NTNU
Innovation and Creativity

MOOSES Game Concepts

Game Concepts for the Multiplayer on One Screen Entertainment System

Audun Kvasbø

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Alf Inge Wang, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

This assignment is based upon the creation and design of new concepts for multiplayer games on one big screen.
The candidate shall create some concepts that are suited for this particular type of games and create game designs for these concepts.
The assignment does not include game implementation.

Assignment given: 18. January 2007
Supervisor: Alf Inge Wang, IDI

Abstract

Today, video games are mostly played at home while alone or together with friends. Multiplayer games are played by sharing a single screen or meeting up online to play against others. Within the MOOSES (Multiplayer On One Screen Entertainment System) project we seek to create a set of games for the new gaming paradigm that allows large numbers of players to share the fun of playing together on a single, large screen.

To create games that are playable within the MOOSES context, one has to consider a series of special factors. These include elements of both user interface design, hardware and software architecture. In this study we describe these factors and how they can be handled with respect to making fun games for a large number of players on a single screen.

Then, in the final and most important part of the study we describe five games that are suited for the special demands of the MOOSES framework - a war game, a football game, a music game, a survival based game and a quiz game.

Preface

This report is written as a Masters Thesis at the Norwegian University of Science and Technology.

The project lasted from mid-January to mid-June 2007.

Thanks to my supervisor, Alf Inge Wang for guidance and to Seth Piper for the concept artwork.

Trondheim, June 15, 2007

Audun Kvasbø

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

As the video game medium has matured, many different styles of playing has emerged. Today, we have games acompassing a wide range of styles, modes and playing styles, but there is one scenario that has not yet been developed. There are games for all screen sizes and a wide range of concurrent players, but there has never been created a gaming system with a significant number of games that allows a large number of individually identifiable players to play simultaneously on one large screen, at all times sharing a single, objective view of the game world.

In this report, we will write about the development of such games, outline necessary systems and feasible architectures for implementing games on this scale and, most importantly, describe some game concepts that could be implemented for use in venues such as cinemas.

1.1 Motivation

With an entirely new game technology at our disposal, we recognize the need for a varied and rich selection of available software and games to help the MOOSES project succeed as a support for a viable gaming environment in the future. A completely new gameplay paradigm demands that games should be specifically designed for use within it, and this calls for a thoroughly thought through set of game concepts. We hope that these game concepts might subsequently be implemented and used with the MOOSES framework, and that this can help large crowd gaming become a widespread genre and that a varying flora of games can be developed for the big screen.

1.2 Problem Definition

This assignment is based upon the creation and design of new concepts for multi player games on one big screen.

The candidate shall create some concepts that are suited for this particular type of games and create game designs for these concepts.

The assignment does not include game implementation.

1.3 Project Context and Stakeholders

This project is carried out as a Masters Thesis at the Department of Computer and Information Science at Norwegian University of Science and Technology, NTNU.

The development of MOOSES game concepts is an integrated part of NTNU's Research Programme on Video Games[1]. In addition to this Masters Thesis, there are several other working groups participating in the MOOSES programme. Most notably, Sverre Morka who is working on a flexible MOOSES client, Aleksander Spro and Morten Versvik who are working on the server modules of the MOOSES framework[2] and the company TellU[3] who is working on commercializing the MOOSES project.

1.4 The MOOSES Framework

Although it will be described in detail later, we feel that a short definition of the MOOSES framework is important also here in the introduction.

The MOOSES framework is a framework for Multiplayer On One Screen games. The first version of the framework was created at NTNU in the autumn of 2006, and its current version has undergone successful tests at the Nova movie theatre in Trondheim

The framework handles the demands of multiplayer games that is to be played on a big screen, and ensures that a game developer will not have to handle all the common tasks of all games such as communication with the hand controls (often phones), integration with billing systems, score boards and other aspects that all games is likely to have in common. The framework is currently in development, and current information can be found at *http://www.mooses.no*.

1.5 Definition of a MOOSES Game

Throughout this report we will be using the term "MOOSES game" quite regularly. The way we use it, it relates to a game that utilizes the MOOSES framework and that is designed to be used in a social setting with a large number of concurrent players.

A MOOSES game as described in this report is meant to be used with current or future version of the MOOSES framework. Therefore we might sometimes allow ourselves to assume inclusion of theoretical future functionality in the framework, in controllers or in hardware or software development.

[1]http://www.ime.ntnu.no/forskning/prosjekter/dataspill
[2]http://www.mooses.no
[3]http://www.tellu.no

Chapter 2

Report Outline

In the following sections we will present a chapter by chapter outline of the contents of this report, along with brief descriptions of contents of each chapter.

2.1 Prestudy

The prestudy contains all discussions regarding preconditions for creating MOOSES games, as well as technical details, prior art and examples of existing multiplayer games.

Chapter 3: Research The Research chapter describes the research focus and methods used in this study as well as our research questions.

Chapter 4: The Moos/Mooses Project In this chapter we outline the framework that this project is built upon - the Moos project that was carried out in the fall of 2006. It is now renamed the MOOSES project, and it is this name that will be used throughout this report.

Chapter 5: Multiplayer Games In this chapter we look at a representative selection of popular multiplayer games from different genres as well as on their defining characteristics. By doing this, we highlight the most important factors that form the player experience in multiplayer games.

Chapter 6: Technical Considerations Here we describe the game design hurdles that needs to be overcome in order to successfully design a MOOSES game. We also look at very high level system requirements (hardware and software) for game concepts. In the end, we go through different possible ways of connecting a large number of controllers to the MOOSES framework.

Chapter 7: Control Schemes This chapter describes different control schemes that MOOSES game developers might choose to implement. It has its main weight on the mobile phone as a game controller, but also includes a selection of different other control methods that fits well into the MOOSES concept.

Chapter 8: Existing Big Screen Games Although the MOOSES project represents something new, there are a couple of existing games that have similarities big enough to deserve mention in this context, namely ”Ghost in the Cave” and ”MSN Newsbreaker”. In this chapter we present these games and their similarities with MOOSES games.

2.2 The Games

The games part of the report contains the five game concepts and designs that we have created for this project.

Chapter 9: The Music Game The Music Game is a rhythm and sound based game that includes teams of a few players competing against each other by trying to match on screen instructions. The teams are judged by how well they perform as a group.

Chapter 10: The War Game The War Game is a game that is inspired by various board games that have their players struggling for world dominance by waging war against neighbouring land areas.

Chapter 11: The Football Game The Football Game is a sports game in which a team of eleven players has to cooperate in order to defeat an opposing team. Both individual skills and team efforts are needed to succeed due to a cooperative approach to important football elements like shooting and goalkeeping.

Chapter 12: The Survival Game In this game the players need to act as a school of fish trying to get away from a common enemy. They can only do this by staying in the middle of the group. The game therefore encourages flock behaviour similar to what can be seen in the wild.

Chapter 13: The Quiz Game The Quiz Game is a game that tests the participants' knowledge on various themes. The players can compete in teams, and the game can be linked to other content showing in a cinema to enhance the audience's participation during for example commercials.

2.3 Evaluation, Conclusion and Further Work

In the final part of the report we look back at the project and forward to future possibilities. We consider our research methods and the success we have had in fulfilling our goals, and define some decisions that we have left for anybody who wish to implement the game concepts will need to make.

Chapter 14: Evaluation The evaluation looks at the research methods and questions and whether or not we have succeeding in selecting the correct methods and if we have found satisfying answers to the research questions.

Chapter 15: Further Work In the further work section we look at considerations that have to be made by a developer implementing our game concepts. We also explain the reasons for the chosen level of detail in some of the designs.

Chapter 16: Conclusion The conclusion concludes the study and contains our own judgement on whether this project has met its goal.

Part II

Prestudy

Chapter 3

Research

As the nature of this study means that it its primary product is new game concepts not necessarily based upon existing sources or on other studies, it is difficult to characterize the study within standard research categories. However, quite a lot of effort has gone into researching the current state of multiplayer gaming and other similar concepts in order to gain an overview over the domain in which the project resides - multiplayer gaming.

3.1 Research Focus

In this thesis, the research focus has been on finding out ways to create engaging multiplayer games that can be successfully played by a large number of players on a single screen. To do this, we have concentrated on studying existing multiplayer games and to find out which, if any of their characteristic aspects and gameplay elements could be successfully transferred to a MOOSES setting.

We have not spent large amounts of time on researching the purely technical implementation aspects of our game concepts, as this would be outside the scope of this study. Instead we have chosen to focus on researching the selection and creation of rules and game mechanics that would enhance our game concepts.

3.2 Research Methods

In conducting this study, we have used two main research methods, namely a literature study and an iterative engineering phase. These two methods has been somewhat intertwined in time, as the creation of game concepts began at the very start of the study and carried on throughout the entire period. The concepts were continuously enhanced by findings from the literature study in an iterative manner.

3.2.1 Literature Studies

Our main sources of information on games has been game magazines (primarily Edge[1] and Game Developer Magazine[2], to which we've had access to a large library of back issues), game

[1]http://www.edge-online.co.uk/
[2]http://www.gdmag.com/

resources on the Internet and two excellent books on games and gaming; The Book of Games [1] and High Score [7]. Apart from these general background sources, specific references are given where pieces of information has been taken from a distinct source. However, as most of research has been done in order to obtain a firm knowledge of background material and not to find specific information, not all sources are cited as it is impossible to distinguish where much of the factual information on various aspects of gaming was originally found.

3.2.2 Engineering

As defined by IEEE, software engineering means the "application of a systematic, disciplined, quantifiable approach to development, operation and maintenance of software" [8].

Although this project does not include actual implementation of software, we have chosen to use principles extracted from software engineering processes in the design of our games, as we believe that this is a method that enable us to continuously enhance our game concepts.

After studying the existing multiplayer games and other prior art we have engineered our game concept designs by using an iterative approach. We first created basic concepts with the main premises in place, and then continually enhanced these early ideas into more advanced and suitable versions. This way of working is described as "the engineering method", amongst other places in [13] where it is explained as a method where "current solutions are studied and changes are proposed, and then evaluated". This sums up our approach to concept design in a very fitting manner, as our game designs have been through many such iterations.

In addition to identifying aspects of existing gameplay that could be used we also wished to look into different difficulties and limitations imposed on games in a MOOSES setting. To accomplish this we used an engineering approach by looking at our concepts and identifying their problem areas. Applying this method enables us to both solve the problems of the game concepts and to identify the more general aspects of these problems at once. Thus, the iterative engineering method excels in offering a valuable contribution to our study as well as enhancing the products of the study.

The final game concepts are results of many iterations as well as frequent discussions with other people with lots if experience and interest within the field of computer- and multiplayer games.

3.3 Research Questions

Although this is an explorative design project, we have used our two research questions to define what we want to find and which problems that we seek to solve. Therefore, we have defined two research questions that fits well with our problem definition. The first (RQ1) deals with our attempt to identify and define the game design problem areas that are specific to MOOSES games and that anyone wanting to create MOOSES games needs to take into serious consideration.

The second research question (RQ2) allows us to utilize the findings that arise as a result of RQ1 to create game concepts that we believe will work in a multiplayer one screen setting. These game concepts are the main product of this study, as they represent the final product of our research.

RQ1 What are the major design considerations that has to be taken into account in order to successfully create a multiplayer one screen game?

RQ2 Which kinds of gameplay, manifested as game concepts, can successfully be implemented as massively multiplayer one screen games?

Chapter 4

The Moos/Mooses Project

In the autumn of 2006, Sverre Morka, Aleksander Baumann Spro and Moren Versvik at NTNU created the MOOS framework for multiplayer games on one screen [10]. This framework serves as a basis for the implementation of the game concepts contained within this study, and an explanation of this framework is therefore needed in this report. The current version of the framework is called the MOOSES framework, and it is this that we refer in this report. All illustrations in this chapter are taken from the original report.

4.1 About the Project

The MOOS project were based upon the wish to create a standardized framework that would enable smooth implementations of games supporting the kind of games this study also concers itself with - games with many simultaneous players on a single screen. The group succeeded in completing this task - ending up with a versatile framework that could support a multitude of different games while still enabling the game creators to utilize the framework to implement things like billing and control handling.

4.2 Design and Implementation

The MOOS framework were implemented using Java technology, thus making the system platform independent. The server itself uses J2SE[1], while the mobile client controllers were written in J2ME[2]. This enables the framework to run on virtually any system that have the necessary power to run any games created for the framework, and that are able to output video to the chosen output unit.

4.2.1 System architecture

The MOOS framework was designed as a highly modular system consisting of a set of pluggable servers and clients that each handles a single task in the system. Below, we list and briefly explain the most important of these, and Figure 4.1 shows the overall high level design of the framework along with the communication channels between the different modules.

[1]Java Standard Edition
[2]Java Mobile Edition

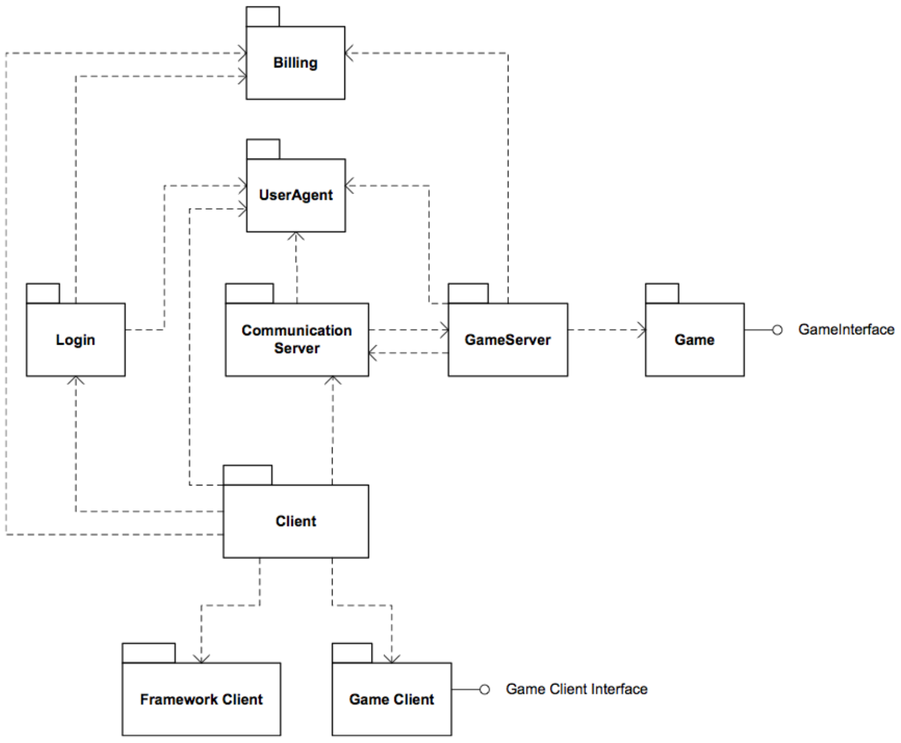


Figure 4.1: High level system design for the MOOS framework

Communication Server This is the component that allows the clients acces to the MOOS framework and that initializes communication between the modules. It allows clients to communicate with each other as if they were directly connected.

UserAgent This module represents the user, and acts as a repository for all user information and handles I/O with the user.

GameServer Interfaces with the game code. Game codes can be written in Java or C++, and the communication with the framework happens through preset functions.

Game This is the game itself, implemented as a pluggable module.

Login This module takes care of user authentication and identification of the user.

Billing The Billing module handles all billing questions. This module does not currently have any functionality in the implemented framework.

Client The module that handles communication between the client and the framework.

FrameworkClient The part of the client that handles framework-specific functionality.

GameClient This is the client side of the game code, controlling game events and possibly client screen handling.

4.3 Physical Setup

In Figure 4.2, we show the actual physical setup that is needed for use of the MOOS framework.

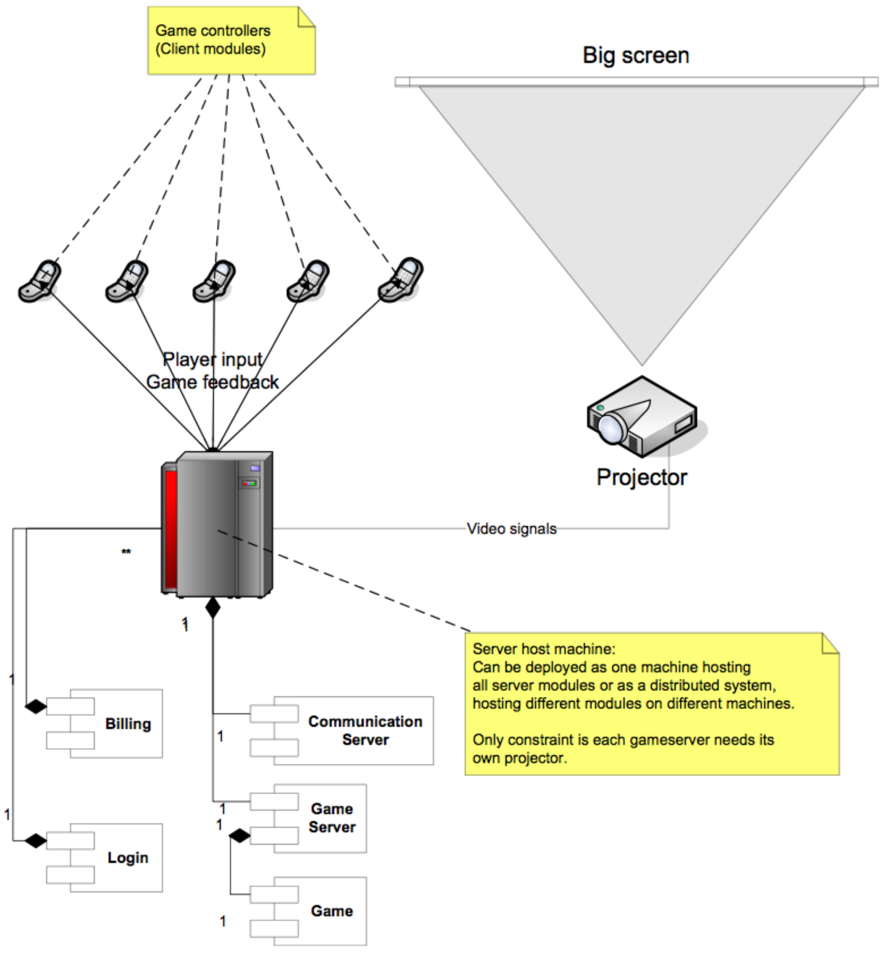

Game controllers
(Client modules)
Big screen
Player input
Game feedback
Projector
Video signals
**
1
Server host machine:
Can be deployed as one machine hosting all server modules or as a distributed system, hosting different modules on different machines.
Only constraint is each gameserver needs its own projector.
Billing
Communication Server
Game Server
Login
Game

Figure 4.2: Physical setup for the MOOS framework

The setup shown in Figure 4.2 is based around one single server that runs everything. However, this is not a requirement, and it is possible to effortlessly install the framework as a distributed system with the services (modules) spread out on a variety of servers. This ensures that the system stays scalable and flexible, and that different installations can be customized to meet the specific needs of different game modules, billing systems and controller choices.

4.4 The Current Implementation

As well as creating a reference design for the MOOS framework, the working group also created a functional installation with a prototype game. In this section we will describe this implementation and its features.

4.4.1 Test game

For their testing needs, the working group selected to modify an old open source game called Liero and to customize this game to fit the needed characteristics. Liero is very suitable for this application, as it is a real time multiplayer game in which all players act at the same time, and where there is no hard limitation on the number of players that can play at the same time.

The game itself is very similar to the more well known Worms series of games, in which a group of players try to kill each other by any means available.

4.4.2 Graphics

While the MOOS framework was created to be independent of host systems and host output format, the fact that testing and development were done with the goal of running the test applications in a cinema setting makes it even more fitting to our game concepts.

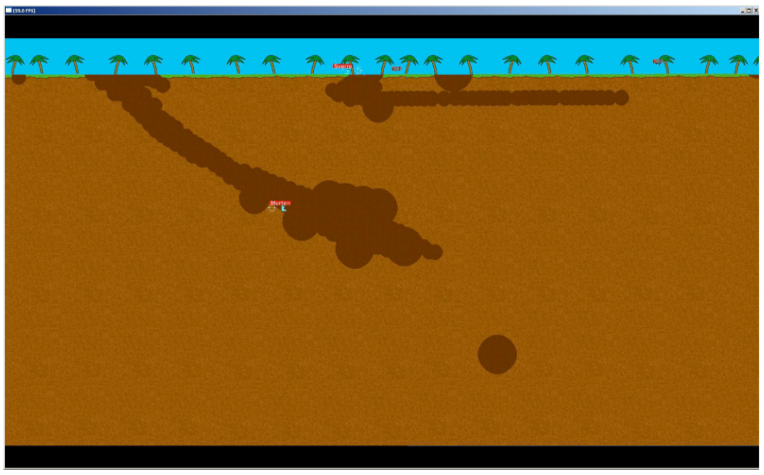


Figure 4.3: Test game for the MOOS framework

The testing (images can be seen in Figure 4.4) was done on a Sony 4K Digital projector on a big cinema screen, enabling the games to utilize a resolution of 1 920 * 1 090 pixels, which according to the report are sufficient for gameplay with at least fifty separate avatars on screen at any given time. However, the projector itself supported a resolution four times as large as this, and as the framework is resolution independent the creation of games that fully utilize the projector's potential would be trivial given that one has access to the (necessarily very powerful) right hardware.

4.4.3 Controls

Although the MOOS framework are not bound to use a specific kind of game controller, the test application and the current implementation of the framework put most of the weight on mobile phones as the game control method of choice. As the report mainly writes about the same things regarding the different aspects of mobile phones as controllers as we do in this report, albeit from a slightly more technical standpoint, we will not repeat their findings here.

Figure 4.4: MOOS testing at Nova kino

Chapter 5

Multiplayer Games

Ever since the first computer games were released players have been wanting to play games in which they can compete against or cooperate with other humans. Even in the earliest days, arcade games like Pong and MUDs[1] like Oubliette were absolutely dependent on having more than one player - they could simply not be played by one player alone. As video games have developed and become more advanced and diversified, so have the multiplayer experience. In this chapter we take a look at a representative selection of the different types of multiplayer games that are available on the market today, along with a description of the different modes of play that these games offer the players.

Then, in section 5.3, 5.4 and 5.5 we look at some game examples from the genres whose games often are multiplayer enabled. Some of the info in this chapter is taken from [1], some from the web pages of the games, and a large part is extracted from our experience obtained from playing the games themselves.

5.1 Game Characteristics

It is difficult to accurately separate the enormous selection of available multiplayer games into discrete categories. Therefore, we have instead chosen to first look at some defining characteristics that help separate between different multiplayer games, and then to look at a selection of games that by themselves describe different important concepts that define multiplayer gaming as of today. It is our hope that by showing these important concepts we are able to give an accurate description of the most important types of games that exist. Moreover, we will divide our study into two parts, namely single- and multiple location games, as this is the single characteristic that most clearly divide the available pool of games in two.

