drücken der Zahlentasten seine alte Eingabe überschreiben.
Sollte während einer Spielrunde ein Client beendet werden oder verliert ein Client die Verbindung zum Server ist ein weiteres Testen nicht möglich und die Anwendung muss neu gestartet werden.

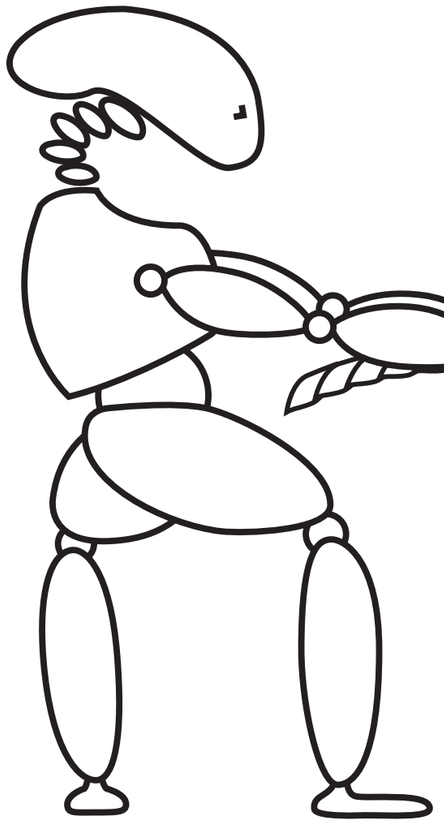
Die Anwendung wurde erfolgreich auf Server- (Windows XP) und Client-Seite (Sony Ericsson K750i) getestet. Der Spielverlauf war dabei reibungslos und das Konzept hat sich als unterhaltsam herausgestellt. Technisch bedingte Probleme z.B. Übertragungsgeschwindigkeiten waren unwesentlich.

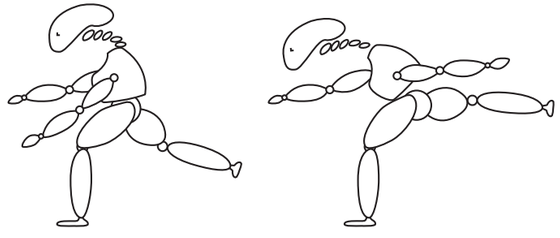
DESIGN
FÜR SEILZIEHEN

Am Beispiel von Seilziehen Wurde ein Design entwickelt welches in das Gesamtkonzept eines dreigeteilten Bildschirmes passen sollte. Die Spielfiguren stellen Aliens mit menschlicher oder roboterhafter Anmutung dar. Das Szenario findet vor einer Citykulisse statt. Die Figuren bestehen aus vielen Einzelteilen, was die Animation erleichterte. Dass die Wahl auf menschenähnliche Figuren fiel, liegt daran, dass der Spieler sich damit leichter identifizieren kann. Zudem lässt sich die Körpersprache der Figuren so besser deuten.



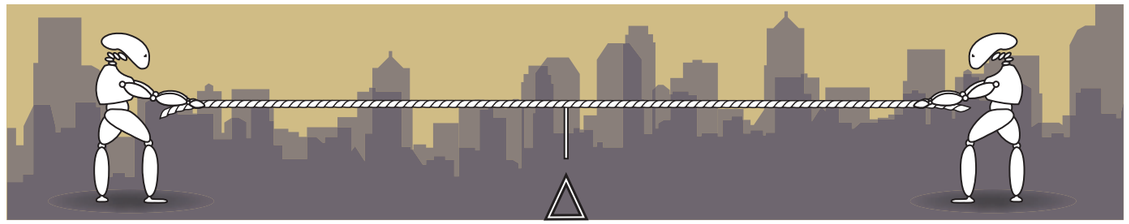


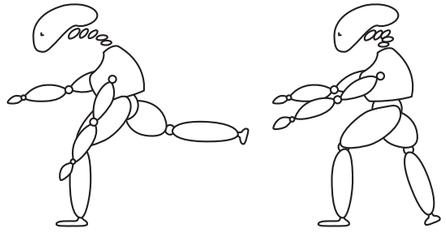




VON DER IDEE

ZUR UMSETZUNG





FAZIT UND AUSBLICK

Nach einer längeren Phase in der Ideen und Konzepte in der Gruppe diskutiert wurden, hatten wir uns eigentlich auf den Konzeptvorschlag eines „FHA-Adventure-Game“ geeinigt. Auf Grund von Schwierigkeit bei der Koordination mit anderen Projekten die gleichzeitig auf dem Public-Screen gezeigt werden sollen (Das Spielkonzept hätte einen eigenen Screen benötigt) haben wir uns trotz des fortgeschrittenen Semesters zu einem neuen Ansatz entscheiden. Bei der Programmierung mussten wir uns zunächst einen groben Überblick über die J2ME-Entwicklung verschaffen und uns in Eclipse, SVN und die parallel entwickelte Rahmenanwendung einarbeiten. Hierbei konnten wir uns auf die Hilfe der entsprechenden Projektgruppe stützen. Trotz der aufgetretenen Probleme konnten wir uns einen Einblick in die Konzeption, Gestaltung und Programmierung (J2ME) verschaffen. Basierend auf dem entwickelten Prototypen kann mit überschaubarem Aufwand eine stabile Anwendung erstellt werden. Das einfache Bedienkonzept ist einfach zu vermitteln und lässt verschiedene Szenarien in denen die Wettkämpfe stattfinden zu. Variationen und Szenarios mit aktuelle Bezüge sind mit wenigen Änderungen und einer anderen Oberfläche auf dem Public-Screen leicht zu realisieren. Eine Erweiterung des Spiels auf eine unbegrenzte Anzahl von Spielern ist theoretisch möglich, indem Clients gleichzeitig als Bridge für andere Geräte dienen. Für eine Verwendung oder Weiterentwicklung der technischen Rahmenanwendung in zukünftigen Projekten kann unser Prototyp als einfaches Lehrbeispiel für einen erleichterten Einstieg dienen.

QUELLEN UND LINKS

Version 2.0 of Mobile Information Device Profile Specification
Java in a Nutshell, David Flanagan, O‘Reilly, 1999
ISBN: 1-56592-487-8

http://www.fh-augsburg.de/~meixner/mob/online/
Mobile Processing
J2ME Skript, Prof. Meixner, FH-Augsburg

http://mobile.processing.org/index.php
J2ME Tutorial, Sten Schmidt, 2002

http://www.midlet-review.com/index?content=articles&id=3
Benhui.net, the harmony of mobile development

http://www.benhui.net/index.php
Dokumentation des Technischen Rahmens