Mobile Game Project Proposal

Project Description

With the world-wide proliferation of mobile technology the field of so-called ubiquitous computing becomes increasingly relevant. Through cell phones, computing resources and software are available to more people than ever before, and entertainment software is no exception.

The entertainment software industry and market is relatively small when compared with traditional types of entertainment such as movies and music, but it is growing extremely fast.

The goal of this project is to build a multi-player turn-based strategy game for Java-enabled mobile telephones.

Why Turn-based?

A turn-based game has a number of advantages over real-time games on the mobile platform.

Potentially, a large number of people who will be exposed to cell phone games don't usually play games, or even use computers at all.

Traditional real-time games usually require the player to have a certain amount of skill and coordination for the game to be enjoyed, whereas a turn-based game does not rely on a high degree of control, since the timing of a player's actions is far less significant and the player will usually have the time to ensure their move is the one they want.

This means that the requirements for novice users should be lower, and the limitations of a cell phone keypad or joystick as an input device are less likely to be a factor in preventing an enjoyable experience.

The performance requirements for turn-based game are also not as high, which allows deployment to a greater variety of devices.

Social Aspects

For many people a mobile phone is as much a social tool as a business one, and a mobile multiplayer game would provide a mechanism for a different type of social interaction.

The turn-based format allows for text-based communication during the course of a game, which is usually difficult during a real-time game, and would be especially unwieldy on a mobile phone. An effective in-game messaging mechanism will however, depend on the ability to provide T9 predictive text functionality that users are accustomed to from Java.

With a turn-based game it is possible to have any amount of time between turns, and this flexibility is a natural fit with the usage habits of mobile phone users. People will use their phone in short bursts or over a prolonged period. It is used to communicate and socialise.

People will typically carry out a conversation using a number of text messages or with a single call. A turn-based game can provide types of interaction analogous to both of these, with the added possibility of interaction between multiple parties at once.

A major drawback is that turn-based games are not popular. This is quite likely due to the fact that they are not as direct as a traditional point and shoot game and are therefore not as attractive.
**Game Overview**

For reasons outlined above the focus will be on the game being easy to learn and easy to use. This should be achieved by providing the minimum functionality required to provide a pleasant experience for players, and making the user interface as consistent as possible.

One example of this would be to keep the number of weapons as low as possible while still providing enough variety that players have a reasonable number of strategies at their disposal.

Another example would be to ensure that as far as is possible all weapons work the same. If any weapon requires a power bar for instance, it should look the same, be present in the same place on the screen, and be triggered in the same way.

**Game Outline**

Setting the possibility of server-based communities of players aside, a game would be initiated by a user creating a game as the game host and inviting other users to join. Creating a game might involve selecting a turn length and selecting or generating a map.

After the invited users accept or decline the invitation, possibly after reviewing the game’s configuration, the host would start the game.

Each player is allocated a team of characters, which are placed at random in the map, and the first player to take a turn is chosen randomly.

The objective of each player is to eliminate all of the characters on the other teams by reducing their health to zero.

The characters on a team are selected to played in the same order throughout the game.

During a turn, a player:

1. can optionally move the selected character about the map (as far as is possible)
2. select a weapon or utility
3. fire the weapon or use the utility
4. type in a message to be sent to the other players along with the turn

Actions 1 and 2 above may happen in any order, and it might be useful for a player to be able to send a message to the other players even when it is not their turn.

The game continues with each player getting a turn until they have no more characters left on their team. Eliminated players should have the ability to spectate for the rest of the game.

Host should have the ability to kick unresponsive or abusive users from the game.

**Related Work**

“Creating Entertainment Applications for Cellular Phones” [1] sets about examining and comparing mobile game development platforms. It focuses on three such platforms: J2ME, Symbian and BREW. The paper draws the conclusion that the choice of development platform should depend on the specific function the application will perform.

“Platforms for Play” [2] describes and contrasts the three most popular game development platforms at the moment (J2ME, BREW and Symbian), as well as a few lesser known platforms. It focuses mainly on financial and economic considerations, as well as taking the issue of compatibility across different phones into account (see Table 1).

“J2ME: The Next Major Games Platform?” [3] heralds J2ME as the next major game development platform. It also provides a discussion of MIDP 2.0 and PersonalJava. It presents an experiment to determine the ability of J2ME to run a 3D game, and concludes J2ME is suitable for running games featuring 3D graphics.
In “Game Design Patterns for Mobile Games” [4] an attempt is made to classify the most common of these dimensions as design patterns, specifically for mobile games. As is accepted practice for the development of patterns, a number of workshops were held in which a number of game design patterns were “extracted” from a collection of games, and a list of 74 commonly occurring patterns was generated.