In Table 5.1 we will list some of the different factors that varies between different multiplayer games, as well as their most common values. The parentheses after the common values can be found again in Table 5.2 where they show the most normal values for the various genres. It is important to note that if a game has both single- and multiplayer modes, we have only taken into account the features of the multiplayer mode.

Of course, one can often see variations that differ from the ones that we have outlined in

[1]Multi-User Dungeon, Domain or Dimension, a game in which multiple players explore a cave structure and fight monsters while gaining experience and wealth

our definitions, but more often than not these are mere variations upon the concepts that we mention.

Table 5.1: Characterizing multiplayer game features

Characteristic	**Common values**
Multiplayer mode	co-location (cl), online (o)
Number of players	virtually any number
Play order	simultaneous (st), sequential (sq)
Screen layout	whole (w), shared (sh), split screen (ss)
Play mode	player versus player (pvp), team vs team (tvt), cooperation (co-op)
Controls	traditional (trd), keyboard (key), mouse (m), other (o)

Multiplayer mode

This characteristic defines whether the various players of a game are located at one single physical location or whether the players meet online (in this report we treat LAN[2] gaming as online gaming, as this is usually a distinction that is not made in the creation of games)

Number of players

Number of players From two players in many games to the millions of persons playing WoW - any number is possible here.

Play order

Defines which order the players act in. The possible values are “sequential” and “simultaneous”, but combinations can also be found (for example in the game described in Chapter 10, where the players select their actions simultaneously on their own private screens and these are acted out semi-simultaneously on screen afterwards).

Screen layout

Here we have three possible values. Either each player has the screen to himself (usually in sequential or online games), the players share the same screen at the same time (the mode used in the game concepts in this report) or the screen is split into parts and each player has one part each (the most usual screen layout for shooters and driving games).

Play mode

Defines the way the players interact in multiplayer games. Possible values are *Player vs. player* where the players fight each other, *team vs. team* which is basically the same only with teams fighting each other. In addition to these two modes, many games also implement different variations over the concept of *co-operative multiplayer.*

[2]Local Area Network

The multiplayer play mode is a very important characteristic that largely decides how a game is played by the users. For example, even though two games like Unreal Tournament (Section 5.4.2) and Gears of War (Section 5.4.3) look very similar on first glance, it soon becomes apparent that their implementation of multiplayer is very different. Although both have competitive modes, the main multiplayer mode in Gears of War is a cooperative one where the players have to work as a team and help each other reach a common goal, whilst the primary goal in UT is to fight each other or a team of other players.

Controls

As we explain in Section 7 (page 47) the way of controlling a game varies greatly. The methods that we mention in Table 5.1 are some of the possible controllers, but the “other” category also covers very many games, for example those who utilize custom controllers, steering wheels, arcade joysticks, microphones and other special game controllers.

5.2 Genres

In addition to the categorizations shown above, most games can be defined as members of one or more genres. A non-exhaustive list of the most common genres whose games are normally multiplayer enabled are shown in Table 5.2, along with the most common characteristics for each genre.

Table 5.2: Game genres and values for some common characteristics

Genre	M Mode	Players	Order	Layout	P Mode	Ctrl
platform	cl	2-4	st, sq	w, ss	pvp, co-op	trd
role play	o	unlimited	sq	w	pvp, co-op	key, m
fighting	cl	2	st	sh	pvp	trd, key
fps[a]	o	2-32	st	w	pvp, tvt	key, m
tps[b]	o	2-32	st	w	pvp, tvt	key, m
party game	cl	2-8	st, sq	sh, sp, ss	pvp	trd, o
racing	cl	2	st	sp	pvp	trd, o
racing (online)	o	2-8	st	w	pvp	trd, o
shoot’em up	cl	2	st	sh	pvp, co-op	trd, key
simulation	o	2-8	st, sq	w	pvp	key
sports	o, cl	2-8	st	w, ss	pvp, tvt	trd, key
strategy	o	2-8	sq	w	pvp	key

[a]first person shooter
[b]third person shooter

In the following pages, we briefly outline the most usual game mechanics from the genres listed above.

Adventure

The adventure game is based around the player solving puzzles and defeating enemies while (usually) travelling around in some game world. Some multiplayer adventure games have been developed, but they are few and far between.

Platform

A platform game is a game in which the player defeats enemies while usually jumping from platform to platform. Here, multiplayer mode is usually co-operative, meaning that the players co-operate in their efforts to clear the levels.

Role play

Role playing games have each player taking on the role of some person in the game world, usually going on quests and trying to solve some problem whilst becoming rich and powerful himself. A prime example of a multiplayer role playing game (in fact a massively multiplayer online role playing game - a mmorpg) is World of Warcraft which we will discuss later on.

Fighting

A fighting game is exactly what it sounds like. Two players try to beat each other up in a ring, utilizing various special moves and combos to do so. The last person to stand or the one who fights best over a predetermined number of rounds wins.

Shoot'em up

Usually seen from above, a shoot'em up is based upon shooting everything that moves as fast as possible. Neverending torrents of enemies appear on top of the screen and the player has to eliminate them all to advance. When played as a multiplayer game, the additional players usually play the same role as the first - only adding to the firepower.

First/Third person shooter

Two of the most popular multiplayer genres are so similar that we describe them as one. The player enter the role of a solider of some kind, seeing the action either from the eyes of the avatar (first person shooter) or from somewhere above and below it (third person). The goal of the game is usually to shoot everything that moves ahead, although sometimes there are team members that should not be fired upon. Typical examples of multiplayer shooter games are Unreal Tournament and CounterStrike.

Party game

Although the variation between party games is great, some generalizations can be made in order to define the genre. Usually, party games consist of a set of small and short games that a group of people can compete in in order to accumulate a score. These mini games can last from a few seconds up to some minutes, but they are usually extremely simple and fast paced. Party games are created for a social setting, and their themes are often bright and humorous. Examples of party games are Mario Party, Singstar and Buzz.

Racing

One of the most developed genres of video games, the racing game is based around going as fast as possible in some kind of vehicle around some kind of course. Many variations have been created, but the basic premise is always the same. The multiplayer experience is either based upon single screen play (split screen) or online matches, and examples of games (as mentioned below) are Gran Turismo and Test Drive Unlimited.

Simulation

Simulation games are games like Sim City, The Sims, Theme Park or Flight Simulator, in which the player is allowed to control a simulation of a business, process, city or another real life situation. This genre is very broad and contains a lot of games that is hard to categorize, for example the only reason racing games is usually not counted a parts of the simulation genre is because there is enough of them to warrant a genre on their own.

Sports

Similarly to the Simulation genre, this includes lots of very different games. Sports games range from meticulous simulations of golf courses and players to wild versions of common sports with altered rules. Traditionally, multiplayer sports games has been player vs. player, but lately some team vs. team games have also appeared on the market. Examples of well known games are the Fifa (football), PGA (golf), NHL (hockey), NBA (basket) and NFL (American football) series.

Strategy

Strategy games are usually based on a war scenario, historic or otherwise. The player needs to utilize limited resources in a way that allows him to beat the other players (his enemies), and often he also needs to collect raw materials and produce the resources needed to do this. Typical examples of Strategy games are the Civilization series, Warcraft and Command and Conquer.

5.3 Single Location Games

Single location games are characterized by the fact that all the players are playing simultaneously at one single physical location, usually using a single screen shared between the players. In this category we typically find fighting games, race games, party games - as well as a lot of games that are basically single player games that also support a split screen mode where two players play at the same time at one screen, but where the game itself simulates the existence of two screens. Some of these games also work as online games, but we consider them to be primarily single location games.

5.3.1 Racing - Gran Turismo Series

The Gran Turismo series is a series of racing simulation games for Sony's Playstation platform.

The series puts its weight on realism, aiming to create a game dynamic that is as close to actual car racing as possible. While the games themselves are based around a single player

Table 5.3: Game info, Gran Turismo series

Platform	Playstation 1, 2 and 3
Genre	racing simulator
Multiplayer mode	co-location
Number of players	two
Play order	simultaneous, sequential
Screen layout	whole, split screen
Play mode	player versus player
Controls	traditional, steering wheel

story, they also contain a multiplayer mode in which two players can drive against each other, often in cars that they have obtained in the single player modus of the game. The players race with the screen split horizontally at the centre (Figure 5.1), in essence giving each player a separate screen on which he drives. The winner is simply the player who is able to negate the pre-decided number of laps in the shortest time.

This mode is the most common way for racing simulation games to implement multiplayer action. Most commonly the games support two simultaneous players, but three and four player modes can also sometimes be seen. However, this exposes the inherent weaknesses in split screen play, as each players available screen real estate of course becomes smaller and smaller as more players need to share the same screen.

Lately, some car games (including the latest Gran Turismo installation) have also supported net play over local networks and the Internet, but so far split screen has been the dominant way of implementing multiplayer in racing games.

5.3.2 Fighting - Tekken series

The Tekken fighting game series is one of the oldest and most renowned series of fighting "simulators" on any platform. It has been used as a benchmark for other fighting games, and has introduced many new features into the genre.

Table 5.4: Game info, Tekken series

Platform	multiple
Genre	fighting game
Multiplayer mode	co-location
Number of players	2
Play order	simultaneous
Screen layout	shared
Play mode	player versus player
Controls	traditional

In multiplayer fighting games like this, the players (usually two) co-exist at the same time

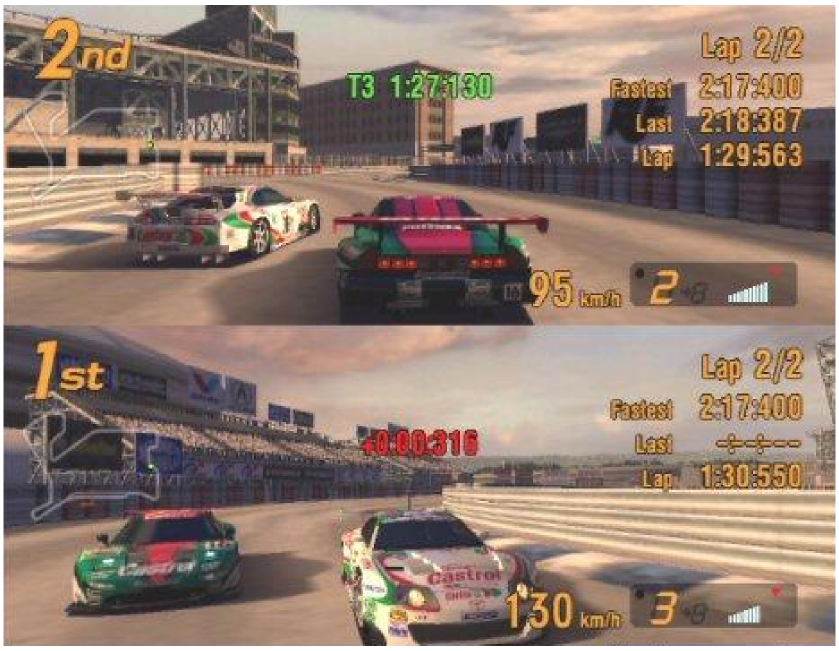


Figure 5.1: Screenshot - Gran Turismo 4 split screen

at the same screen. As the gameplay is based around the players beating each other up, this is the logical way to view a game like this, and there is very little variation in presentation between different games in this genre.

The game is played by trying to execute intricate combinations of button presses that causes the on screen fighters to perform ”combinations” of varying complexities, dealing damage to the opponent. The system of combinations allows novice players to play with only a basic set of moves, whilst the expert player have a large number of different strategies and moves to choose from. For example, a player in Tekken 3 has a total of 75 moves to choose from, 25 that is shared between all the different game characters and 50 that is unique to each character [4]. Add to this the fact that there is thirteen different characters in the game, and it becomes apparent that there is more than enough for the enterprising player to learn and master.

In fighting games, the number of players are naturally limited to only two or at most a few, as the gameplay would otherwise prove to become too cluttered and incomprehensible for the players, and all feeling of control would be lost if one tried to implement a one screen fighting game in this genre with a great number of players appearing on screen simultaneously.

5.3.3 Party Quiz - Buzz

Featuring a completely different form of gameplay than what can be found in the aforementioned games, Buzz relies on the basic game mechanics of TV game shows.

Played by up to eight players simultaneously via custom controllers (Figure 5.4), Buzz is centered around a quiz mechanism in which the players answer trivia questions and are awarded points based upon the correctness of their answers.

The game contains several different play modes that differ in the way questions are asked and the way points are awarded. Examples of these different modes are “everybody answers

Figure 5.2: Screenshot - Tekken

Figure 5.3: Screenshot - Buzz: the Big Quiz

Table 5.5: Game info, Buzz

Platform	Playstation 2
Genre	party game
Multiplayer mode	co-location
Number of players	2-8
Play order	simultaneous & sequential
Screen layout	shared
Play mode	player vs. player
Controls	custom (see Figure 5.4)

as fast as they can, point awarded for answering correctly first", "all correct answers gets points, the fastest answer gets the most points", "be the first to react when the correct answer appear on screen". In addition, there are modes in which the players compete more directly by for example allowing the winner of a round to choose an opponent that will lose some of his aquired points. These modes often appear at the end of rounds, and thus have a balancing function in the game, as the other players will tend to try to steal points from whoever is ahead in the game. In the end though, although Buzz has several different modes and many ways to score points, the entire game always keeps firmly rooted in its central concept - the video game implementation of a TV game show.

5.3.4 Party Music: Singstar

Similarly to Buzz, Singstar is also a game that it is possible to play all on your own. And even more than we see with Buzz, it is a game that it is impossible to have any real fun with unless you play it with a group of people in a social setting.

Table 5.6: Game info, Singstar

Platform	Playstation 2
Genre	party game
Multiplayer mode	co-location
Number of players	2
Play order	simultaneous
Screen layout	shared
Play mode	co-operation / player vs. player
Controls	custom (microphone)

Singstar is based around a system that allows the game code to recognize to which extent the players are singing in rhythm and tune with a song that is played on screen karaoke style (Figure 5.5).

This game is one of the most basic gameplay concepts that has been a success lately, the entire game is based upon one simple and engaging principle. Although it is technically

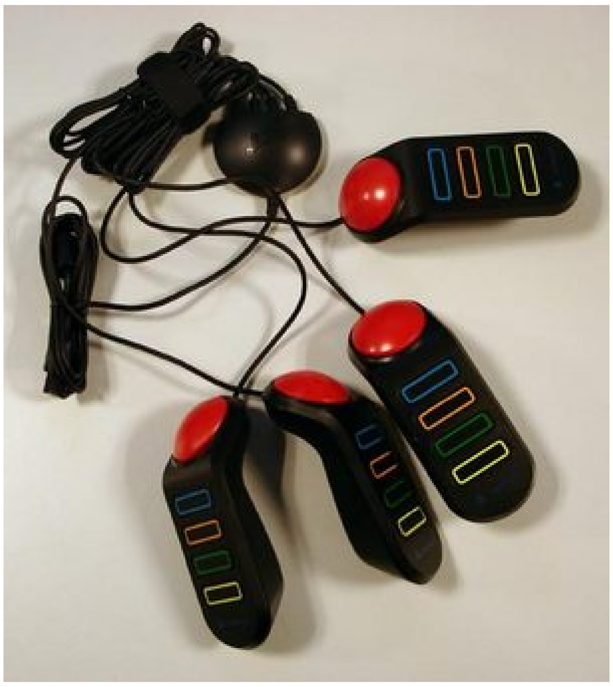


Figure 5.4: Custom Buzz controllers

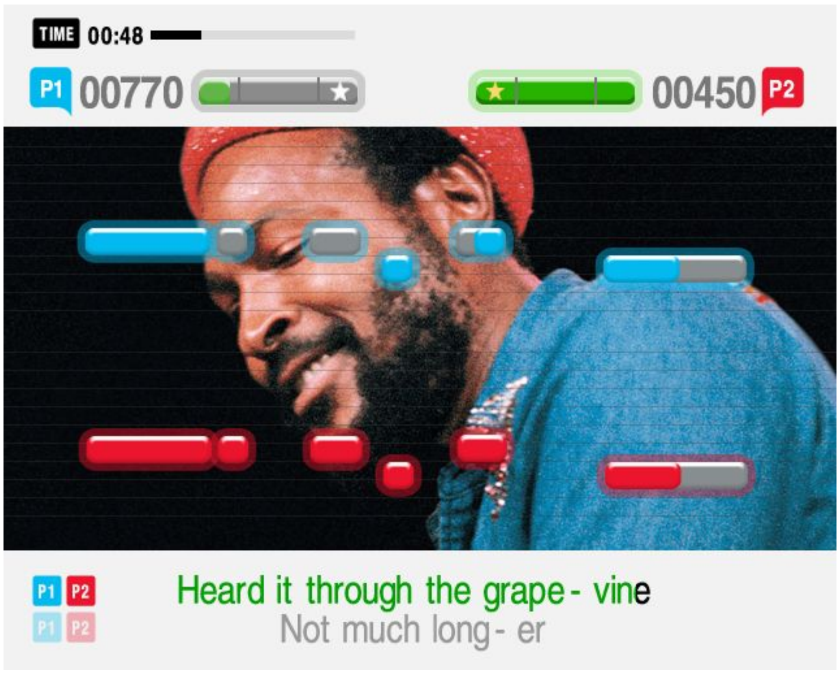


Figure 5.5: Screenshot - Singstar

somewhat advanced (due to the need to compare the players' singing to the predefined frequencies and rhythms of the songs), the player never sees this complexity and instead meets a game that is extremely simple to grasp and play.

In addition to the innovative use of microphones as sources of input, SingStar also supports video input via a special accessory. This allows the players to see themselves on screen while they play - adding even more to their immersion in the game.

5.4 Online Games

Even though games that are played by many simultaneous players in different physical locations have been around for about 30 years now (the aforementioned game Oubliette is believed to have been the first online multiplayer game), it is only during the latest years that we have seen this genre expand into becoming a major factor in the world of games. With MMORPGs[3] like World of Warcraft, online multiplayer gaming has become a huge hit amongst gamers and new games are now appearing at an ever-increasing rate. Some of the games in this section also have single location multiplayer modes, but they are placed here as we consider their online modes as the most important.

5.4.1 MMORPG - World of Warcraft

As the leading exponent of the Massively Multiplayer Online Role Playing Game genre, World of Warcraft (WoW for short) have recieved lots of attention since its launch in late 2004.

Table 5.7: Game info, World of Warcraft

Platform	PC
Genre	mmorpg
Multiplayer mode	online
Number of players	virtually unlimited
Play order	simultaneous
Screen layout	separate
Play mode	player vs. player, co-operation
Controls	keyboard/mouse

The game is placed in the game world of the old and popular Warcraft series, in which different races fight for domination and resources. Upon starting play, the player chooses a profession and a race, a choice that will follow the players character throughout the game. As in virtually all other role playing games, gameplay is then centered on gaining experience and valuables and building up a character. There is a great amount of choice in the character development - for example the player is able to learn different trades and crafts, thus choosing his path through the game.

[3]Massively Multiplayer Online Role Playing Game

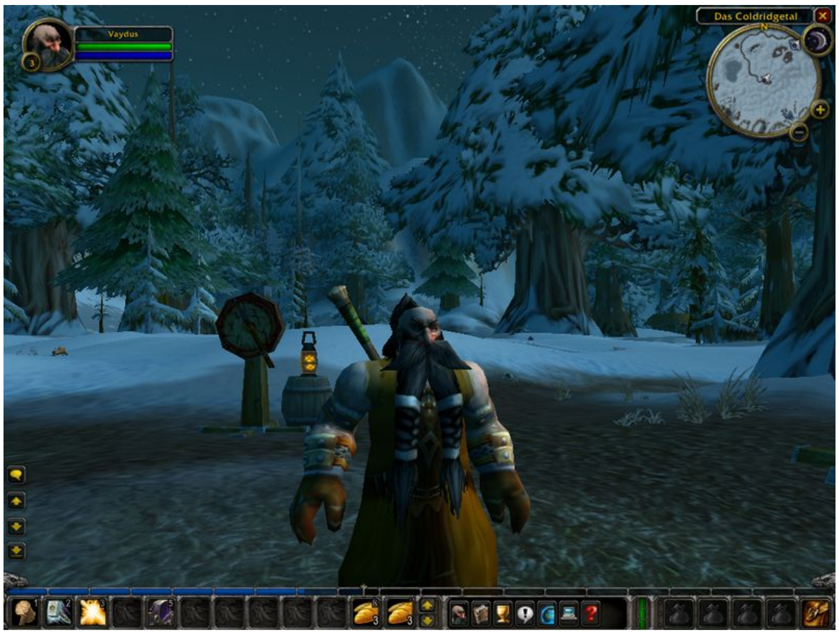


Figure 5.6: Screenshot - World of Warcraft

As the player reaches a high ”level” in the game (there are 70 in all), the play is more and more centered around joining a ”guild[4]” and completing increasingly complex and difficult quests. Some of these quests can demand as many as fifty or more players to co-operate in order to be successfully completed, and require lots of training and planning before an attempt is made.

World of Warcraft is the most successful MMORPG ever made, with more than 8,5 million active players [6].

5.4.2 First person shooter - Unreal Tournament 2004

Representing the ubiquitous First Person Shooter, the Unreal Tournament Series is by many held to be the prime example of an online enabled shooter multiplayer game.

Table 5.8: Game info, Unreal Tournament 2004

Platform	PC
Genre	first person shooter
Multiplayer mode	online
Number of players	12
Play order	simultaneous
Screen layout	separate
Play mode	player vs. player, team vs. team, co-operation
Controls	keyboard, mouse

[4]A group of co-operating players

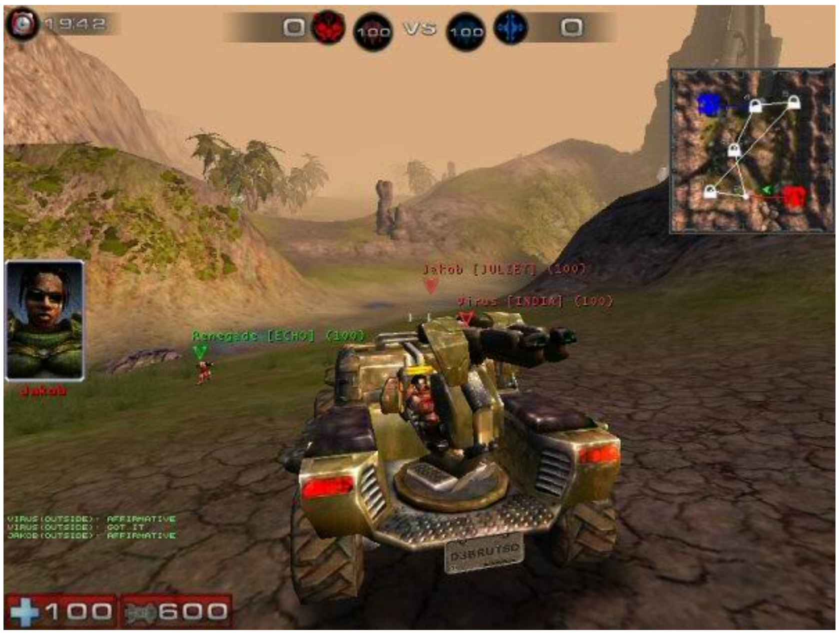


Figure 5.7: Screenshot - Unreal Tournament 2004

Based around the principle of letting the players loose in a confined space and filling the space up with equippable weapons, Unreal Tournament 2004 and its predecessor Unreal Tournament are the most well known FPS games that have almost all their emphasis on player versus player gameplay.