“Contextual Virtual Interaction as Part of Ubiquitous Game Design and Development” [5] presents the notion of “rich interaction”, which is essentially complex interaction sets. Among other things, the paper describes guidelines for creating rich interaction from simple interfaces, and promotes the notion that interesting interactions can be obtained from simple user interfaces. This gives hope to mobile application developers that despite the input and display limitations inherent in mobile platforms such as PDAs and cellphones, the creation of non-trivial applications is possible.

<table>
<thead>
<tr>
<th>Constraints</th>
<th>J2ME</th>
<th>Symbian</th>
<th>BREW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Restricted to 30 - 64k</td>
<td>A few MBs. However, OTA installations limit to 128 - 256k</td>
<td>A few MBs. However, OTA installations limit to 128 - 256k</td>
</tr>
<tr>
<td>Application Speed</td>
<td>Must run over JVM so slower.[1]</td>
<td>Compiled to machine code.[1]</td>
<td>Compiled to machine code.[1]</td>
</tr>
<tr>
<td>Data Storage</td>
<td>Record Management System (RMS) and JDBC. Must use allotted memory.[1]</td>
<td>Can use large amounts of variables sized memory[1]</td>
<td>Built in database. Can use large amounts of variables sized memory[1]</td>
</tr>
<tr>
<td>Cross Platform Flexibility</td>
<td>Very portable. A large problem is that it cannot automatically detect screen sizes. [1]</td>
<td>Different version of Symbian exist, but applications can port within versions without modification. [1]</td>
<td>BREW can only run on Qualcomm chipsets [1]</td>
</tr>
<tr>
<td>Market Penetration</td>
<td>Java enabled phones are prevalent. J2ME can run over BREW and Symbian [1]</td>
<td>Symbian has 41% market share [1]</td>
<td>BREW has small support based in the USA[1]</td>
</tr>
<tr>
<td>Development Cost</td>
<td>Free</td>
<td>Free</td>
<td>Compiler, registration with Qualcomm, TRUE BREW certification[5]</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Mobile Development Platforms
Mean Opinion Score tests are a means to collect subjective data, and are often used to measure the perceived quality of telecommunications equipment and services. The authors of “Subjective Quality Assessment for Multiplayer Real-time Games” [6] attempt to apply and extend this testing strategy to measure the perceived quality of a real-time networked multiplayer mobile game.

In “Realtime Multiplayer Games over Mobile Networks” [7], existing strategies for coping with high latencies in the mobile, wireless networked environment are outlined, and a new one is introduced.

“Overviewing Scientific Research for (Mobile) Gaming” [9] is an attempt to outline the various research areas in gaming in general and mobile gaming in particular. The areas mentioned were:

1. **User Behaviour**
2. **Quality of Service**
3. **Game Related (genres, design etc.)**
4. **Network Structures and Protocol**
5. **Security and Cheating**
6. **Mobile Support**

### Work Detail

An outline of the main project tasks follows, with a Gantt chart in Appendix A.

1. **Coding**
   1. **Map Editor/Generator** Maps will form the arena in which games will take place. They will either need to be generated or manually created using an editor. Maps will likely be a pixel-based image which defines the initial (deformable) terrain.

2. **Physics** Implementation of the interactions between characters, weapons and terrain.

3. **Graphics** Facilities for animating and displaying game elements. Graphics and physics are closely linked as the shapes of game elements will affect the types of physical interactions.

4. **User Interface** During the course of a game the user will need to interact with the game via a system of menus and power bars, as well as directly positioning the characters and aiming weapons.

5. **Networking** As the game will be networked and multiplayer, both client and server networking code will need to be written.
   1. **Client**
   2. **Server**

2. **Artwork** The services of an artist will be required to create artwork for each game element.

   1. **Map** Map artwork is likely to be a collection of props and tile sets.

   2. **Character(s)** The game will use artwork for a single character for each member of a team, with different colour schemes overlaid for each team.

   3. **Weapons** Weapon artwork will be required to indicate the current weapon selection to users.

3. **Testing**

   1. **Usability and Game-play** Once an initial implementation has been written, it will be useful to observe people using and playing the game. Hopefully the game’s major issues will be illuminated.

   2. **Fixes and Modifications** As far as is possible in the limited time, issues discovered in the previous task should be resolved.
Outcomes

The System

The general goal of this project is to explore and compare different aspects programming on mobile devices. In order to achieve this a game that can take advantage of the mobile environment will be designed and implemented. Such a game could employ the networking capabilities of the phones and make good use of the graphics and sound functionality available.

A turn based game, similar to the popular Worms games, would be an ideal type of game to explore the Java 2 Micro Edition (J2ME) programming environment and the different operating systems available.

Questions/Issues tackled

The various aspects that will be explored are as follows:

J2ME as a Mobile Game Platform

The mobile game will be implemented in the J2ME programming language. In recent years J2ME has been enhanced to enable the writing of computer games. During this design process the strengths and weaknesses of these enhancements will be assessed.

Development Tools

Different development tools will be used to assist in the development and programming of the turn based game. In order to develop the game in it's most efficient manner and in order to take advantage of different features as well as possible an analysis of development tools will be taken in order to determine which development tool is best suited for our purposes.

Mobile Operating Systems

There are two operating systems that currently dominate in terms of usage, namely Symbian and .Net. During the game design process a decision will need to be made to gear the game toward a specific operating system. This decision will be made according to the different features, strengths and weaknesses of each operating system.

Interface To Wireless Network

An API or set of APIs will be identified which could help games application developers interface their games to a wireless network.

Expected Impact of the Project

The majority of work in this project will be to produce an enjoyable game for the mobile cell phone environment. Should this end be achieved the project will be considered a success. However, due to the open ended and exploratory nature of this project it is possible that we will come across further relevant areas of interest to evaluate, but it is hard to judge exactly what these areas might be at this early stage.

With a successfully completed game the strengths and weaknesses of the different aspects of mobile programming which are explored will be examined from the perspective of mobile game development and deployment.

Conclusion

This project should provide useful insights into the mobile game development process and technology as well as end-user requirements.

Furthermore, the experience of building a game in this project will create a platform for projects that seek to investigate the mobile cell phone environment further.
References


<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Coding</td>
<td>24 days</td>
</tr>
<tr>
<td>2</td>
<td>Map Editor/Generator</td>
<td>7 days</td>
</tr>
<tr>
<td>3</td>
<td>Physics</td>
<td>7 days</td>
</tr>
<tr>
<td>4</td>
<td>Graphics</td>
<td>7 days</td>
</tr>
<tr>
<td>5</td>
<td>User Interface</td>
<td>7 days</td>
</tr>
<tr>
<td>6</td>
<td>Networking</td>
<td>8 days</td>
</tr>
<tr>
<td>7</td>
<td>Client</td>
<td>8 days</td>
</tr>
<tr>
<td>8</td>
<td>Server</td>
<td>8 days</td>
</tr>
<tr>
<td>9</td>
<td>Final Integration of ArtHc</td>
<td>3 days</td>
</tr>
<tr>
<td>10</td>
<td>Artwork</td>
<td>16 days</td>
</tr>
<tr>
<td>11</td>
<td>Map</td>
<td>10 days</td>
</tr>
<tr>
<td>12</td>
<td>Characters</td>
<td>14 days</td>
</tr>
<tr>
<td>13</td>
<td>Weapons</td>
<td>10 days</td>
</tr>
<tr>
<td>14</td>
<td>Testing</td>
<td>7 days?</td>
</tr>
<tr>
<td>15</td>
<td>Usability and Gameplay</td>
<td>2 days?</td>
</tr>
<tr>
<td>16</td>
<td>Music and Soundtracks</td>
<td>6 days</td>
</tr>
<tr>
<td>17</td>
<td>Report</td>
<td>29 days?</td>
</tr>
<tr>
<td>18</td>
<td>Definition and Theory,Box</td>
<td>5 days?</td>
</tr>
<tr>
<td>19</td>
<td>Design Chapter</td>
<td>5 days?</td>
</tr>
<tr>
<td>20</td>
<td>Implementation and Testrun</td>
<td>5 days?</td>
</tr>
<tr>
<td>21</td>
<td>Report Outline</td>
<td>1 day</td>
</tr>
<tr>
<td>22</td>
<td>First Draft</td>
<td>4 days?</td>
</tr>
<tr>
<td>23</td>
<td>Final Report</td>
<td>6 days?</td>
</tr>
<tr>
<td>24</td>
<td>Poster</td>
<td>6 days</td>
</tr>
<tr>
<td>25</td>
<td>Web Site</td>
<td>6 days</td>
</tr>
<tr>
<td>26</td>
<td>Paper</td>
<td>6 days</td>
</tr>
</tbody>
</table>

Figure 1: Task Overview

Appendix A