The games have very many modes of play, for example Deathmatch (all players against each other, the one who kills the most opponent wins), Team Deathmatch (the same, with teams), Capture the Flag (Deathmatch with an opportunity to win by capturing the enemy base), Last Man Standing (all-out-shootout, the last to survive wins) and Invasion (the players cooperate against a swarm of computer controlled enemies) [5]. All these modes are played by a number of players on a common server which keeps track of the world state, and many servers also maintain a ranking system that allows the players to compare and compete against other players in a more persistent way than just from match to match.

The Unreal Tournament games (together with the Counterstrike series) have been one of the main games in the creation of game playing leagues all over the world, with big competitions and professional teams being set up, actually allowing a small number of players to make a living from playing only these games.

5.4.3 Third person shooter - Gears of War

While Gears of War looks like a standard third person shooter at first glance, it soon becomes apparent that the way this game is played is something out of the ordinary. Instead of focusing on player versus player fights (though this is still also an option), the game makes the players fight as teams having to support each other.

Where other shooter games that have utilized team elements in the gameplay have often just made this an extension of normal player vs. player multiplayer action, the creators of Gears of War have managed to make the players fully dependent on each other all through

Table 5.9: Game info, Gears of War

Platform	Xbox 360
Genre	third person shooter
Multiplayer mode	online
Number of players	8
Play order	simultaneous
Screen layout	separate
Play mode	co-operation
Controls	traditional

Figure 5.8: Screenshot - Gears of War

the multiplayer experience.

The players have to work as a team all through the game, making sure they cover each other and give each other support in their efforts to defeat the enemies.

Thus - although it looks like any other shooter on the outside, Gears of War represented something new in the world of multiplayer shooters and set an example for how one might utilize new concepts in an old and well known setting.

5.4.4 Real time strategy - StarCraft

Although it was released as far back as in 1998, the real time strategy game StarCraft is still played by lots of enthusiasts all over the world today. The game is based around the war between three races - the Zerg, the Protos and the Terran (humans).

The game is based around the same base mechanic as all other RTS games, namely the three step process of gathering resources, arming and fighting wars. The players all start out with a very limited set of equipment, and only though expansion and research are they able to expand their army to a level that enables them to wage war against the other players.

The wars themselves are fought in real time, with the units directly attacking each other at will (Figure 5.9), with the player controlling them through orders and directives.

Table 5.10: Game info, Starcraft

Platform	PC
Genre	real time strategy
Multiplayer mode	online
Number of players	8
Play order	simultaneous
Screen layout	separate
Play mode	player vs. player
Controls	keyboard & mouse

Figure 5.9: Screenshot - Starcraft

Units that are not given explicit orders will try to attack enemy units in their proximity by themselves, but this is usually done in a non-optimal manner compared to through direct player intervention. The winner of the game is the player who is left standing after all the other players' units have been obliterated.

5.4.5 Sports - Mario Strikers Charged

Although this is not a traditional sports game we feel that it is a good example of sports game for this setting. Where most sports games are meticulous recreations of their real world counterparts, Mario Strikers Charged is a football game in which most of the rules of football has been removed and lots of new concepts have been introduced, all with the goal of creating an optimal multiplayer experience.

The game can be played by two or four players, in either one-on-one or two-on-two

Table 5.11: Game info, Mario Strikes Charged

Platform	Nintendo Wii
Genre	sports
Multiplayer mode	online & co-location
Number of players	4
Play order	simultaneous
Screen layout	shared & separate
Play mode	player vs. player, team vs. team
Controls	motion sensing and traditional

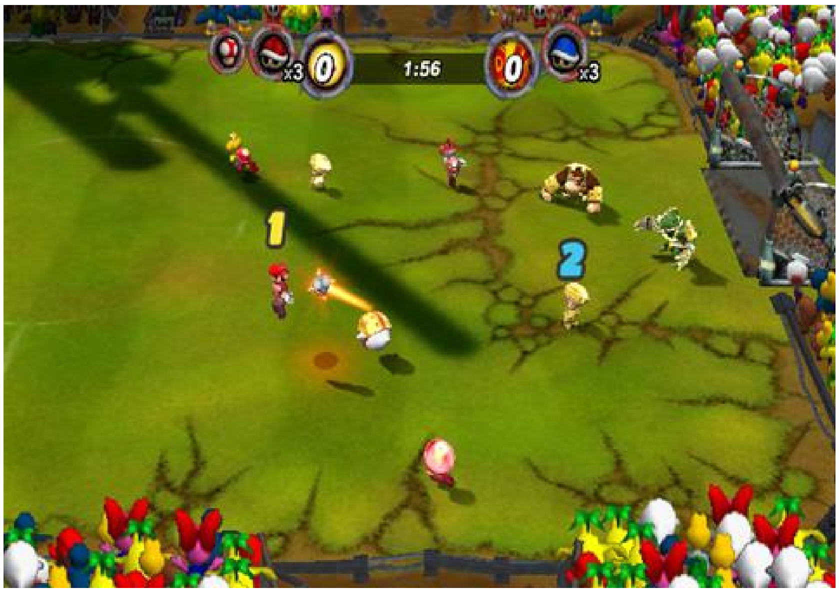


Figure 5.10: Screenshot - Mario Strikers Charged

modes. Two players can play on one console, but if four players wants to play they need to go online.

The goal of the game is the same as in any football game - to score goals, but to do this the players need to use various power ups and special equipment as well as special tricks.

Mario Strikers Charged is regarded as a very successful attempt on creating a fast and fun football game, probably because of its focus on playability and pace rather than on realism.

5.4.6 Racing - Test Drive Unlimited

Although the racing itself in Test Drive Unlimited (TDU for short) is fairly conventional, the game brings some major inventions to the relatively new online multiplayer racing game genre. Although the game is out for more platforms, we concentrate on the XBox 360 version here, as this is the one with the most advanced multiplayer features.

Table 5.12: Game info, Test Drive Unlimited

Platform	XBox 360
Genre	racing
Multiplayer mode	online
Number of players	nearly unlimited
Play order	simultaneous
Screen layout	separate
Play mode	player vs. player, team vs. team
Controls	traditional or steering wheel

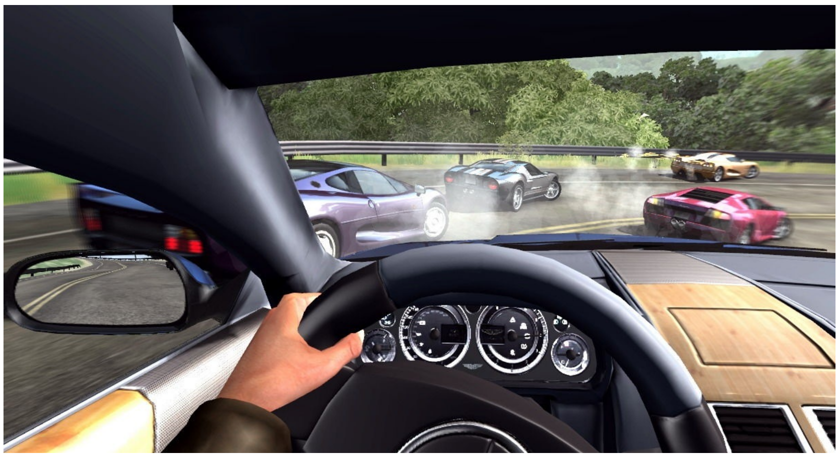


Figure 5.11: Screenshot - Test Drive Unlimited

Released early 2007, the game features the entire island of Ohau, the main island of Hawaii. This means that the game features more than 1.600 kilometres of accurately modelled roads that the player is free to roam at will, looking for the races and activities that are spread throughout the island.

The multiplayer aspect of the game is fully integrated with the single player experience. In essence, one can say that TDU owes more to Massively Multiplayer Online Role Playing Games than to regular racing games when it comes to player interaction, as the various players playing online are all part of a persistent online world where everybody can affect each others game experience.

There is a lot of different ways that players can interact with each other, the most notable are covered in the list below:

Predefined multiplayer races The game contains a set of predefined races in which players can race each other. The races are either on time, first to goal or a competition to reach the highest speed at predefined points within a set amount of time.

Player created challenges All players can create challenges for other players to participate in. These are stored at defined points to which players can drive and find a list of challenges. The creator can define a starting fee for all participants as well as a prize for the winner(s) of a challenge. In this way, a player-to-player economy where (in game) money can be made by creating content is created.

Guilds Players might create or join guilds, or ”clubs” as they are called, enabling groups of friends to cooperate and take on other guilds in racing challenges.

Item trading All players are allowed to buy and sell cars to and from other players in an open marketplace with a free market economy. Due to the fact that some cars are hard to get and rarely available, this further enhances the in-game-economy as some cars can fetch very high prices on the market.

Instant Challenge The player can also choose, upon meeting another online player, to create an ”instant challenge”, meaning that a custom one-on-one race is set up between the two players with a wager on who will win.

All in all, the game contains more than a hundred predefined multiplayer challenges as well as a virtually unlimited number of player created challenges, making the game unique in the multiplayer racing genre due to the huge amount of content and its continuous development.

5.5 Mobile Gaming

As mobile phones with net connections have become ubiquitous, we have witnessed the first attempts to put mobile games online. Although the concept of mobile multiplayer gaming has not yet fully taken hold in the western world, players all over Asia are playing mobile multiplayer games every day, and in this section we will take a look at one of the most successful games in this genre, Samurai Romanesque.

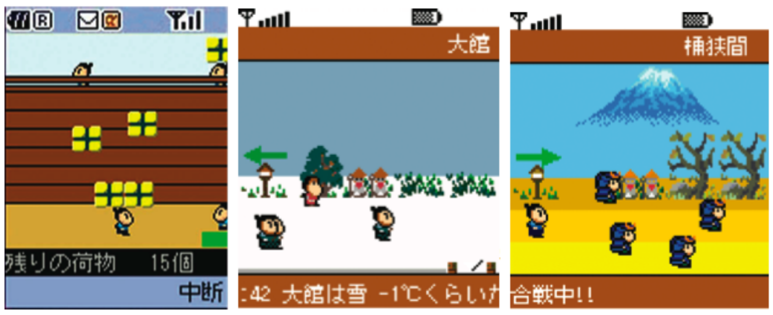
5.5.1 Samurai Romanesque

Samurai Romanesque is a persistent-world samurai themed game in which the player take on the role of a samurai in ancient Japan. He or she has to build a character from scratch, training to enhance his skills in three areas; mind, sword and physical fitness. The training takes place through various mini games, all of them thematically linked to the main game but also distinctly different in terms of game mechanics. An example of such a minigame in which the player has to move goods at a wharf is shown in Figure 5.12(a).

The game takes place in a huge world - there are a total of 1,000 villages and 3,000 other locations the player can visit. To see all of the sites in the game would take six months of continuous play [9]. The game also includes real time weather data from the area the player is currently in, enhancing immersion by including real-world data into the gameplay. A player moving in two different kinds of weather can be seen in Figures 5.12(b) and prestudy:games:samurai:c.

The multiplayer aspects of Samurai Romanesque are realised by allowing every player to have a different set of goals or motivations, encouraging variation in player behaviour. All actions in the game affects other variables, and often things that are not expected can occur as a result of a player action.

In addition to classical role playing mechanics (fights, duels and warring factions), Samurai Romanesque also has a couple of innovative tricks up its sleeve. Firstly, the spontaneous



(a) (b) (c)

Figure 5.12: Screenshots from Samurai Romanesque (from [9])

wars can suddenly break out, and all players who are present at the site of the war (in-game, not physically) will need to report for duty (log on to the game) within a set period of time. This can often lead to groups of many thousand players cooperating to reach a common goal, for example defeating an enemy stronghold.

Another innovative concept in Samurai Romanesque is the inclusion of romance (hence the name). At any time, a samurai could meet a woman that needs his help escaping a bandit. If the player chooses to help her out, he will have a chance on engaging her in conversation. If he then plays his cards right, there is a chance that the woman eventually becomes his wife.

Being married does not change anything within the game itself - but it enables the player to have a son and to continue playing after the death of his avatar without loosing his achievements. This is done by the player taking on the role of his own son that has inherited all his traits and riches.

Samurai Romanesque is, as far as we have been able to find, the most successful multi-player mobile game on the market, with the ability to handle more than 500,000 concurrent players from the more than 10,000,000 persons with compatible hand sets.

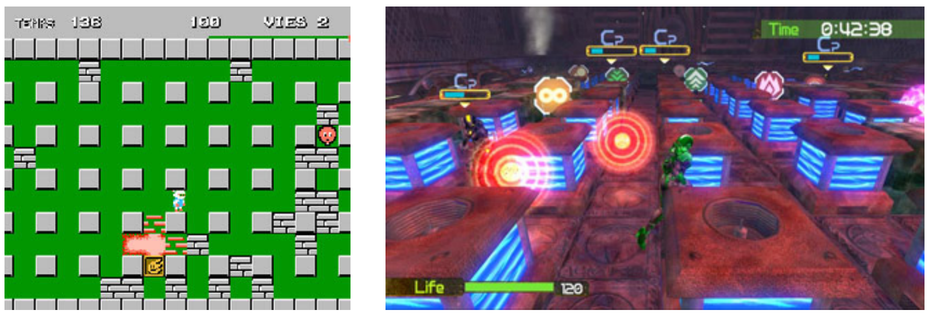
Chapter 6

Technical Considerations

Although this project's final product is a set of five game concepts, we believe that it is important to give some consideration to the most important technical issues that have to be tackled in order to successfully create any MOOSES game. In the following chapter we will outline these, without going into excessive detail on every count.

6.1 Point of View

A very important aspect of MOOSES games is the need for an objective viewpoint. Most multiplayer games, regardless of whether they are online or co-location games, utilize different screens or different screen areas for the different players, allowing the game to display the game graphics from a viewpoint that is relative to the current situation of every player.



(a) Objective (b) Subjective

Figure 6.1: Subjective and objective viewpoints

A MOOSES game, however, needs to maintain an objective viewpoint at all times, meaning that all the players will see the exact same picture at all times. This can be compared to the situation in very simple multiplayer games such as PONG, or some newer games such as fighting games, where the players also share a common playing field.

An example of the distinction between objective and subjective points of views can be seen in Figure 6.1, where Figure 6.1(a) shows the original Bomberman game for the Nintendo Entertainment System, released in 1984. In this game, the playing field is seen from above, and all information was available to the players at the same time. In Figure 6.1(b), however, we see a new version of the same game whose point of view has been changed to a subjective one with the camera moving relative to the player. Obviously, such an approach to graphics could not work in a MOOSES game.

The implementation and design of games with objective viewpoints is not a problem with games in which there are only a very few players and relatively simple movement patterns, but in MOOSES games where there can be upwards of fifty players at once it is definitively a factor that needs to be taken into consideration at the very first stages of game development. If this is not done, it is very possible that one might end up having to face insurmountable problems later on as one tries to implement games that are not fit for implementation with an objective viewpoint.

Another aspect of the objective viewpoint problem is that while other games often have the luxury of being able to display player-specific information directly onto the game screen, a MOOSES game can only show the most basic information this way. This means that any information that needs to be kept secret to some players has to be conveyed by other means, most probably by utilizing the personal screens that the players would have if they use their mobile phones as controllers (see Section 7.2, page 48).

6.2 Automatic Zoom

Although all players need to see the same thing as well as stay on screen at all times, there is no demand that the entire playing field is always visible. If a game sees all players crowding in on a single part of the screen at some times, the game could automatically zoom in on this area to enlarge the player avatars and ease recognition. This would not break the rule of having an objective viewpoint as long as the zooming is only carried out to the extent that all players are still present on screen.

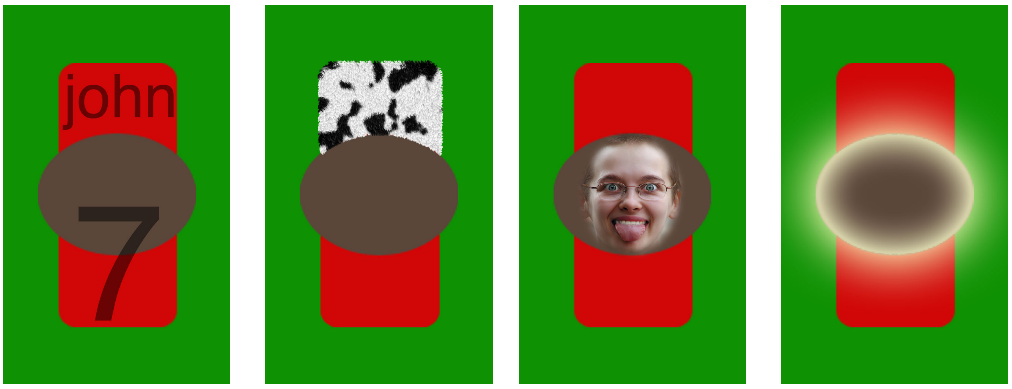
It is even possible in some games that the screen could zoom in on a section of the screen and simply move the players who would end up outside the screen to the edge of the area shown. However the use of this method is very dependent on having a game to which it fits, and it must be used with great care as not to confuse the players unnecessarily.

Another possible way of enhancing the view of the game action can be found in games where the action happens sequentially, and where the game can show only the parts of the playing field where something happens at the moment - thus having the zoomed area moving around screen until all relevant action has been shown. The War Game could for example benefit from this display method, showing one battle at the time until all wars have been fought.

6.3 Recognizing Players

In MOOSES games a major technical and game play related concern is the question if players are able to recognize their avatars on screen as the number of active players approaches, and maybe even exceeds, fifty at any one time. This is a very new problem that is not existent in other multiplayer games, due to their limited number of concurrent players or their lack of an objective viewpoint.

Therefore, the question of how to recognize individual avatars is one that will have to be handled in any MOOSES based game. In the following section, we will look at some possible solutions that could be applied to various types of games, as well as look at how this could be applied in our game concept from Chapter 11, The Football Game. Of course, the methods mentioned here can be used in combination to make avatar recognition even easier for the player.



(a) Labelling (b) Colour and pattern (c) Picture (d) Forced focus

Figure 6.2: Methods of ensuring player/avatar recognition

6.3.1 Labelling

The most obvious way to solve the problem of player identification is simply to label each player with a nickname, a real name or a number. This ensures that any player can look at the screen at any time and find himself, but the process potentially includes quite a lot of searching and reading, something that might jeopardize the player's ability to simultaneously keeping up with the game play.

The players could be labelled with automatically generated numbers (technically and related to user interaction this would be the easiest solution), names chosen by the game (for example in a game with predefined roles) or by a name or nick that they upon decide themselves. An example of a possible result of labelling players in this way can be seen in Figure 6.2(a)

6.3.2 Colours or patterns

In addition to having a different label on every player, one can possibly apply different colours and/or patterns to every player or to groups of players in order to separate them into different teams. Also, if there is not to many players and they are not separated into teams, it should be possible to separate the players solely by having a distinct colour per player.

Similarly, the use of different patterns on avatars can serve the same purpose as different colors, adding another level of difference between different players, enabling a game to facilitate more simultaneous players with a smaller number of different colours.

Examples of this method of differentiation can be seen both in the example graphics in Chapter 10 (Figures 10.1 and 10.2) and in Figure 6.2(b). Although this is a very effective and intuitive concept, it is very important to be careful with its use as the number of player increases, as the separation between the different possible colors decreases greatly when more colors are used. The inclusion of different patterns would enable a game to use colours with more players. Also, the use of only colors and no patterns carry a risk of alienating players that are colour blind - an accessibility concern that will have to be handled in actual game implementations.

6.3.3 Pictures

As an alternative to labelling the avatars with names or numbers, a MOOSES game can instead utilize pictures of some kind to identify the different players. This could be done by allowing players to choose from a predefined pool of pictures (as it is done in some games today), or by allowing the players themselves to upload their own pictures to the game. In a game controlled by means of mobile phones, one could even utilize the cameras embedded in most modern mobile phones to allow players to upload an actual picture of themselves for use as an avatar identifier. An example of how this could look can be seen in Figure 6.2(c).

However, in allowing players to upload their own avatar pictures, one inevitably invites some players to try and sabotage the system by uploading unfitting or obscene pictures, so the inclusion a function like this would probably need to also include some kind of control and moderation of the uploaded pictures.

6.3.4 Forced Focus

In addition to using one of the methods above (or several of them in combination), a game with many players on screen at any time should include a system that allows any player to draw attention to his or her avatar at will. This system could for example be implemented by having a special button on the controller that triggers some reaction on the relevant avatar, enabling the player to make his avatar stand out for a moment if he loses control of where it is located on screen.

For example, having a button that makes an avatar light up (Figure 6.2(d)) whenever it is pressed would greatly ease the task of looking for it during game play, even though the avatar also have some other kind of identification.

6.4 Hardware Considerations

It is only now within the last couple of years that standard off-the-shelf hardware have become sufficiently powerful to be able to run graphical simulations in the kinds of resolutions that are needed to successfully create immersing environments for players in a game played on a cinema screen. Today, a very-high-end gaming PC system will be able to run highly sophisticated graphics in a large format, and in the near future even higher resolutions will probably be feasible on standard hardware.

The digital projector used at Nova Kino in Trondheim, the test system for the big screen gaming project, is able to run natively in two different resolutions. The lowest one, 1920 x 1080 pixels, are supported by today's standard gaming equipment, both PCs and the major consoles. Its other resolution, at roughly four times the pixel count (4096 x 2160 pixels) is more of a challenge for today's gaming systems, but high end professional graphics controllers are already able to effortlessly display such resolutions, and as all other high end system performance gains this too will eventually trickle down into off the shelf hardware. For instance, some newer graphics controllers are able to output to two screens at 1920

* 1080 today. Combined with operating system support for multiple graphics controllers (Microsoft Windows already has this feature today, as does most other major operating systems). Thus, it seems inevitable that hardware development will soon catch up with the cinema digital projectors and enable multiplayer gaming at full resolution using nothing but standard off-the-shelf equipment.

6.5 Basic Software Architecture

Although all the separate game concepts in Part III will each have sections on their own specific system architectures, the basic premises of MOOSES games calls for a common base system architecture for all the games. Therefore, we will take a brief look at the main modules that any MOOSES game would have probably built upon. Figure 6.3 shows a very basic overview of the main modules that would have to be implemented.

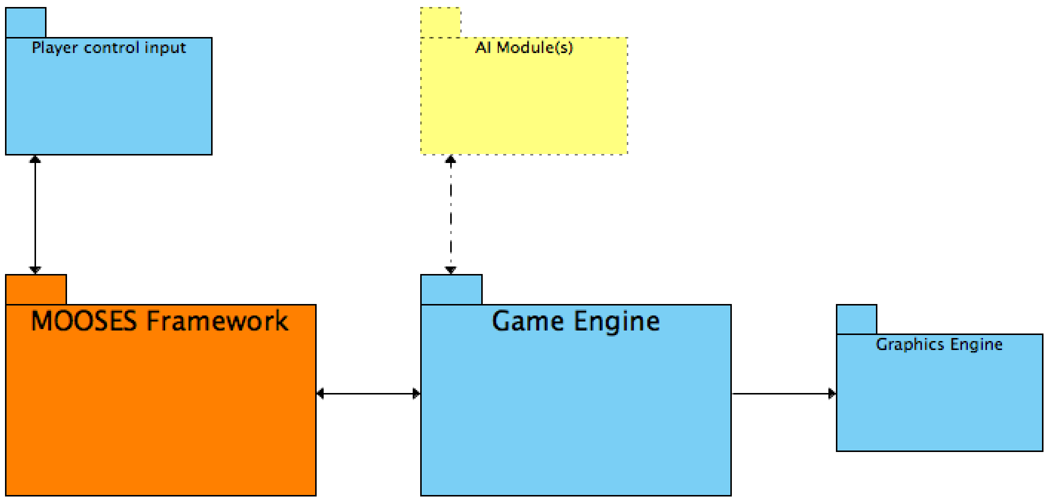


Figure 6.3: Base game software architecture

6.5.1 Game Engine

Very broadly, the game engine can be said to be the module representing game itself. This system module is the one that handles most of the game specific functionality as well as all the communication between the different modules. Therefore, it is very probable that this module should be separated into two parts in an actual implementation - one highly reusable part that handles MOOSES-specific and general communication, and one that handles all the functionality that is specific for any particular game. However, in this report we will treat the game engine as one singular module, as highly implementation-specific details are out of the scope of this report.

The following list outlines some of the functionality that would usually need to be present in the Game Engine module in a MOOSES game:

Generic Functionality:

game initialization This part of the game engine is responible for initializing the other components and setting up communications between them.

intermodule communication After all the submodules have been initialized, this module maintains communication between them and ensures that every game module behaves as expected.

framework communication This module is the interface that supports communication between the game code and the MOOSES framework. In addition to in-game controls, it also handles requests such as player adding, score calculations and saving and other events on which the framework has to be part of the loop.

Game specific functionality

game code What is perceived as the game itself from the players point of view. This module handles all game-specific code and / or handles the modules that does this. It also communicates with the graphics engine and with the generic functionality modules to utilize framework functionality.

game module communication Handles communication between different parts of the game code, such as AI modules, score modules and world representation modules

world representation Keeps an authoritative representation of the game world at all times. This is the system element that all changes in the world are written to and from where the arbitration, scoring and graphics modules gets their data.

6.5.2 Remaining modules

Here we will briefly describe the other modules shown in Figure 6.3. Of course, specific games could require other modules to be included, but these would be specific to those games and are therefore not covered here.

player control input

The module that handles input from the players. Could be generic for many games, but could also require custom functionality in some games.

AI module(s)

If a game is to have any artificial intelligence (such as computer controlled players) it needs AI functionality. This is not necessary in all games,

graphics engine

The graphics engine handles all graphics generation and output to the projector or screen. It can be generic or specific for a game. Graphics engines can also often be successfully licenced from third parties.

6.6 Connecting Controllers

As we are dealing with a potentially large number of concurrent controller connections, it is important to have a system that are able to handle this whilst still retaining a high level of responsiveness. In [10], the emphasis is placed on connecting mobile phones via a Bluetooth interface, but in this section we will also look at other methods of connection that would be able to connect a sufficient number of players.

6.6.1 Mobile phone via Bluetooth

As mentioned above, the current test implementation of MOOSES utilize mobile phones connected via Bluetooth as its controllers. This is a well suited and satisfactory solution for most kinds of games, but Bluetooth as a controller connection method also has some clear limitations.

Advantages

The following is a list of notable advantages of using Bluetooth as a data carrier between handset and game system:

availability Because more and more modern phones feature Bluetooth, it is a control method that includes very many potential users, and that will continue to include more and more as time passes.

compatibility Bluetooth is a clearly defined standard that should work with any supported handset without special adaptation by the game or framework developer.

Limitations and disadvantages

Although Bluetooth is clearly well suited for this application, there are also some notable limitations, which we will describe below

a limited number of concurrent connections A Bluetooth master hub is only able to natively support up to seven separate controllers. This means that one has to use several separate controllers to facilitate more players. However, there exist controllers that combine more than one master unit in one controller, effectively creating a workaround for this problem.

network noise This problem is intimately connected to the previous, as an increased number of concurrent connection in a limited space creates interference between the units. However, this should not create a very large problem as long as the number of connections are within the range that we are interested in for this study [11].

limited speed With current applications, it is not probable that Bluetooth connections will cause any delay in data transfer between controller and server, but if future games grow to demand higher data throughput it is very possible that the limited bandwidth available could prove to be a bottleneck.

6.6.2 Wireless networking

As an alternative to Bluetooth, one might choose to utilize the ubiquitous Wlan technology. This method of connection shares the advantages of Bluetooth (although it is not as common in handsets yet), but largely eliminates the limitations on concurrent connections and bandwidth. In addition, wireless LAN connections use much more power than connections using Bluetooth.

6.6.3 Other wireless

Most retail wireless game controllers use some kind of proprietary radio interface in their communication with their host. This technology is good for smaller applications, but it is very limited when it comes to concurrent connections and functions like controller screens and similar bandwidth demanding features. In addition, proprietary connection means that controller technology would have to be customized for MOOSES gaming, causing increased costs without really adding any value.

6.6.4 USB connection

When considering wired connections, the choice that first comes to mind is USB. Today, virtually all peripherals that are connected externally to a system (and many that are connected internally) use this standard. USB support both many devices (127 per root hub with virtually no limitation on the number of separate root hubs) and high bandwidth (480Mbps). Apart from the obvious disadvantages of being a wired standard, USB is therefore well suited for such applications as MOOSES games.

6.6.5 MIDI

Although it is a very old standard, MIDI has some factors speaking for it in this setting. Firstly, MIDI supports a very large number of addressed units (127 units on a bus, each with over 100 different channels available, and a large number of addressable keys on each unit). Although there are some limitations in the amount of data that can be sent simultaneously, we believe that MIDI is well fitted for utilization in game controlling. However, an implementation would meet the same limitations as a proprietary wireless one, as there is no readily available game controllers that use this standard, so custom controllers would have to be made, along with software support. Therefore, we believe that USB is a better choice in most situations.

Chapter 7

Control Schemes

When aiming to keep the players in control of a game with as many as fifty different participants one needs to pay close attention to the different possibilities that exists in the field of avatar manipulation and game control. It is very important to give the players an easily understandable control system for all games, ensuring that even player with little or no experience in playing on traditional systems can enjoy playing the games. Here, we will look upon a few different potential control methods for MOOSES games, emphasizing mobile phones as the most realistic alternative both now and in the future.

7.1 Traditional Game Controllers

Most regular computer and video games rely on a multi-button hand controller such as the Dual Shock on the Playstation series or the XBox 360 controller, as seen in Figure 7.1. These controllers allow the player to directly manipulate the on-screen action via a large number of buttons. The Dual Shock III, for example, has 16 different buttons, some of them pressure sensitive, in addition to two analogue sticks and an accelerometer measuring three dimensional motion of the controller itself, allowing for a very high degree of control sophistication.

These controllers, however sophisticated, are not ideal for most MOOSES settings, as they are created for a multitude of different games, up to and including very complicated

(a) XBox 360 (b) Playstation 3

Figure 7.1: Controllers for XBox 360 and Playstation 3

games with an enormous amount of control freedom. In a game, it is very important to keep the controls as simple as possible, thus enabling simple and understandable manipulation of each players on screen avatar, regardless of his or her prior eperience in game playing. Even if one were to make more advanced MOOSES games than we advocate in this study, it is very probable that the complexity would need to be based upon other factors than control freedom, as this is something that it is near impossible to translate from home gaming and onto a game with a massive amount of players on screen at any given time.

7.2 Mobile Phones

In the MOOS report [10], emphasis are given to the control of multiplayer games by means of mobile phones. In many ways, mobile phones equipped with Bluetooth or Wlan wireless networking are ideal for these kinds of applications. Firstly, basically all phones share a given set of buttons, namely the numeric keyboard with one button for each number from 0 to 9, in addition to * and # buttons. Also, almost all phones have some kind of a navigational pad or joystick. In addition, all new phones with Java support have a screen that enables additional information to be sent to a player, augmenting the objective and common information that are relayed via the main screen.

7.2.1 Control methods

When using a mobile phone to control games, there are a few major control schemes that are sensible to use and that users will be able to understand. Below, we will explain what these schemes are and show possible keyboard layouts for each of them.

Race car control

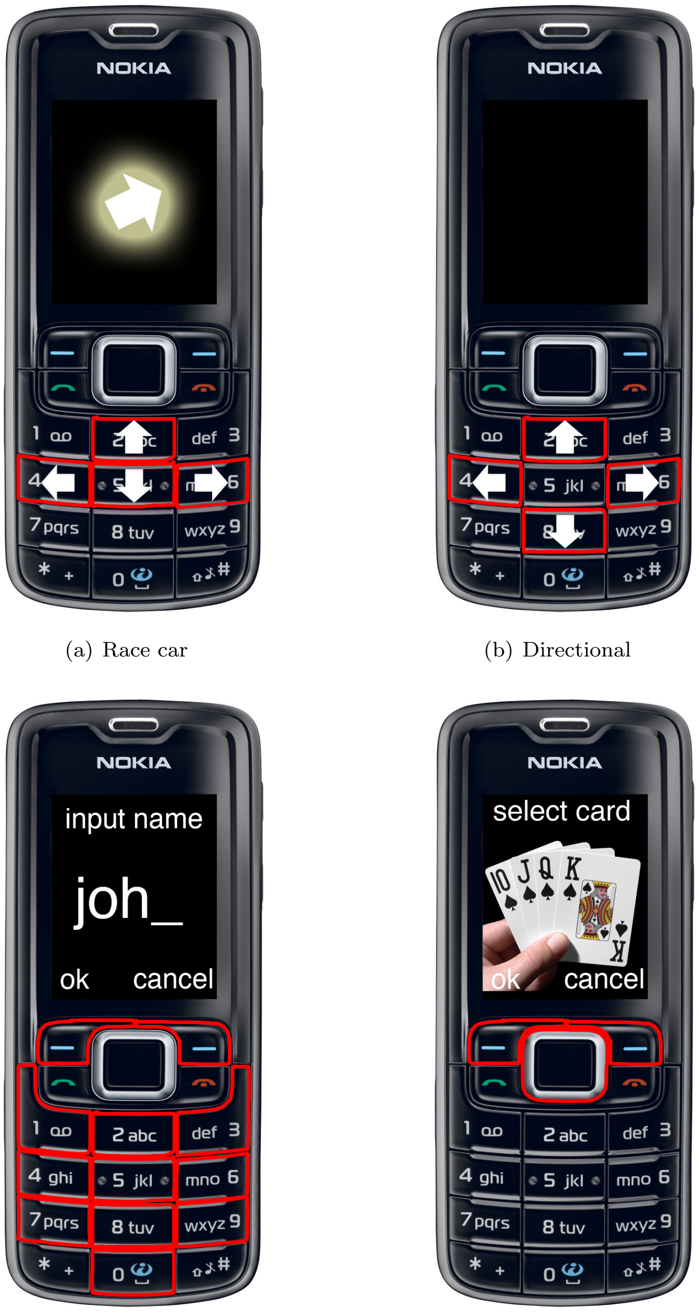
As shown in Figure 7.2(a), the Race Car control scheme tries to mimic that of a car, with buttons for acceleration, deceleration and for turning to each side. This is a very simple and intuitive way of controlling a game, and it is very well suited for simulations in which the player are in control of a real world object that moves around on the screen and that has real world physical attributes such as weight and speed - meaning that any input from the user serves as a correction to the previous speed and direction.

Directional control

This control method, as shown in Figure 7.2(b), is fairly similar to the Race Car controls in that it has four directions. However, as the controls are now absolute and instantly changes the direction of the controlled object, it is more suited to games that feature item or menu selection as their main way of controlling the action. Also, it is well suited to real time action games in which movement are direct without regard to physics.

Numerical control

For games that require the player to input numbers and or text in any substantial quantity, or to select from a ordered set of alternatives, the direct mapping of the numeric keys on the phone (as in Figure 7.2(c)) to the game controls are probably the most sensible way to



(a) Race car (b) Directional

(c) Numeric (d) Screen based

Figure 7.2: Different control schemes

implement controls. Games that would be suited for this control method include quiz games similar to Buzz (as mentioned earlier) and other trivia based games.

Screen based control

In addition to the three aforementioned control methods that are based upon the direct mapping of the keys on the handset to functions within the game itself, one could also implement a control system that bases itself fully upon the fact that the handset has a screen of its own. This could for example include having the user selecting secret alternatives on his own screen, choices that would then be reflected on the main game screen after the choice is made. Games that would take advantage of this could for example be, as shown in Figure 7.2(d) and discussed further down, card games that base themselves upon secrecy between players.

Built-in menu augmentation

Many new mobile phones supporting Bluetooth have a built in system that allows externally connected systems to actually append their own menus to the built-in menu system of the phone. For very simple games (or to select clients to download before playing) this could be a good controller choice. However, as implementations vary between different phone systems this could also potentially lead to confusion and frustration amongst the players, so it is doubtful whether this technology would be a good choice for MOOSES games.

7.2.2 MIDP 2.0

As a thorough look at the technical implementation of the mobile client software is not necessarily within the scope of this study, we will not go into any depth about MIDP 2. However, it is important to note that by demanding that user phone clients support this Java implementation one gains a set of important standard features such as graphics, animation and sound support. In addition, and most importantly, MIDP 2.0 defines a set of standard buttons that needs to be mapped on the phone, meaning that the creator of a game are able to confidently address keys on any compatible device without needing to worry about the layout of the device itself.

7.2.3 Personal screen

Another important feature that mobile phones allow game creators to utilize is the concept of a private screen available to each player. In games where all players share an objective viewpoint, all players will have access to the same set of information. In quite a few games this is OK, but in many games this would put severe limitations on what kind of gameplay that could be implemented. For example, we could imagine a game like for instance Poker implemented as a MOOSES game - the inclusion of private screens for each player would be absolutely necessary. As we described in Chapter 4, the MOOSES framework supports such functionality, and this is also utilized in some of our game concepts.

7.2.4 Further opportunities and future technologies

As mobile terminals grows more and more advanced and as the MIDP implementation grows to take advantage of this, it is conceivable that one will become able to take advantage of

Figure 7.3: Nintendo Wii Remote

an increasing number of next generation features featured in the handsets. For example, it is possible that the camera that are included in almost all new phones nowadays can be utilized in the games. The MOOS project mentions the possibility for players to use their own picture as an element in the game (for example to mark their avatar as theirs), but we can also envision other possible uses for cameras that far exceed this level of ambition. For example, it may some day become possible for the player to use his camera as a live input controller that utilizes live footage to control the on screen action. Games similar to this have been made available for handsets already, but the technology is currently far from mature enough. Other technologies that might have an impact include gyroscopes and accelerometers. These would allow a player to control games by moving the handset in space, similarly to the way games are controlled on the Nintendo Wii today. There have been made phones that have included at least accelerometers, but only at an experimental level. Some of these technologies will probably become real and relevant to gaming. However, as "which" and "when" are only possible to speculate upon, we will refrain from utilizing such technologies in our game concepts.

7.3 Other Controller Types

In addition to off the shelf game controllers and mobile phones, there is a virtually unlimited number of different methods that one can use to control MOOSES games. Below we will mention some of the possible controllers that could be used, but it is of course impossible to create an exhaustive list.

7.3.1 Wii Remote

The Nintendo Wii uses accelerometers, infrared and a gyroscope to translate player and controller movement into control actions.

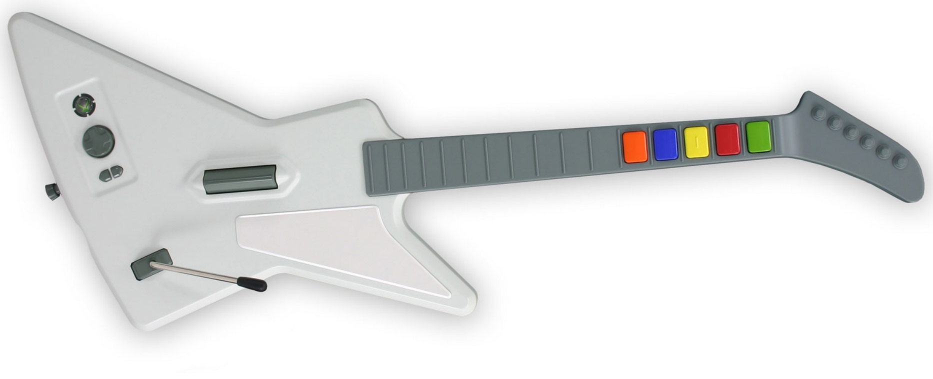


Figure 7.4: Guitar Hero controller

As shown in Figure 7.3, the Wii Remote looks like a normal remote control with very few buttons. However, it is a highly versatile controller that are suited for many different kinds of games, and as it uses Bluetooth as its host connection method it is possible to use with other systems than the Wii without problems.

7.3.2 Game specific custom controls

Some games, like for example the Guitar Hero series and the Dance Dance Revolution series require very specialized custom controllers to work as intended.

For example, Guitar Hero (Figure 7.4) uses a controller that looks like a full size electric guitar, enabling an extremely good mapping between the controller form factor and its utilization in a game. However, it is obvious that such an approach has only limited use in for example a cinema setting, as the re-usability of the controller for other games are virtually nil, thus effectively barring the use of several different games at once and making the replacement of games potentially more expensive. That being said, in an environment where one is only interested in implementing a specific game and where the resources to do it is available, a custom game specific controller would of course be the optimal solution.

7.3.3 Motion sensing / Video input

As the developers of Newsbreaker (Section 8.2, page 57) and Ghost in the Cave (Section 8.1, page 55) have done, it is now fully possible with new technology to use video cameras to film the players and to use this as control input for a game. An implementation of this requires advanced image recognition software to be created, but offers a very intuitive (if somewhat unsophisticated) way of controlling big screen games. The big disadvantage of this solution is that it is extremely difficult, if not impossible, to make the system recognize individual players in a consistent manner.

7.3.4 Custom in-seat controls

As a solution in between the high versatility of using mobile phones as controllers and the binding of custom controllers to a specific game, there is the possibility to create a domain specific controller that are versatile enough to facilitate several different functions, but still specific to the MOOSES application.

An example of such an approach can be seen in some newer aircraft entertainment systems, where each seat features its own wired controller that are custom made to fit perfectly with its application. In a large scale installation of a MOOSES system we believe this to be a viable solution as an alternative to using mobile phones, as it lowers the demand on the technical proficiency of the player.

Chapter 8

Existing Big Screen Games

As mentioned in the introduction, there is not very many existing implementations of games that resembles what we're describing in this report. However, some prior art does exist, and in this chapter we will describe the most important example that we are aware of.

8.1 Ghost in the Cave / EPS

In the period ranging from 2003 to 2005, students and researchers at KTH[1] in cooperation with various other research bodies, implemented a game named *Ghost in the Cave.* Although this was more of an experiment in user interaction than it was a traditional game, we feel that its design and structure of the project merits inclusion in this report (and we will continue to call it a game from here on).

8.1.1 About the game

Ghost in the Cave is a game that explores the possibilities for creating interaction between multiple players and on-screen avatars. Thus, the basic premises for the game is somewhat similar to the kind of games we try to create in this study.

The basic mechanics of Ghost in the Cave is as follows:

Two teams, both consisting of N players divided into a leader and a group consisting or the rest of the players compete in a race to make a fish attain the same state of mind (see Figure 8.2) as a ghost inside a cave. To achieve this, the player leading each team has to move or sing in a way that the game engine understands as a sign of him being in a specific state of mind. In addition, he or she has to use directional sound or swimming movements to steer the fish into the cave to find the monster.

In addition to this, the teams behind each player control the background music by, respectively, making rhythmic sounds and movements. The more each team moves or sounds, the faster their fish is able to move.

Although this game does not rely on traditional game elements such as goals and scoring, [12] states that

> Finally the use if a theatre and game metaphors influences the designer's approach to the user. In an interactive game environment the user becomes a

[1]Kungliga Tekniska Högskolen, Stockholm

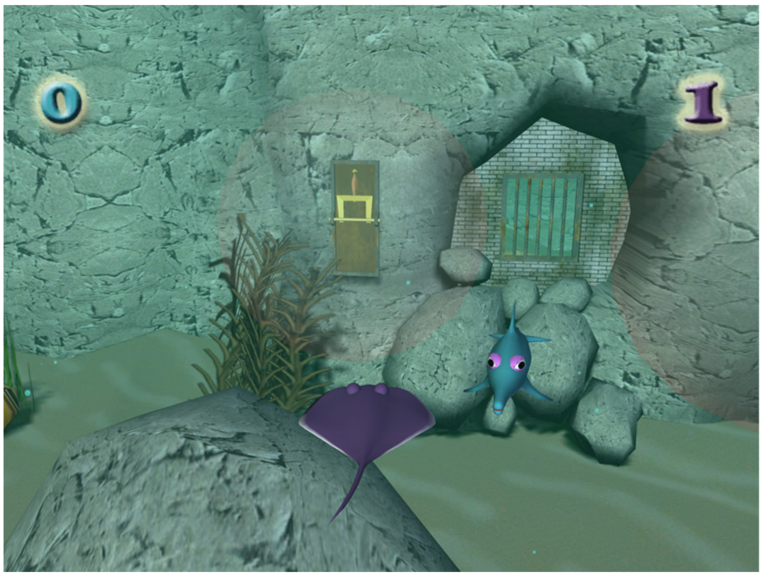


Figure 8.1: Screen shot from Ghost in the Cave

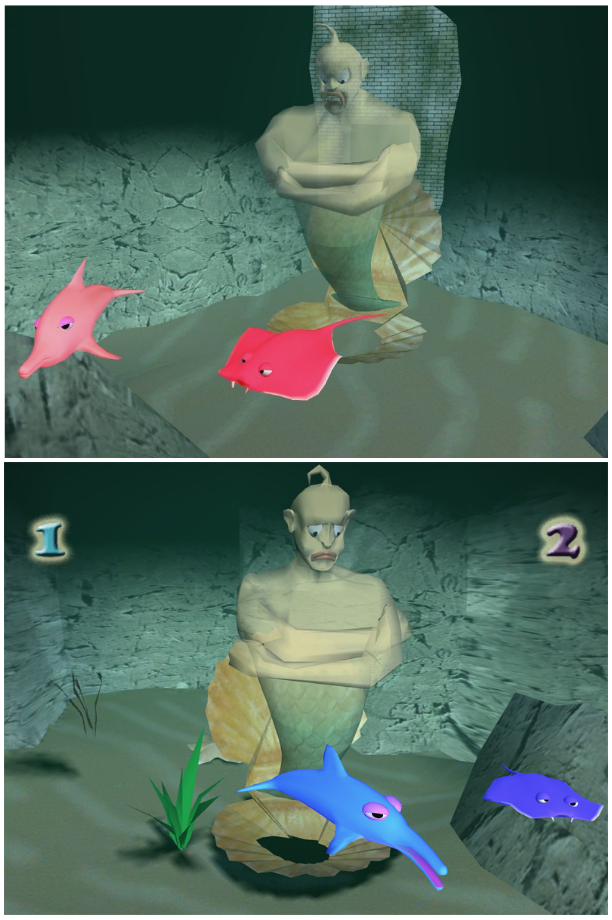


Figure 8.2: Moods from Ghost in the Cave

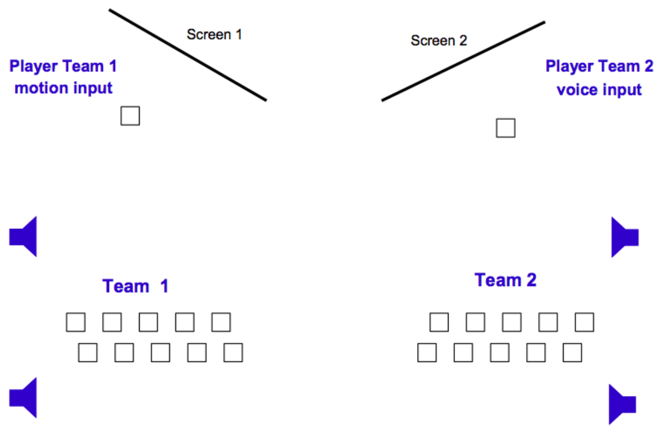


Figure 8.3: Player setup for Ghost in the Cave. Taken from [12].

> member of an audience, a participant or a player. They are made participants and become co-writers, co-directors and co-actors with a possibility to influence the outcome of the game.

This describes a situation that is very similar to what we are trying to accomplish in this study, and we therefore belive that many of the findings of the EPS group also applies to our study.

A more detailed description of Ghost in the Cave can be found at [2].

8.2 Newsbreaker

In May 2007, the American web portal site and TV channel launched the game *MSNBC Newsbreaker*. The game itself is a rather traditional imlementation of a Breakout game, but the controls makes the game interesting with respect to the MOOSES project. According to [3], the game is controlled by an entire cinema audience as a single player, with the crowd's collective movement input into the game via image recognition software running on a consumer grade Mac Mini. As the article states:

> NewsBreaker Live will have moviegoers of all ages swaying, swinging and rocking in their seats by utilizing groundbreaking motion-sensor technology dubbed "CrowdGaming," which tracks the entire audience's collective movement to control the game. Working together as a human joystick, the audience will move in sync to smash up msnbc.com's colorful brick spectrum of news with a bouncing ball and paddle. As each brick breaks, real-time headlines from msnbc.com fall. The audience then accumulates points and knowledge by using the paddle to capture the headlines before they drop off the screen while simultaneously maneuvering the paddle to keep the bouncing ball moving to break new bricks.

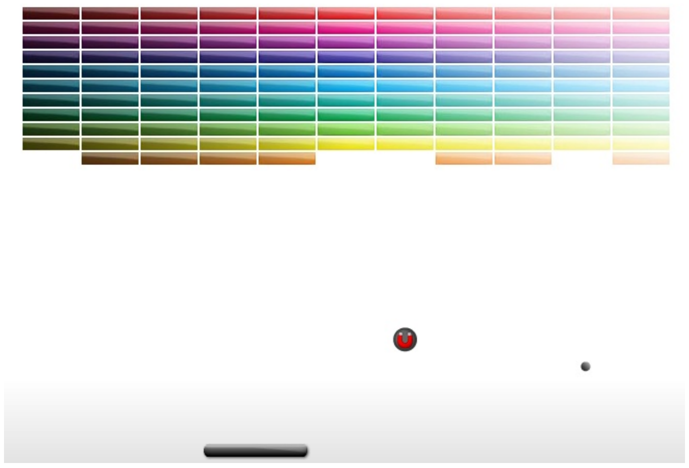


Figure 8.4: MSNBC Newsbreaker screenshot

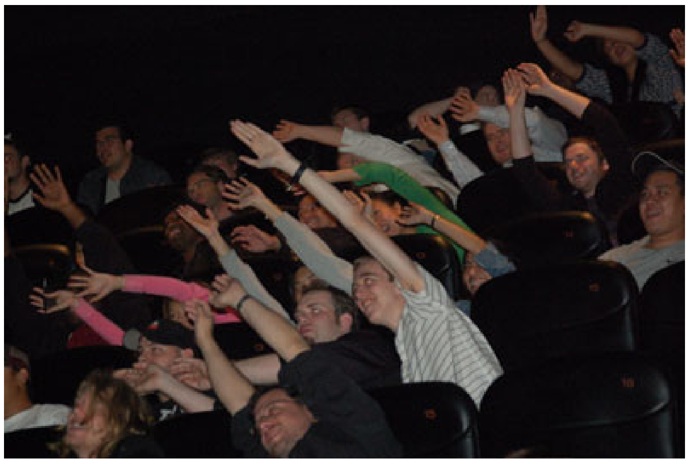


Figure 8.5: A cinema audience playing MSNBC Newsbreaker

8.3 Summary

As the reader can see, the two other main one screen multiplayer games that we have been able to track down are both based upon the entire audience participating as groups, not as individual players. This makes us fairly certain that the MOOSES project is trying to accomplish something rather unique, and that the result of the project will probably represent something completely new.

Part III

The Games

Chapter 9

The Music Game

In this chapter, we will describe a sound and rhythm based game that allows players to act together as a group competing against other bands.

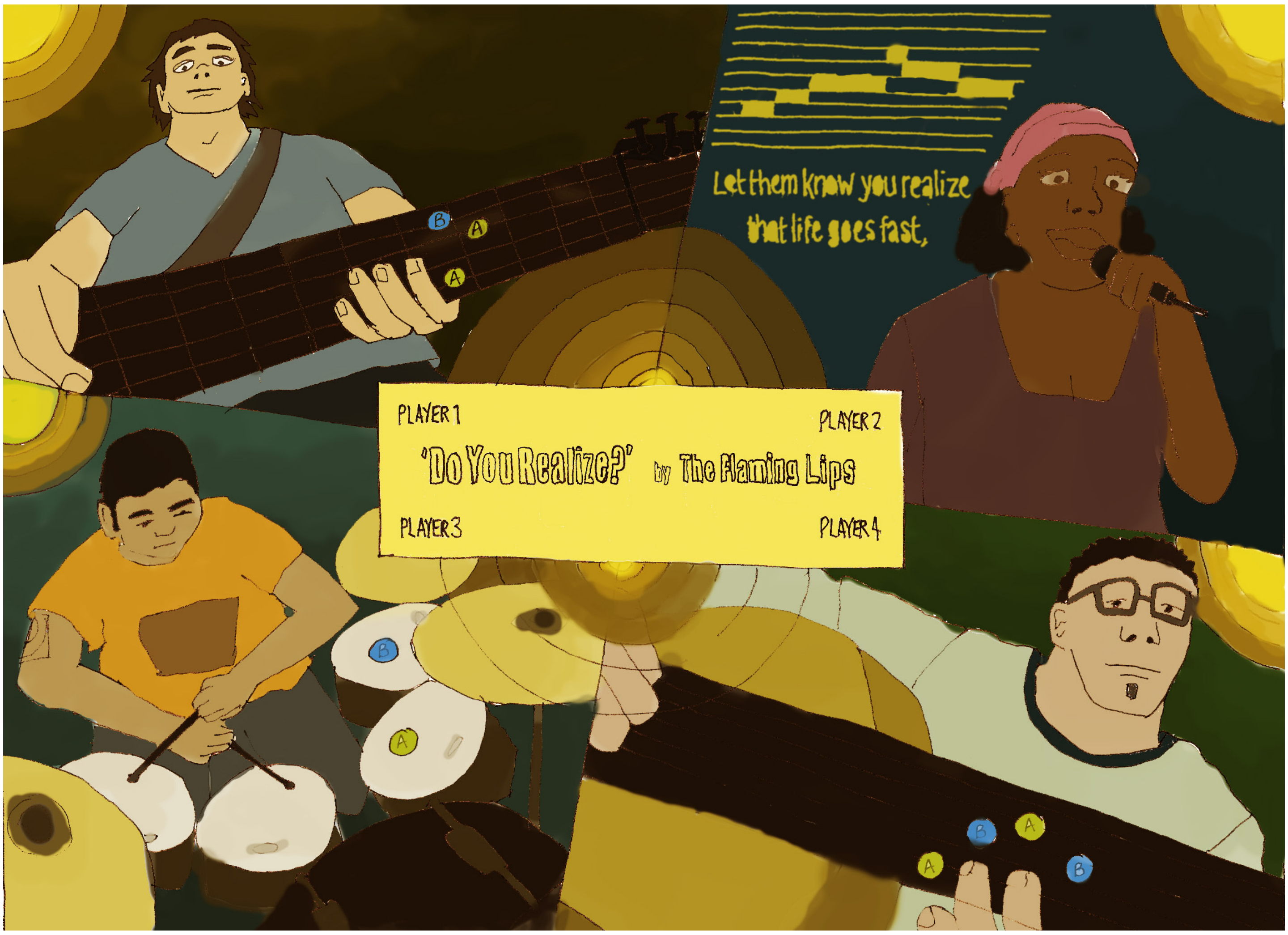
Table 9.1: the Music Game infobox

Game Information	
Central concept	Different bands competing to stay in sync/tune
Number of players	20 (up to five bands of four persons each)
Controls	Traditional game controllers or custom controls
Play mode	Team vs. Team

9.1 Introduction

Many players, both causual and dedicated, have in the later years engaged in playing games based upon musical interaction between the player and the game. Up until now, these games have been based upon single-play or sequential single-play functionality more than upon concurrent cooperation between several players, something that is presumably based upon difficulties that arise when trying to engage several players in this kind of interaction, difficulties that could be as simple as a lack of space and the danger of different players interfering with each other playing.

However, here we will outline a game that includes up to 20 players at once, all of them playing a part a simultaneous musical team contest.


Let them know you realize
that life goes fast,
PLAYER 1
PLAYER 2
'Do You Realize?' by The Flaming Lips
PLAYER 3
PLAYER 4
B
A
A
B
A
B
A
A
B

Concept art for the Music Game

instructions directly on the instruments. All instructions are shared by players having the same role.

9.2 Concept Outline

Games based upon rhythm and tone can be very immersive for the player and has been proved to function very well in social settings. We wish to take this acknowledgement a bit further, and to allow many players to cooperate in teams within a superset of the traditional game mechanics found in these music based games. Thus, we wish to allow groups of players (”bands”) to play together as one, competing against other, similar teams. This is a game setup that scales very well, and that we believe are well suited for use in a theatre setting.

9.3 Game Mechanics

As is also seen in many single player music games, the main part of the screen are filled by a stylized image of a group of artists. For each different person in the ”band” there is also a specific section of the screen instructing the player as to what she should at any given time in order to successfully fulfill the objective of the game, which is to somehow be in sync with the artist on screen. This can take the shape of singing, drumming, playing the guitar or any other instrument that are included in the game.

As the song progresses (the game should include a large database of different songs, and this database needs to be updated regularly to keep the selection of content fresh), the players are continuously rated as a team based upon how well they are able to keep in sync with the instructions on screen, and they are always able to see how well they are doing in comparison with the other bands currently playing. This is done by having a set of note bar inspired graphics passing over the screen as the music progresses, enabling the players to see what they need to do in order to succeed in following the song. Another possibility is to put instructions directly onto the instrument of the on screen band members who would represent all the teams. See the concept art for this game for examples of how this could look.

As we see it, the following role setup is ideal for a team of four players. If more players are added, one could simply add additional players to any of the four existing roles:

Singer The player has a microphone and sings along with the original lyrics of the chosen song. The performance is graded on its rhythm and its degree of similarity with the original.

Guitar High frequency of tones (buttons) that needs to be hit. Some leniency should be allowed in the judging of rhythm, as the focus of this instrument is on hitting the correct buttons in the right order.

Bass Compared to the guitar, the bass has a slower pace, but the timing of the button presses is much more important. The players needs to have virtually perfect timing to get a high score, but this should be attainable due to the somewhat simpler patterns.

Drums As in the real world, the drummer in a group needs to keep the rhythm of the band. The role of the drummer is therefore to hit his buttons based on repetitive patterns that undergo slight changes as the song progresses. The grading rules for the drummer should be somewhere between those of the guitarist and bass player.

The group gains a score as a whole based on their collective performance, but it could also be possible to rank the players within a group and to make cross-group rankings based on different instruments, but this is aspects that would have to be play-tested before one knows if they add value to the experience.

9.4 Controls

As we state in Table 9.1, this game could be controlled both by custom game controllers or by standard controllers, which in this setting would probably mean mobile phones. Custom controllers would be very expensive and limit the game with respect to future expansions and changes, but would also increase the player's feeling of being one with the music greatly. On the other hand, using mobile phones would simplify the game design greatly, and we believe that it would not detract too much from the experience. We believe that the most suited mobile phone control scheme is the directly mapped ”numerical control” scheme shown in Figure 7.2(c).

One exception (at least until the phones becomes fast enough) from the fitness of mobile phones as controllers in the Music Game is for the singer in the band, for whom the controller must be able to transmit sound into the game engine. This player would probably need a wireless microphone of some kind, but as these are readily available and cheap we do not see this as a problem for the implementation.

9.5 Software Architecture

While this game is very well suited for use within the MOOSES framework, special considerations has to be made in order to ensure that the sound recognition technology needed functions together with the framework.

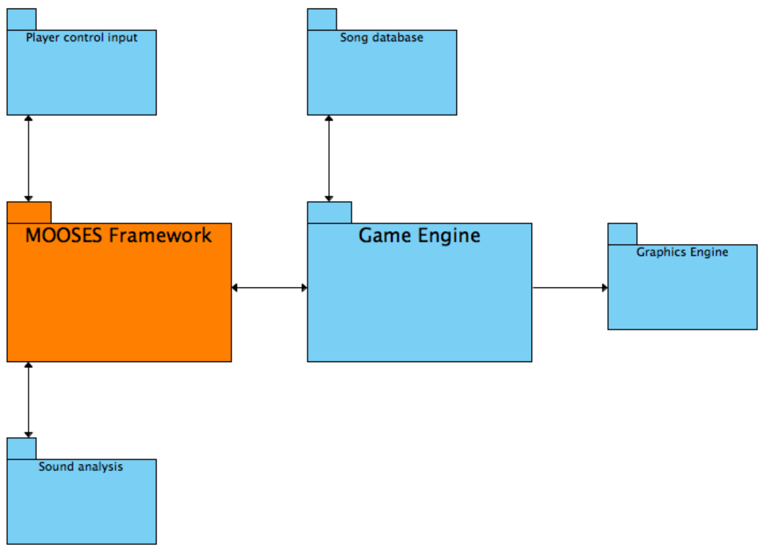


Figure 9.1: Software components for the Music Game

As shown in Figure 9.1, the software architecture for the Music Game has two extra components compared to the base MOOSES game architecture, the song database and the sound analysis module. They will be described in the paragraphs below:

Song database As we stated earlier, the viability of the Sound Game requires that the selection of songs is up to date at all times. Therefore, the game will need a constantly updated song bank that the game itself can fetch songs from.

This database can be centralized and serve a large number of game installations and venues. In fact, this distributed model is the preferred architecture as updates of the song base would then become a simple one-location operation.

Sound analysis As the game will rely on frequency analysis to grade the quality of the singer in each band, this module is needed. It has to be able to take in several different voice streams from various microphones and to analyze these in relation to a benchmark pitch and rhythm. These data would then need to be fed to the MOOSES framework as standard control data, ensuring compatibility with current and future framework versions.

9.6 Alternative Gameplay

One conceivable gameplay alternative in the Music Game is an implementation that uses classical music as its medium and in which all the players are playing in the same symphonic orchestra. This would not demand too big changes in the game engine (given that the number of different instruments is still kept somewhat small) and would lead to a game in which everybody needs to cooperate - or to fight against each other.

9.7 Related Games

Although the Music Game is in many ways similar to other games in the same genre, we believe that the special MOOSES setting allows for a totally different game experience, as the inclusion of larger groups and thus a larger audience/crowd creates an extremely immersive experience for the players. Games that explore similar concepts as the Music Game include Singstar (singing), Donkey Konga (drums) and Guitar Hero (guitar).

Chapter 10

The War Game

In this chapter we will describe a game inspired by the board game ”Risk”, with the important difference that this is played by a large number of players, all (initially) controlling a separate country, fighting each other in real time after simultaneously deciding upon the course of action each player wishes to take.

Table 10.1: the War Game infobox

Game Information	
Central concept	War between land areas
Number of players	10 to 25
Controls	Mobile phone, screen based
Play mode	Player vs. player

10.1 Concept Outline

In the board game Risk the players compete to meet a predetermined goal that could be something like “conquer Europe and Australia” or “eliminate the green player”. All the players then pursue their own goals by taking sequential turns, each attacking the opponents that he believes to be in the way of his agenda.

In this game we aim to create much of the same effect, the difference being that all players have the same goal - world domination - and that the fighting all takes place simultaneously after the players have chosen their desired actions. Thus - the gameplay are separated into distinct phases:

1. **Deployment**
   The players receive new forces and places those at their own will in the areas that they already control.

2. **Decision**
   All the players decide on their next course of action - singling out the enemy territories that they want to attack or their own areas they want to defend.

3. **War**
   All the actions that were decided upon in the previous round are acted out and all battles are fought.

All the selections and decisions are done on the player's private mobile screens, and are not shown to the other players until the decision period has ended.

10.2 Game Mechanics

We will now describe each of the game phases in more detail, explaining the different modes of play.

10.2.1 Game start

The initial setup for a game with 11 players is shown in Figure 10.1. The borders are dynamically drawn at each game start, ensuring that the number of land areas that are available to fight over are consistent with a multiple of the number of players in the game. Although it is not properly reflected in the illustration, it is also important that areas border on a roughly equal number of neighbouring areas, to ensure fairness in initial area allocation.

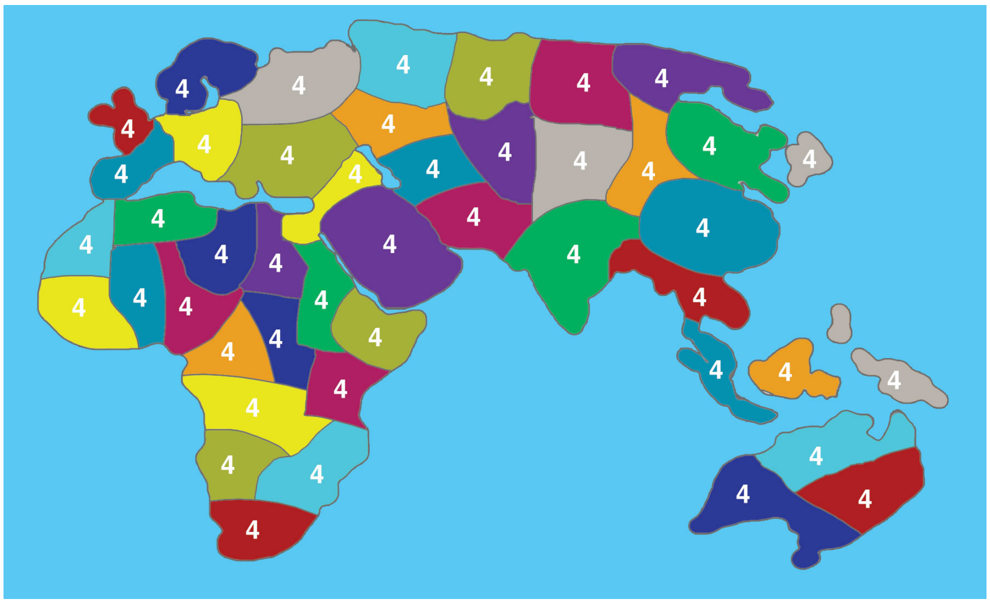


Figure 10.1: Initial level setup for eleven players

10.2.2 Deployment

In this phase, all the players receive more forces based upon the number of land areas they control as well as a random factor. More areas means more forces, but because of game balancing concerns this is not a linear relationship. A weaker player will therefore receive more forces per land area than a stronger player even though the stronger player receives the

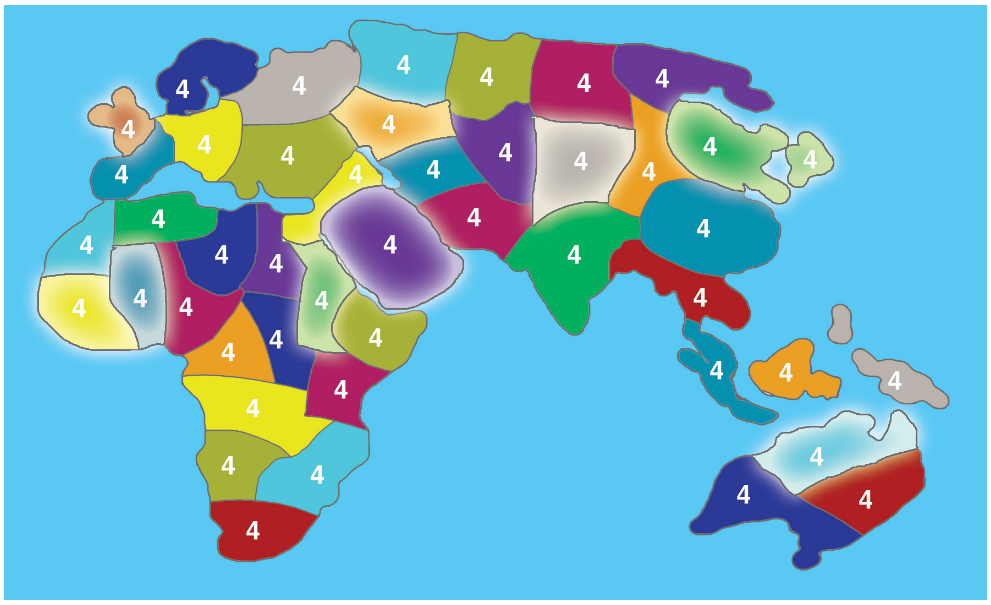


Figure 10.2: Force deployment with selected areas highlighted

most actual forces. This also means that a stronger player will have a harder time defending himself than a weak one.

The deployment is decided upon privately on each player's screen, but the total number of forces that everyone receive are shown publicly on the main screen to give everyone an idea of what the opponents will receive. Also - the actual land areas upon which players choose to place forces are shown, giving their enemies a hint of what they are up against. However, the actual number of forces placed on each area are not shown, giving room for deceit as a tactic (albeit an expensive one, as this would require the player to "sacrifice" forces to deceive the others).

Examples of screens from this phase is shown in Figures 10.2 and 10.3. As one can see from Figure 10.3 the players are given a time limit to make their choices. In addition, all players can report to the game that they are finished, and when all players have done this the remaining time will automatically reach zero and the game will enter the next phase.

10.2.3 Decision

In the decision phase, the players decide privately what they wish to do in the coming war round. This is done in the same way as in the deployment round, by using the mobile phone and its screen as a context sensitive controller. An example of a possible screen during this phase can be seen in Figure 10.4. The player can select one action per owned land area from the following set of actions:

full scale attack Attack a neighbouring area with all forces that reside in the active area. If the war is won, all except one of the forces from the initiating area will be automatically transferred to the conquered land. In this attack, the forces used gains some extra force to reward the player for going "all in".

limited attack The player chooses the number of forces he wishes to attack a neighbouring

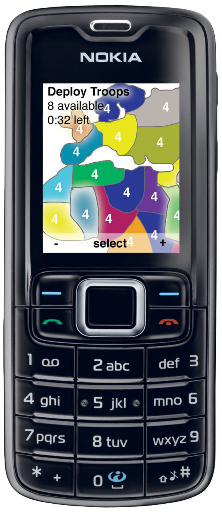


Figure 10.3: Force deployment control scheme

area with. This attack is not as strong as the full attack, but the risk is also smaller, as a defeat is not as dramatic.

move troops Enables the player to move any number of forces from one area to another within a range that decreases with the number of troops that are to be moved. For example, the player can send a small number of troops to any land area on the map, but only move a really large army to the area next to the originating land.

defend If the player chooses to do nothing with an area during a round, the defending forces gains some extra force in the possible defence of an area. This extra force comes as a number of extra forces if the area is attacked. These forces are kept if they survive the war. This is the default action that is selected if the player does not make a choice for a land area in any round.

As in the deployment phase, the players have a time limit as well as the ability to report that they have made their selection. When all players have made their selections, the game passes over into the War Phase.

10.2.4 War

The last and deciding phase in a game round is the War Phase. In this phase, all the actions that was decided upon in the previous phase are executed, and their results are shown on the big screen. While this phase happens simultaneously for all areas in the game world, it is shown sequentially to the players in order to maintain the possibility for the players to actually follow the action as it happens. Thus, the wars happen one after another in an ordered random fashion around the game map.

Deciding results

Generally, wars are decided in the same way as in most board games based upon war between areas - by dice. For every army unit that is fighting in an area one die is “thrown”. The

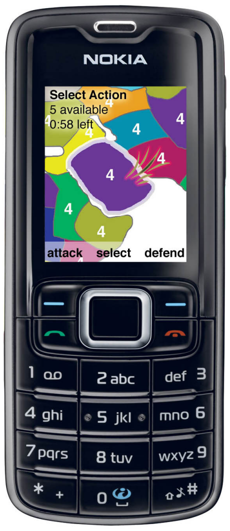


Figure 10.4: Action selection

dice is then sorted according to their value and compared one to one from highest to lowest. Each losing die then represents a losing army unit for the player who “threw” it, and this unit is removed from the game. This process is repeated until only one player has units remaining on the land area, and this player then keeps this area.

Example[1]**:** The red player attacks the green with a full attack of ten army units, the green has a defending force of eight (all numbers in this game description are arbitrary and play-testing is needed to decide on real values). The players throw ten and eight dice respectively, and the dice are sorted according to their values (Figure 10.5). The highest value for each pair wins - if there is a draw the result is decided from a set of rules that can be found in Table 10.2.

Table 10.2: Dice draw arbitration

Defender choice	**Attacker choice**	**Result**
Anything	Limited Attack	Defender wins
Defend	Full Attack	Defender wins
Anything else than Defend	Full Attack	Attacker wins

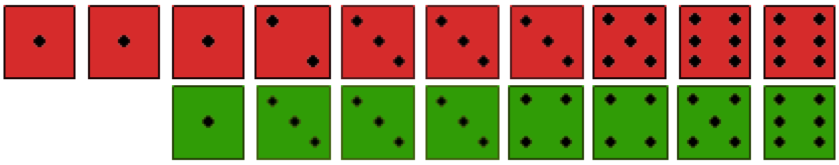


Figure 10.5: Dice scenario phase one

After phase one of the dice fight between the two players, red win with two of his dice,

[1]created by throwing actual dice

green with six, which leaves green with six units left and red with four. As there is no way to end a war before one player is eliminated from the area, the dice are “thrown” again, and the results can be seen in Figure 10.6

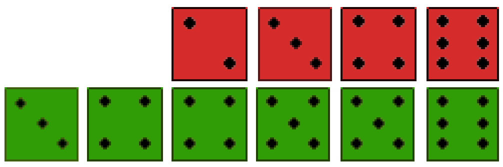


Figure 10.6: Dice scenario phase two

As we can clearly see, green is very lucky in this round and wins with all his dice, hence causing the red player to loose this war and his forces from the attacking area (leaving one sole army unit guarding the land).

Special considerations

While the main rules for war can be used in most cases, there are some special situations that needs to be have some extra rules when deciding upon the winner of a conflict. Underneath, we will explain these situations and the rules that apply.

Chase attack If an area is both the origin of an attack and the victim of another attack (from a different area than the one that is attacked) at the same time, the forces that are attacking the neighbour are not counted as defensive forces in their area of origin.

Border war If forces from two areas meet at the border (they attack each other), the fight takes place at the border, and any surviving forces from the winning side then attacks any remaining forces at their origin in a new fight immediately afterwards.

Multiple fronts If an area is attacked from two sides at once, the attackers are counted as one army first (with their dice put together). If the attackers win, they then fight a new war between them with the surviving forces. The rules from Table 10.2 applies as normally for their respective dice.

10.3 Alternative Rule

To enable shorter play time for the War Game, it is possible to integrate the Deployment Phase and the Decision Phase. By doing this, the deployment of new forces becomes a choice along with the others in the Decision Phase, and a player who chooses to put new forces on the playing field would therefore not be able to wage war on anybody in that same round. This change of rules would make the game shorter by putting fewer forces on the field and by increasing the chance of large armies attacking smaller ones. It is however possible that some balancing concerns could also arise, but this would have to be tested in an actual implementation as play balancing is a very subtle art that it is impossible to nail completely at design time.

10.4 Controls

This game will be controlled by the screen control scheme described in Section 7.2.1 (page 50). We believe that this is the method best suited to controlling this game, as the screen is extensively used to give feedback to the user throughout the different game phases. This game has high demands on the controller side software, as this needs to be able to relay lots of information back to the players all of the time. Figure 10.3 and 10.4 shows possible screens from the mobile phone during play.

10.5 Software Architecture

Similarly to the Football game, this game also follows the base software architecture in Section 6.5 (page 43) closely.

From a technical standpoint, the War Game can be made to be as advanced or as simple as one wants. There is no need for advanced graphics, but the game could clearly be made more interesting by adding realistic graphics.

As this game has the most advanced gameplay mechanics of the games that we describe in this report it is important that the graphics and the control system are able to successfully immerse the player and to keep him oriented in the game universe.

There is only one module that needs any kind of advanced intelligence programming, and that is the map initialization code. This module needs to be able to separate the map into an arbitrary number of separate areas, each with a roughly similar number of bordering areas, and to divide these areas between the players in a way that gives every player a similar starting point and equal chances for victory.

Chapter 11

The Football Game

Here, we will describe a sports game in which every player controls a player on one of two opposing teams. As well as being in control of their own on screen avatar, the players also participate in an all-team effort whenever one of their team-mates attempts to score a goal.

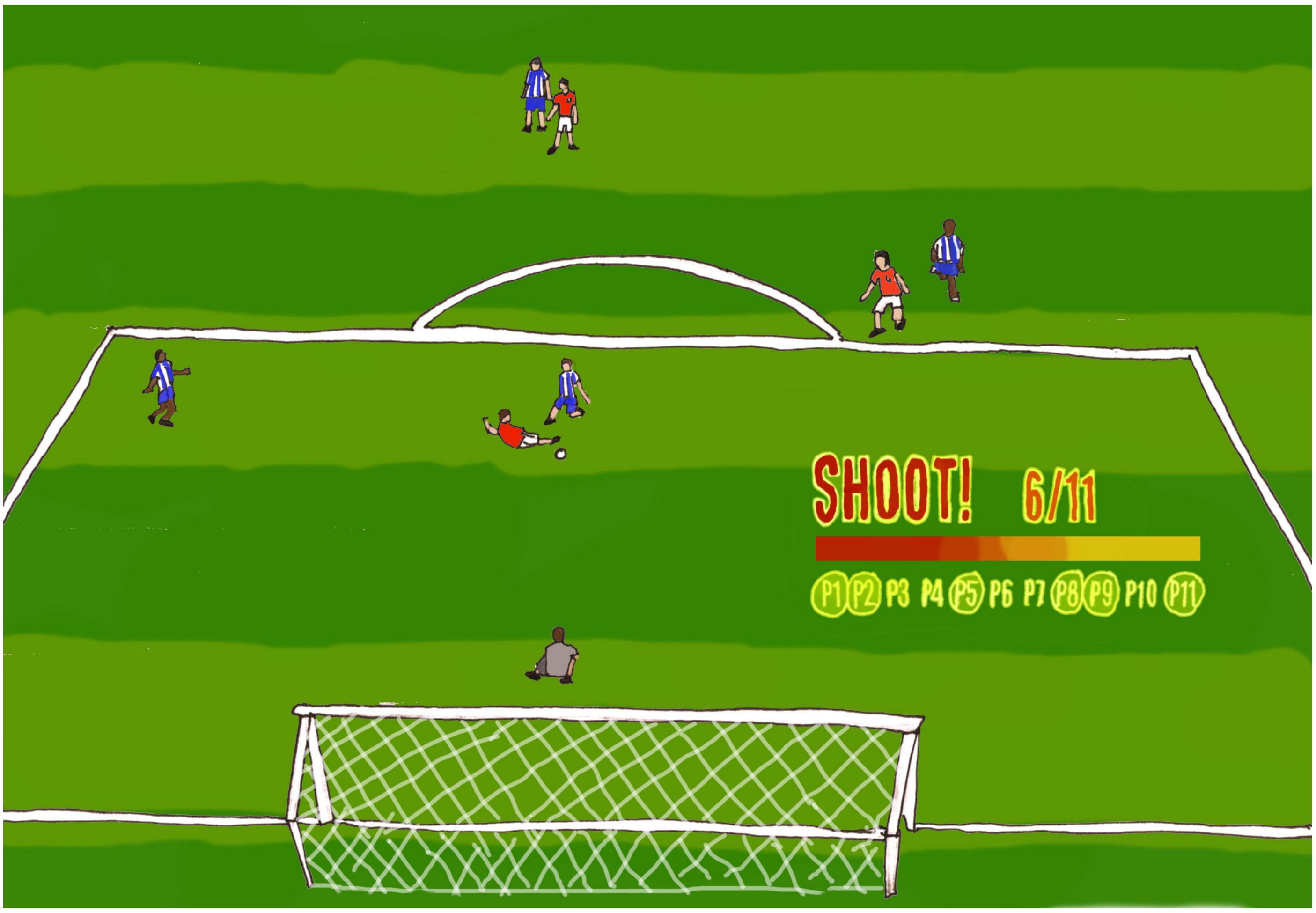
Table 11.1: the Football Game infobox

Game Information	
Central concept	Football match between two teams
Number of players	11 or 22
Controls	Traditional game controllers or custom controls
Play mode	Team vs. Team, Co-operation

11.1 Concept Outline

While virtually everybody who have spent a certain amount of time playing video games have played football games, this has mostly been done alone or against a friend. What we will try to do with this multiplayer football game is to include all 22 players in a football game in which everybody has to take part to succeed as a team.

When played in its most basic way, this game will play exactly like a normal football video game, with 22 players - 11 on each team - each having his or her own designated position and role. The defence (including the keeper) tries to keep the opponent from scoring a goal and the midfielders and attackers tries to score a goal on the opposing team. A mock-up screen shot from the game can be seen in Figure 11.1



Concept art for the Football Game

part of field shown at the moment of shooting

However, when dividing the players like this, there is a danger that some players will get to interact with the ball less than others. In an attempt to stop this from damaging the gaming experience for the players we will implement some measures to ensure that cooperation within the team pays off. The first of these measures is a system that gradually (and visually) punishes players who play in an egoistic manner as opposed to those who cooperate with their other team members. This will be done by making each player have a fitness marker on his or her avatar that indicates how fit the player is at the moment. This fitness will be decided by several factors, the most important being the time since the last time the player touched the ball, and the fitness of the other players that the player has passed the ball to earlier in the game. In this way, we ensure that it is in a players interest to always pass the ball to the player that has participated least until the current time, creating an incentive for teamwork.

Another important measure that will ensure that all players are included at all times is the inclusion of a shot timing feature that enables all players to participate when a shot is fired at one of the goals. The way this is done is by displaying a shot bar (as shown in Figure 11.2) on screen just before a shot is made, after which all players who wish to participate in the shot can try to match a point on this bar to help make the shot harder and more precise. Or, if the player is at the team whose goal are being shot at, he can (in the same way) help the keeper become more agile, faster and more likely to catch a shot.

11.2 Game Mechanics

As we assume that the reader is already familiar with the game of football (soccer), we will not go into detail about the rules of the game itself, we will instead explain how they will be executed in this game in the situations where just implementing the rules of football is not enough to make the computer game work.

Thus, where not defined otherwise, the rules of normal professional soccer can safely be assumed to apply within this game as well.

11.2.1 The Shot Bar

As mentioned in the preceding section, an important factor in making this game include all players at all times are the shot bar. This bar (for the rest of this section we will be referring to Figure 11.2) is displayed whenever a shot is made, and its functionality is as follows:

- Whenever a player presses the “shoot” button, the shot bar are displayed prominently on screen in the state shown in *t1*
- After a very short period of time, another bar appears on top of the shot bar, placed on the left side of it, as shown in *t2*
- This bar then starts moving very fast towards the right (see *t3*, approaching the green “sweet spot” placed on the right half of the shot bar
- All players on both teams have the opportunity to attempt to press their own “shoot” button when the vertical bar is as close to the sweet spot as possible, adding to their own team´s shot bar score for this particular shot. *t4* shows the optimal point for the players to press the “shoot” button, as the slider bar is in the middle of the sweet spot

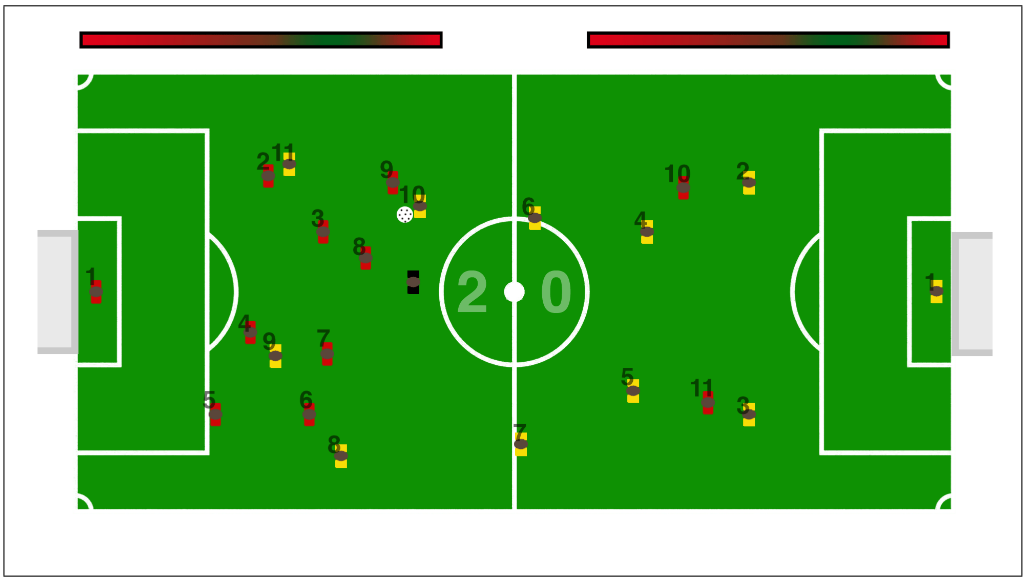


Figure 11.1: Mock-up of screen shot

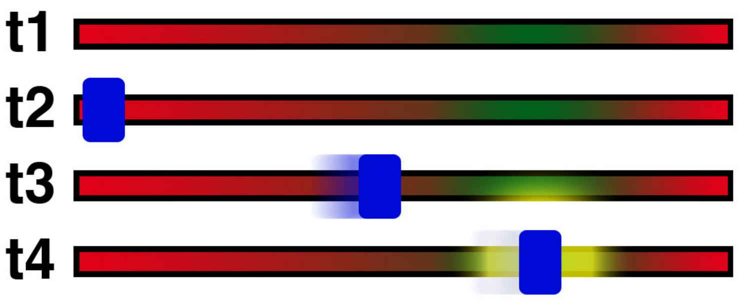


Figure 11.2: The shot bar

- After the vertical bar has moved all the way over to the right side of the shot bar, the combined shot bar score of both teams are calculated and this is then used to decide whether the shot was a success or not

An important point about the shot bar is the distribution of points related to the degree of success that the users have had in hitting the center of the bar. To encourage participation, we suggest that the points are awarded as described in Table 11.2 (the scale is of course continuous, all intermediate scores are also used.). Also, the numbers used are arbitrary, it is their relative inner relationship that is important here, and only by play testing can actual values for them be found.

Table 11.2: Score table for Shot bar

Situation	**Score**
player does not try to press the shoot button at all	- 10
player tries, but fails miserably	- 1
player tries, but misses slightly	0
player hits the green area, but is not dead on the center	2
player hits the sweet spot perfectly	5

This point distribution helps ensure that the players actually participate in the game at all times, as their failure to do so would damage their team greatly. To give feedback to the players who manage to hit the right point at the shot bar (as well as those who try and fail, but then to a lesser degree).

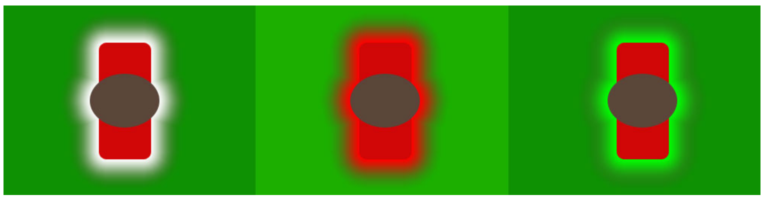


Figure 11.3: Visual feedback on shot bar success

Players who do this will also have their fitness increased and there will be a graphical representation of their degree of success as shown in figure 11.3. This graphic shows neutral, bad and good timing, respectively.

11.2.2 The Fitness System

As mentioned earlier, this game will also utilize a fitness system to help motivate the players for team cooperation. This system will award players fitness points based upon their level of cooperation with other players on their team, as well as especially awarding cooperation with players that have not interacted with the ball in a while. A player's fitness status is shown at all times as an indicator on the players avatar. An example of how this could be implemented graphically is shown in Figure 11.4, where the player has respectively low, medium and maximum fitness levels.

The fitness level of a player affects most of his or her attributes, such as speed, shot accuracy, shot power and technical proficiency. Thus, a team wishing to play at its optimal needs to keep a close eye on this factor. This in turn creates the desired effect, namely awarding cooperation amongst the team mates.

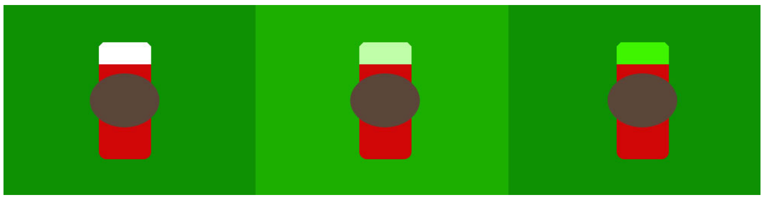


Figure 11.4: Visual feedback on fitness

The player actions that affect a player's fitness is shown in Table 11.3

Table 11.3: Factors affecting a player's fitness

Situation	**Effect**
Player passes to a player with high fitness	Increase
Player passes to a player with low fitness	Decrease
Player misses the timing on the shot bar	Slight decrease
Player hits the timing on the shot bar	Increase

In addition to the factors mentioned in Table 11.3, a player's fitness will automatically increase over time, more quickly so if he or she stays within the area of the field that the allotted role dictates. Similarly, moving far away from this area would automatically decrease fitness and prompt the display of an indicator showing the direction of the correct area. This does not mean that a player could not go for a solo run towards the goal if he or she wants to, only that this is not behaviour that is actively encouraged within the game.

11.3 Controls

While it is possible to envision controlling this game via a mobile phone, we believe that for a highly interactive game like this, the use of more traditional game controllers would greatly increase the level of control that the players are able to exercise over their avatars. This is mostly due to the lack of directional controls on most mobile phones, as well as the inability to use both hands to control the action. Also, the Shot Bar concept requires a high level of timing between on screen action and controller input, a level that we believe it would be hard to achieve with mobile phone controllers. Therefore, we advise that custom seat-mounted controls or some kind of game controllers situated at the users seats would be the superior option for the Football Game.

11.4 Technical Aspects

As with most of the other games outlined in this report, the Football game is not necessarily especially graphics intensive. We envision a very functional graphical style with cartoonish styling, as this enables the players to more easily keep an overview of the entire play field and the action thereon.

One decision that would have to be made at implementation is whether one should include an AI component that would enable a team of human players to meet a computer controlled opponent if the number of players is not big enough to create two full teams. This would complicate the development of the game, but also enhance the value and versatility of the game.

11.4.1 Software Architecture

This game closely follows the base software architecture outlined in Section 6.5 (page 43). The only additional component is the addition of an opponent artificial intelligence module if this is to be implemented.

We will not therefore not include a more detailed view of the software architecture for this game, as we believe the previously described architecture to be sufficient.

Chapter 12

The Survival Game

In this chapter we will describe a game based upon the movement of a school of animals as they are chasing for food or being chased by predators. It is a smaller game than the rest of the games described in this report, and as such is suited for short, intense gaming sessions. Therefore, it is especially well suited for situations like pre-movie playing at a cinema or for use in other short-term situations.

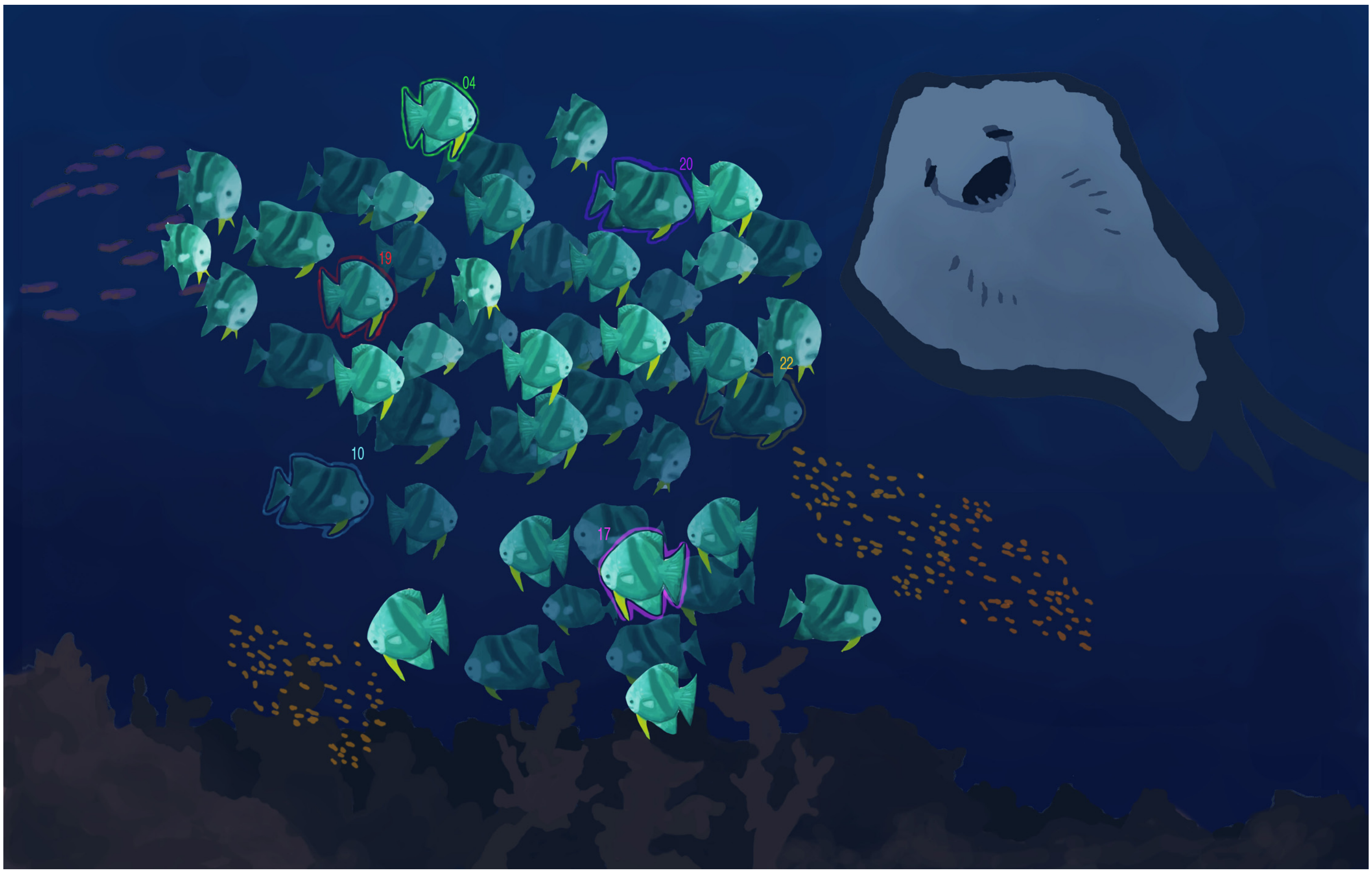
Also, the game scales well downwards, enabling gameplay for a small number of players as well as larger audiences.

Table 12.1: the Survival Game infobox

Game Information	
Central concept	Survival of enemy attack through flock behavour
Number of players	up to 50
Controls	Mobile phone, race car style
Play mode	Players vs. common enemy

12.1 Concept Outline

If one has ever watched animal programs on TV, one has probably seen stunning shootage of large schools of animals moving as one individual (an example can be seen in 12.1). This game tries to simulate that kind of uniform movement, with each player acting as a single entity in a large flock. First, the players try to catch food that appears in the level. By eating this food the avatars gains strength. After a while the enemy, for example a large predator, enters and tries to catch individual members of the flock. As a reaction to this, the players all have to escape to survive. The ones that succeed will be the ones that move in accordance with the rest of the flock in such a way that they are at all times concealed by the surrounding individuals.



Concept art for the Survival Game

There are only a few human players left, and the enemy has just made a sudden turn

We believe that the fact that every player is in full control of his own avatar, combined with the absolutely necessary cooperation with the rest of the players, means that the game will offer a truly different gameplay that could only be achieved in this setting.

12.2 Goal

The goal of any player is to place at the top of the pack. The ranking is very easily decided from the playing time of each player. The player that survives the longest time gets the highest score.

12.3 Game Mechanics

As the game starts, all players are lined up at one side of the screen. A few[1] items of food is placed at another side of the screen, and the players have to race towards the food to gain energy and grow. This repeats for a little while. At this point in the game, all players are equally strong and moves at the same speed. However, the players that has eaten the most food will be larger and heavier and thus have a harder time turning. On the other side, they will also, based on their higher mass, be able to push lighter players out of their way as they try to catch food. with the enemy placed somewhere at a safe distance from them. At a semi-random point in time, a short while after the game has started, the enemy appears and starts moving towards the players, trying to catch the ones that are sticking themselves out from the crowd. This happens dynamically, so that the enemy (in a natural way) may change his target (or target group) as often as it would be natural for a predator.

As the avatars move, there will be a constant push towards the center of the group, as all players search to be at the maximal distance from the fringes where the enemy is most likely to attack. Hence, the avatars will need to act like bodies with physical attributes that decides their ability to push themselves towards the middle. For example, the weight of the avatar (based upon the amount of food consumed) will decide how effectively he is able to push other avatars out of his way on his way towards the center.

When an player avatar is caught by an enemy, it gets hurt or killed. The amount of damage an avatar takes from a hit is decided from the severity of the attack as well as from the relative strength of the avatar (based upon the amount of food eaten) as well as the angle and speed of attack. If the avatar is killed, it is replaced by a computer controlled replica. This replacement of dead players with computer controlled avatars ensures that the remaining players always has a set of bodies to hide between, even if very few players remain.

12.3.1 Balancing factors

To ensure that the game is balanced and interesting at all times, and that it does not go on forever, there are a few rules and limitations to the gameplay. These following rules are used to ensure this:

- The players slowly use up their gained energy until they are back at base level. At this level, they can only sustain one major or a couple of minor blows from an enemy

[1]Fewer than the number of players

Figure 12.1: Group movement similar to the one the game tries to simulate

- As the players use their energy, they also become lighter and therefore lose their advantage compared to the players that ate less during the first phase. This ensures that the game outcome is not decided in the first phase.
- Swimming faster than the default speed uses more energy, thus making running away have a cost to the player.
- The enemy gains speed and agility over time. Also, the avatars that replace dead players grow over time, gaining energy. Thus, a very skilled player will not be able to play on for ever after all the other players have been eliminated.

12.4 Controls

To ensure that this game is controllable and that the players always know which avatar is their own, the entire game takes place in slow motion. The exact speeds that shoud be used are impossible to decide upon without playtesting the game, but we believe that running a game like this in the realtime or near realtime speeds of the simulated motions would render the game unplayable.

Apart from this, the game is controlled by a racecar control set as described in the prestudy. We believe that this is the control scheme that most closely resembles the motions that we simulate, as it takes into account the fact that the controlled body has speed, mass and inertia, and that all motions are adjustments to the movement the body had in the previous moment.

Thus, a mobile phone used for controlling this game would be configured as shown in Figure 12.2, with four directional buttons and a minimal set of status information displayed

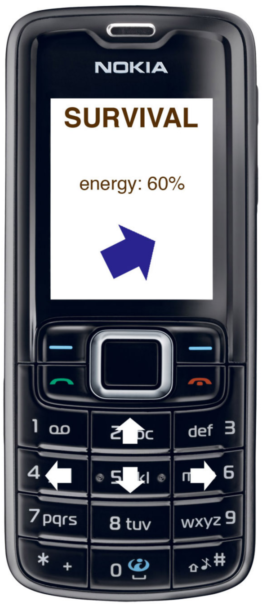


Figure 12.2: Cell phone controlling the survival game

on the device screen. This is a very basic and simple control scheme that would be playable on virtually all mobile phones, ensuring maximum compatibility with player's equipment.

12.5 Technical Aspects

In technical terms, this game is rather straightforward. The main challenge will be to ensure that control input from the players are handled in real time, ensuring fair and balanced possibilities for all players.

12.5.1 Software Architecture

As the main sections of the MOOSES framework are described in the prestudy, the parts that are mentioned there will not be described in detail here. However, we will explain the most important system modules shown in Figure 12.3

Game Engine

This module handles all communications with the other modules as well as the game logic itself, and all communication with the framework.

User control client

This client runs on the mobile phone of the player, keeping the private display updated and ensures that control input are relayed correctly to the framework.

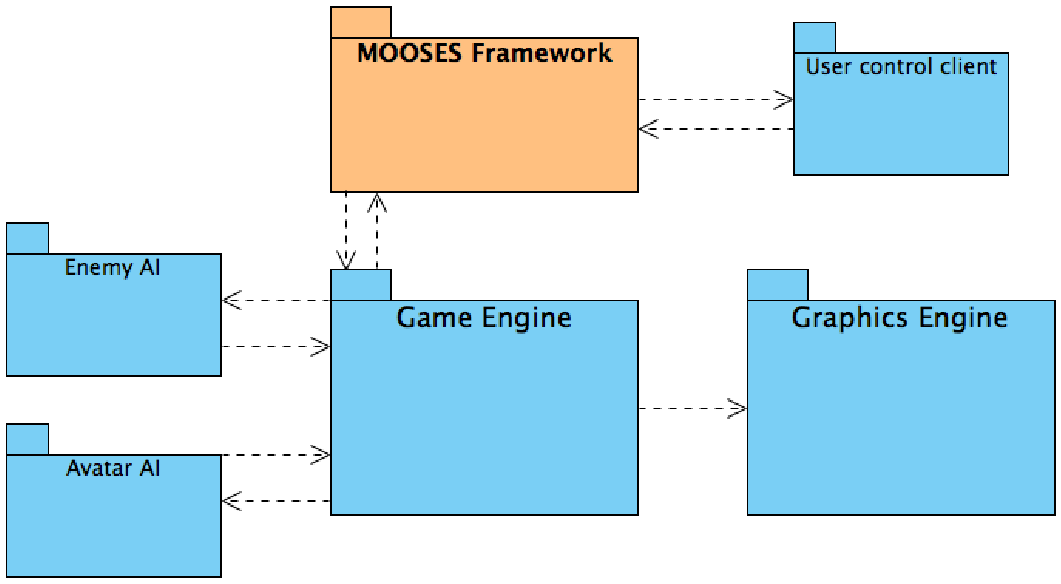


Figure 12.3: Top level system design for the Survival Game

Enemy AI

For this game to work, it is extremely important that the enemy artificial intelligense acts in a predictable way and as expected by the players. For instance, the selection of targets must be handled dynamically and realistically, making sure that players and avatars are targeted in the way that encourages the kind of motion that the game requires.

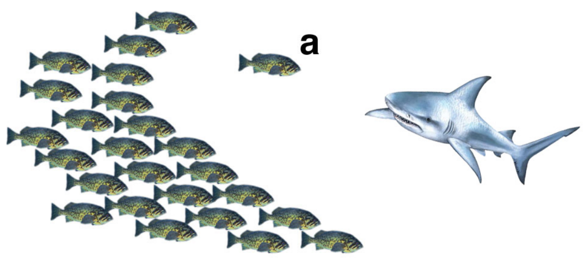
When faced with a situation like the one pictured in Figure 12.4, the enemy (represented by the Shark) will attempt to attack the lone fish marked with an **a** in Figure 12.4(a). In Figure 12.4(b) the enemy will attempt to go in the direction of the group of fish marked **b**. Within this group, the enemy will target a random fish controlled by a human player if there is any, if not it will go for one of the computer controlled fish.

Avatar AI

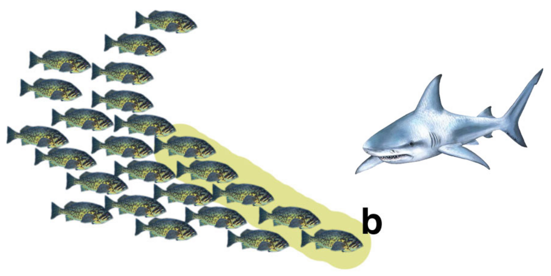
After a player has been caught by the enemy, it is immediately replaced by a computer controlled replica. This replica will attempt to act like a human player, trying to stay in the middle of the pack just like any other player. Thus, the task of the avatar AI is to try to replicate human player actions as closely as possible and to enable the remaining players to keep on playing as if the entire field was still controlled by other players.

12.6 A Shorter Alternative

If one wants to implement a simpler version of the Survival Game in which the playing sessions takes somewhat shorter time, it is possible to eliminate the feeding phase of the game and go straight into the mode in which the players escape the enemies. This would remove a layer of complexity from the game and makes the game mechanics easier to comprehend, but in our opinion it also removes some of the realism from the game as well as simplifies it.



(a) single target



(b) multiple targets

Figure 12.4: Enemy AI priorities

As this alternative is simply a subset of the full game, however, its implementation would not have any large impact on the development time, and it is therefore probable that its inclusion in a finished version of the game would add to the diversity of the game and therefore be worthwhile.

Chapter 13

The Quiz Game

Although it might seem like a rather obvious application for the MOOSES framework, we believe that a quiz game would have great potential and that it is therefore worth exploring in this report. The game that we will describe in this this chapter works a bit differently than the rest of the games described in the rest of this report in that it does not necessarily require the full screen for play, and in that it can be used as an augmentation of the immersion in the other parts of the cinema experience, or as a supplementary source of entertainment during commercials.

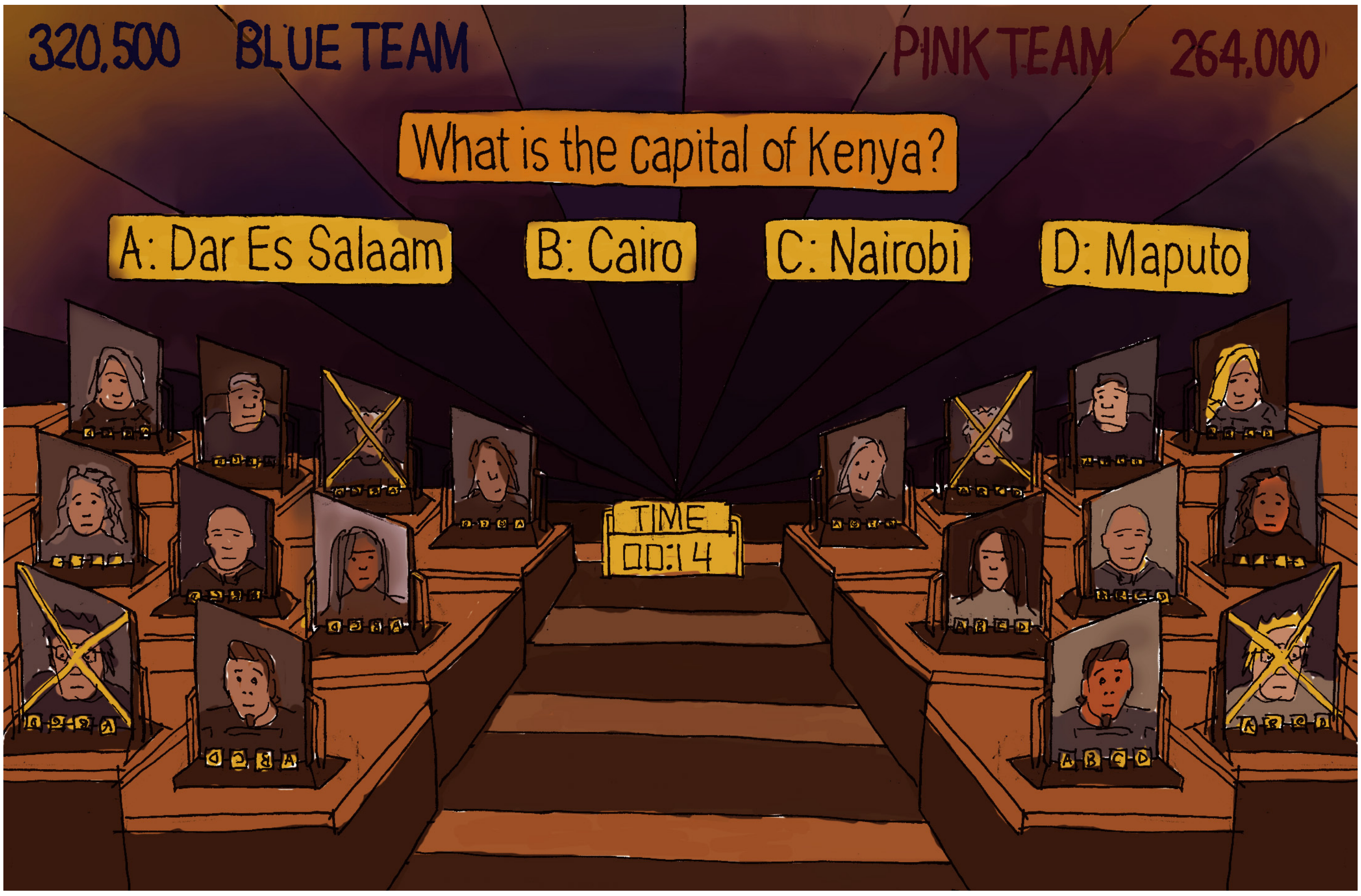
Table 13.1: the Quiz Game infobox

Game Information	
Central concept	Game show style quiz
Number of players	Virtually unlimited
Controls	Mobile phone, direct mapping
Play mode	player vs. player

13.1 Concept Outline

This game could be played in two modes. Either it could be played as a standalone game within the confines of the currently implemented framework, or it could be played in the way mentioned above, with the game graphics overlain over other content shown on a cinema screen. However this distinction is not especially important for the basics of the game itself, so we will not pay very much attention to this where it is not explicitly needed.

The Quiz Game is based around the same mechanics as most other quiz related games, with a question, some alternative answers and points awarded for answering correctly as fast as possible.



Concept art for the Quiz Game

Two team configuration

13.2 Game Mechanics

As anyone that has played a quiz game (most notable lately is the Buzz series mentioned in Section 5.3.3 on page 25) knows, there is a multitude of ways to ask players questions and reward correct answers. The questions can be text-based, musical or they can be based upon videos or pictures. The players can be rewarded for answering first, answering at the right time or answering the same as the others, and players can have interaction between them by for instance being able to steal points from each other.

13.2.1 Base mechanics

The base mechanics of this game is very simple, and can be summarized as consisting of three steps:

1. A question is posed to the participants.
2. The participants select and submit their answer.
3. A score is awarded.

However, as mentioned above there is a large amount of different ways to accomplish each step in the game:

Step 1 variations

Question types:

- Text question ("Which of these is the capital of Burkina Faso?")
- Sound question ("Which band plays this song?")
- Video question ("From what film is this movie clip?")
- Question pair with vote ("1: What is your favourite color?, 2: What is the most common favorite color in this audience?")(see 13.2.2)
- Question related to other on-screen content (see 13.3)

Step 2 variations

Answer modes:

- Answer correctly as quickly as possible
- Answer correctly within a given time
- The team all answers, final answer decided by vote (see 13.2.2)
- Cast a vote, then select supposed majority vote (see 13.2.2)

Step 3 variations

Scoring options:

- Players with correct answers all get a similar amount of points
- Players gets points based on swiftness of correct answers
- Teams with correct answer gets similar points
- Teams with correct answer gets points based on level of agreement and or swiftness of correct answers
- Points are awarded or retracted from players based on their level of conformity (or lack thereof) in a question pair with voting.

13.2.2 MOOSES advantages

We believe that an expansion of this concept to such a large setting as the MOOSES projects allows us to take the Quiz Game concept a step further, making room for team play, other types of questions to answer and an even more social experience than the player gets by playing comparable games at home.

Below, we list some possible game options that a quiz based game with fifty or more participants would allow:

Team play With the inclusion of many players, one also gets the opportunity to have them compete in teams. This would make the game somewhat equal to the way quizzes in bars and other social settings are performed. One possible way of implementing this is by majority vote, in which all the players on a team vote independently on the answer they believe is correct, and where the collective answer of a team are decided by a popular vote. The teams can be defined in advance by a team leader or they could be created randomly or based upon selection criteria.

Majority vote Question pairs without a predefined answer, like for example ”What is your favourite color” and ”What is the favorite color of the majority of this audience?” This is of course also possible with fewer players, but it only starts to make sense when there is quite a few answers to make the statistics from.

13.3 The Quiz Game as an Overlay

If used as an overlay game while other content is being shown, this game would present itself as a semi-transparent field of text placed on top of the content that is shown on the cinema screen. An example of how this could look is can be seen in Figure 13.1.

In this mode, the Quiz Game could be used by a cinema to help ensure that the audience pays attention to what happens on screen. This could be a great way for a cinema and MOOSES operator to enhance the audience’s willingness to view the commercials. For example, a quiz could be arranged alongside the commercials (for those who wished to participate of course). The questions could be related to the commercials shown and the winner could be awarded something of value to him, such as free tickets or other real prizes. We believe this to be a concept of great potential, but the full exploration of the commercial opportunities is not within the scope of this report.

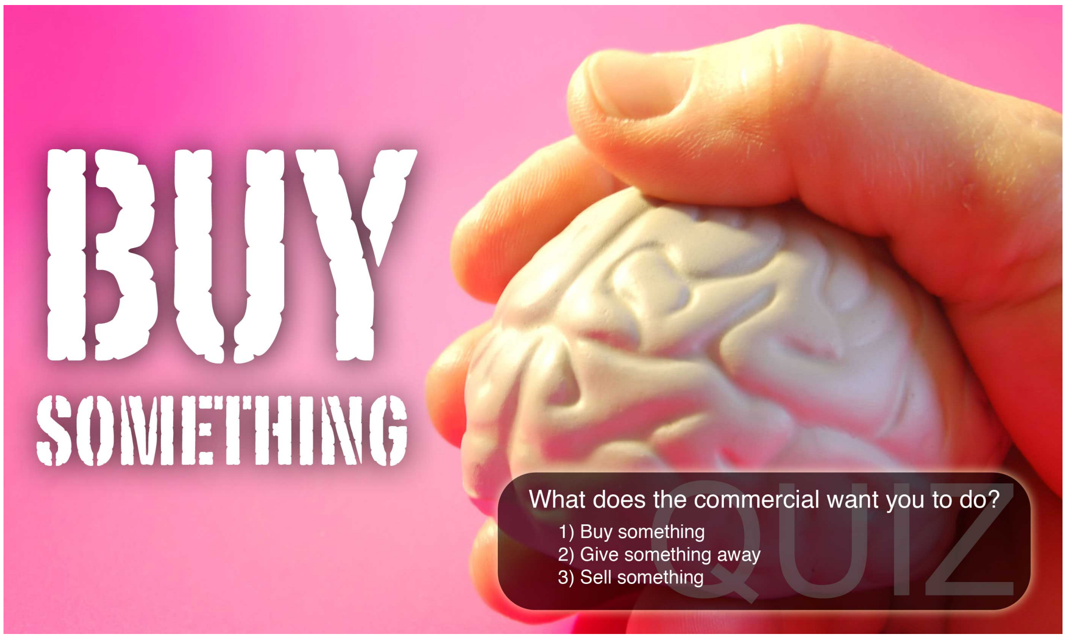


Figure 13.1: Quiz question overlain on cinema screen content

13.4 Controls

As this came is based upon the quick selection of an alternative from a set of alternatives, the most suited mode of control is the Numerical Control described in Section 7.2.1 (page 48). This control scheme gives an intuitive and quick way of selecting the correct alternative and is very easy for the player to understand. In a situation where a quiz game is used along with other entertainment in a cinema setting, custom seat-mounted controls would also be a viable option.

13.5 Technical Aspects

If played as a "normal" MOOSES game, the Quiz Game is probably the least technically demanding of all the concepts we describe. It does not require any kind of AI, the graphics are not necessarily demanding on the hardware, and the controls can easily be implemented using a small Java client or even the built-in menu augmentation function of most phones (see Section 7.2.1).

On the other hand, if this game is to be played as a complement to the other content shown on the cinema screen, some new technical issues will have to be tackled. As of now, there is no mechanism in place to overlay graphics from the MOOSES framework upon an existing picture. However, this should not be too difficult to achieve, using a video compositing system connected to both the video source and the MOOSES system and implementing an alpha channel in the game graphics, enabling a transparent display of questions as shown in Figure 13.1.

Updating questions

To make a game like this interesting and ensuring that it has a long life, it is absolutely necessary that the question base that is used are updated regularly. This could easily be accomplished by using an online database of questions and by letting the game itself update its internal question database (in case the connection to the question server is lost) regularly.

If the game is used as an enhancement to other content, an easy update mechanism is even more important to implement, as this would mean that questions related to other content would have to be added whenever content is added or shuffled. Ideally, there should be some kind of synchronization between the two systems, but as this demands change in the cinema systems itself this is a rather complex system to develop. Without this synchronization, a system with timed questions would probably be the most suited, although this leaves lots of room for error.

13.5.1 Software Architecture

Figure 13.2 shows the top level software architecture for the Quiz Game. In the following section we will explain the main components of the system and their respective tasks.

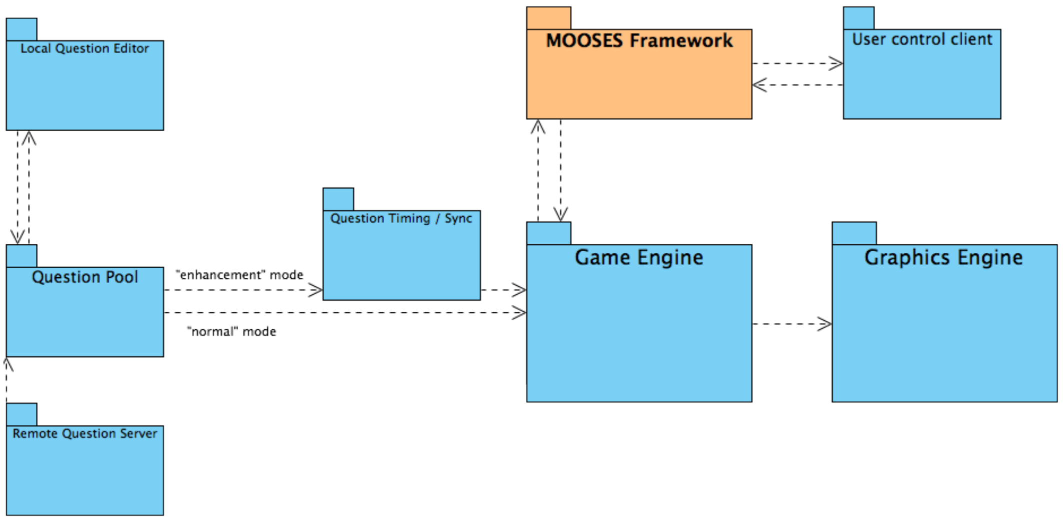


Figure 13.2: Top level system design for the Quiz Game

game engine This component controls all the others and manages the communication between them. It also keeps track of game progress and contains the game-specific run time code.

user control client The user control client runs at the player's mobile phones if the game is controlled by this method. Otherwise, it is not required, as custom quiz controls would presumably be part of the MOOSES installation in venues with such controls installed.

question pool This is the system component that holds the questions that will be presented to the players and that keeps track of which questions has been used and when. It is also responsible for polling the *remote question server* to check for new and updated questions. The question pool is also responsible for keeping track of question categories to ensure that questions are chosen in an appropriate manner.

local question editor Although questions are chosen from a central pool, the local operator should be able to customize the categories and difficulty of the questions asked to ensure that the game experience is optimal for many different audiences. This component enables an operator to customize game sessions as well as to add localized questions to the local *question pool.*

remote question server This component is basically a database with some logic that is able to hold a large number of questions and to maintain these in ordered by different categories (difficulty, category, freshness[1])

question timing / sync If the Quiz Game is run as an ”enhancement” to other content as described above, this system component ensures that the questions are asked at the right time. If no synchronization service is implemented, it asks pulls specific questions from the pool at specific times. Otherwise, it communicates with external systems to ensure that the correct questions are presented to the user at the correct times, as dictated by the external system.

13.6 Further Possibilites

In addition to the gameplay mechanics outlined in this chapter, it would be interesting to implement this game with an alternative game mechanic centered around a voting process. In this version of the game, everything would work as before, except that there would be no right or wrong answers, and that the questions would take the form of opinion polls about various subjects. The players could then be rated after a round regarding to for instance how well their choices corresponded with those of the rest of the audience.

This concept could also possibly be extended to be used during the screening of a film, in a process where questions related to the action could be asked to the audience during the movie itself. However, this would probably require custom controllers so as to not interfere with the movie-going experience of those not wanting to take part.

[1]for questions related to current events

Part IV

Evaluation, Further Work and Conclusion

Chapter 14

Evaluation

In this chapter we will look at the MOOSES game concept project to see if we have succeeded in finding satisfying answers to our research questions and if we have been able to create game concepts that fit well within the confines of the MOOSES project. We will also look back at our research methods to see if they suited the project.

14.1 Research Questions

In this section the reader will find a repetition of our research questions as well as very short answers to them based on the findings described in this report.

14.1.1 RQ1

As stated in Chapter 3:

> *What are the major design considerations that has to be taken into account in order to successfully create a multiplayer one screen game?*

By thoroughly looking into many aspects of multiplayer gaming we believe that we have been able to identify the main pitfalls and considerations related to making MOOSES games. We have found out various ways of controlling such games and the limitations on graphics, gameplay and other factors that are imposed on MOOSES games due to their nature.

14.1.2 RQ2

As stated in Chapter 3:

> *Which kinds of gameplay, manifested as game concepts, can successfully be implemented as massively multiplayer one screen games?*

After considering many different concepts and killing off a significant number of them, we believe that we have been able to find a set of games that would function very well in a MOOSES setting. Although it is possible that some aspects of the games would have to be adjusted as a consequence of issues found during play testing, we feel sure that the base premises for the games are functional, well balanced and most importantly fun, rewarding and challenging to the player. Of course, as a virtually unlimited number of games could be created that would fit well into the MOOSES setting, our game concepts are just the ones that we have the most belief in and that we believe are possible to implement in a short time with the current MOOSES framework.

14.2 Research Method

As stated in the Research chapter, we have used an iterative approach to our game concept designs. After researching the material available on multiplayer games and summarizing our knowledge we spent lots of time creating, recreating and tweaking the concepts until we had concepts that we believed to work in a MOOSES setting. This iterative method is especially important when dealing with action games like the Football Game and the Survival Game.

As we have not been able to implement any of our game concepts (that would be far out of scope for the project as well as a way too monumental task given the time and resources available), we believe that our choice of method has been optimal.

However play testing is very important in game design, so it is possible that some aspects of our games will have to be changed in an actual implementation, even though we have tried to minimize this by thinking thoroughly through all game rules and balancing factors. In the end, though, no planning and safeguarding in design can replace the hands-on experience of actually playing a game, so we can only hope that our efforts in game balancing have been worthwhile and that our game concepts will eventually pass the ultimate test - actually playing them.

Chapter 15

Further work

As this report only covers the basic design of the game concepts, future work groups continuing our work would have to actually develop the games to advance on our work. In this chapter we describe some of the considerations that would have to be made to successfully create the games described in our designs.

15.1 Software Architecture

While we have described a very high level system architecture for each game concept, it is eventually up to the developer of the game to decide upon an architecture that he/she is comfortable with and that fits the environment the game will be run in.

As the MOOSES framework is not completely finished, it is difficult to accurately create system architectures that cover all eventualities. This, combined with the scope of this report, is why we have chosen not to go into further detail on the system architectures of the separate games, apart from where such explanations have been important to understand details in the game designs themselves.

15.2 Play Testing

In any game development project, play testing is alpha and omega to ensure that the game is thoroughly balanced and that the rules and parameters of the game suits the concept. As this assignment does not include any implementation of the games themselves, it is important to keep this in mind when considering the specific rules of the game concepts.

While we strongly believe that all the five game concepts described in this report represent very viable MOOSES games, we are also sure that an actual implementation of any of the games would require rigorous testing of the game mechanics as well as many tweaks and small changes to the finer points in the inner workings of the games.

In addition to this, the many parameters that we make no attempt to define in this report will need to be found and properly defined. For example, while we outline a fitness system and the concept of the shot bar in the football game, we say nothing at all about the exact accuracy needed in the timing of shots or about the rate of decrease of fitness (apart for some illustrative numbers to illustrate the idea). The reason for the lack of such details in the game concepts is that we believe that such decisions needs to be taken by the game developers themselves while creating the game, as is only after the game is playable in some way that it is possible to know exactly which values of such variables would offer the right balance between challenge and accessibility for the player.

15.3 Future Proofing

As we describe in our brief look at hardware consideration, the technology behind the MOOSES project is currently at an important point in its development. While the technology to fully utilize the most advanced digital projectors in gaming systems is not fully developed, we believe that it is only a question of a small period of time before commercially available graphics systems have caught up with the demands of the high definition theatre equipment.

Most probably, this will happen before the concept of MOOSES gaming become widespread, and it is therefore important that games implemented for the platform are built in a way that enables easy upgrades to graphics subsystems without changing too much of the underlying technology.

Similarly, the Java technology used to enable control via mobile phones are not completely mature yet. There are still differences in implementation between different hand sets and manufacturers. However, we feel confident that this situation will not last too long, as there is continuous efforts made in turning mobile phones into powerful software platforms on their own.

After all this is said, however, we believe that the technical concerns regarding MOOSES gaming does not pose any important obstructions to the MOOSES concept itself. Due to the nature of crowd gaming, the content and playability of the games - the ”fun factor” - will be much more important than any technical solutions and/or graphical sophistication. And, at least in our humble opinion, our games should have enough of this intangible quality to succeed.

Chapter 16

Conclusion

When we first began working on this project, we wanted to create a set of games that enabled a large number of players to share the fun of playing games.

Through an extensive research effort into multiplayer games and their characteristic factors we feel that we have been able to create five quite different game concepts that represent a solid set of games for the MOOSES framework. We are very pleased with the way the concepts has turned out, and hope that we one day get to see them implemented as actual games.

We have also identified what we believe to be of the most important factors that needs to be taken into consideration when designing a multiplayer game that is to be played by many players on a single screen.

One additional benefit of this study, one that we did not necessarily foresee when starting up, is that we also ended up looking quite thoroughly into methods of controlling games with mobile phones, work that we hope can be used further on - both in MOOSES contexts and other environments.

Part V

Appendices

Bibliography

[1] *The Book of Games.* gameXplore, 1 edition, 2006.

[2] Ghost in the cave. http://www.speech.kth.se/music/projects/Ghostgame/, 04 2007.

[3] Msnbc's newsbreaker live goes live. http://kotaku.com/gaming/taste-the-rainbow/msnbcs-newsbreaker-live-goes-live-258765.php, 05 2007.

[4] Tekken 3: General moves and conventions. http://www.netcomuk.co.uk/ mx-avier/TEKKEN3/moves.htm, 05 2007.

[5] Unreal tournament 2004. http://en.wikipedia.org/wiki/Unreal_Tournament_2004, 05 2007.

[6] World of warcraft: The burning crusade continues record-breaking sales pace. web, http://www.blizzard.com/press/070307.shtml, 03 2007.

[7] Rusel Demaria and Johnny L. Wilson. *High Score - the Illustrated History of Video Games.* McGraw-Hill, 2002.

[8] IEEE. Ieee standard glossary of software engineering terminology. *IEEE*, IEEE Std 610.12-1990, 1990.

[9] Jan Krikke. Samurai romanesque, j2me, and the battle for mobile cyberspace. *IEEE Computer Graphics and Applications*, January/February 2003.

[10] Sverre Morka, Aleksander Baumann Spro, and Morten Versvik. Multiplayer on one screen - a framework for multiplayer games on one screen. Technical report, NTNU, 2006.

[11] Gianni Pasolini. Analytical investigation on the coexistence of bluetooth piconets. http://ieeexplore.ieee.org/iel5/4234/28579/01278302.pdf.

[12] Marie-Louise Rinman, Anders Frieberg, Ivar Kjellmo, Antonio Camurri, Damien Cirotteau, Sofia Dahl, Barbara Mazzarino, Bendik Bendikesen, and Hugh McCarthy. Eps - an interactive collaborative game using non-verbal communication. *Proceedings of the Stockholm Music Acoustics Conference*, August 2003.

[13] Claes Wohlin, Per Runneson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering, an introduction.* Kluwer Academic Publishers, first edition, 2000.

Concept Art

Although the following concept art pictures were also shown in their respective contexts, we feel that their limited size in the report does not do them full justice.

Therefore, we here show them in full page size to ensure that their details are not lost.

Again, thanks to Seth Piper for his excellent work with these drawings, as well as helping us understand what we really wanted with the games by forcing us to consider how their graphics should be presented.

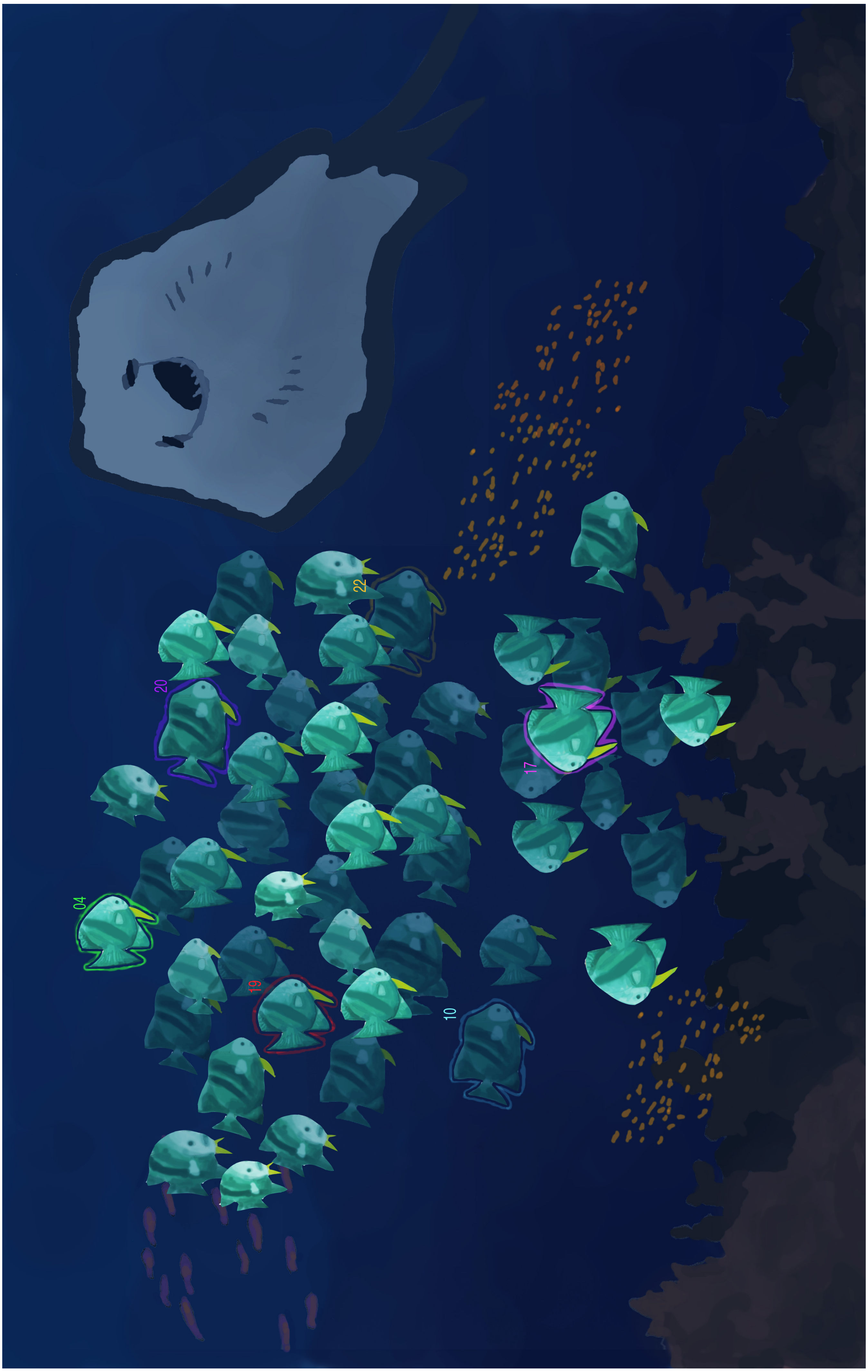
The Music Game

Let them know you realize
that life goes fast,
PLAYER 1
PLAYER 2
'Do You Realize?' by The Flaming Lips
PLAYER 3
PLAYER 4
B
A
A
B
A
B
A
A
B

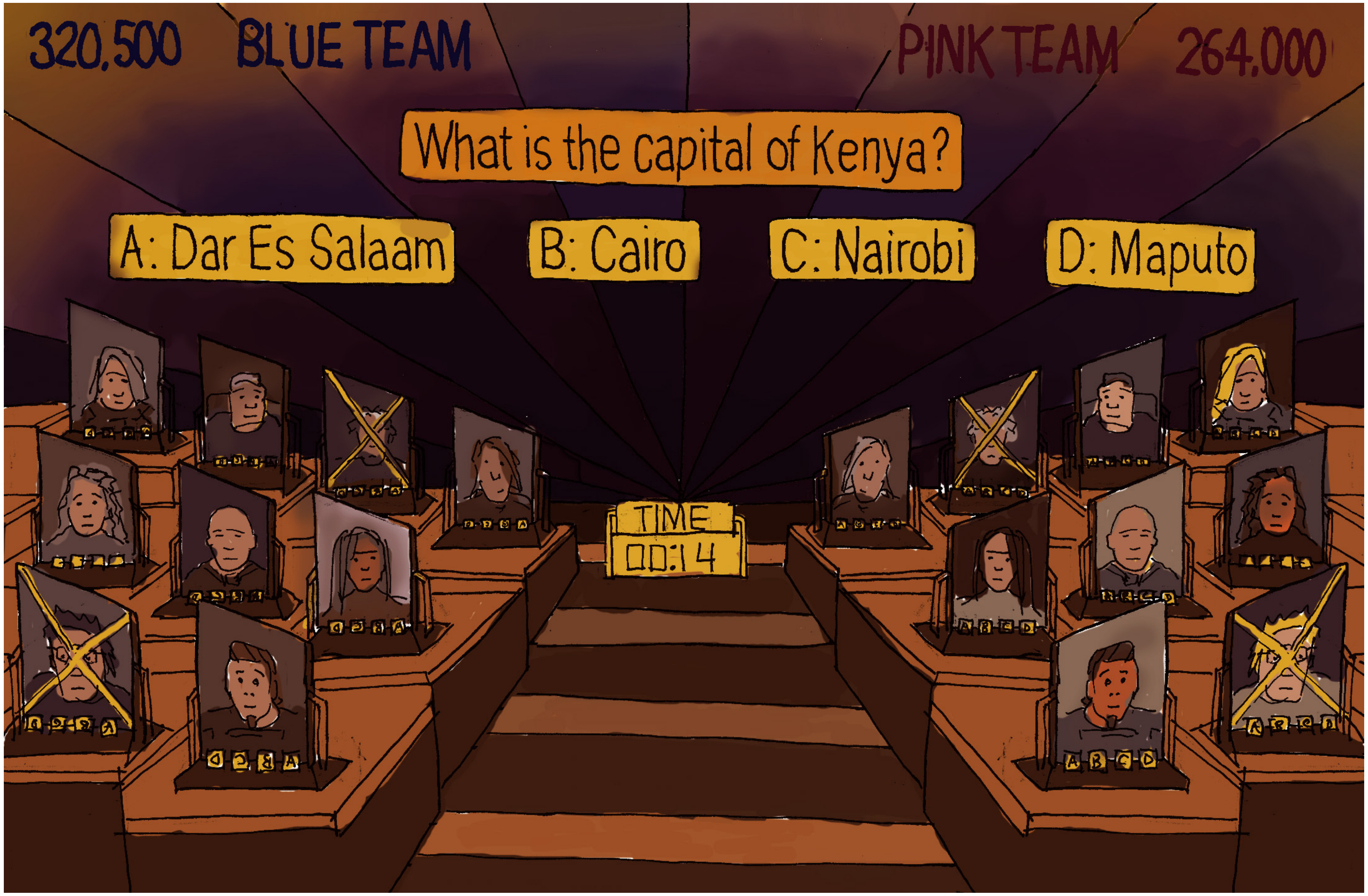
The Football Game

SHOOT! 6/11
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11

The Survival Game


22
20
17
04
19
10

The Quiz Game


320,500
BLUE TEAM
PINK TEAM
264,000
What is the capital of Kenya?
A: Dar Es Salaam
B: Cairo
C: Nairobi
D: Maputo
TIME
00:14