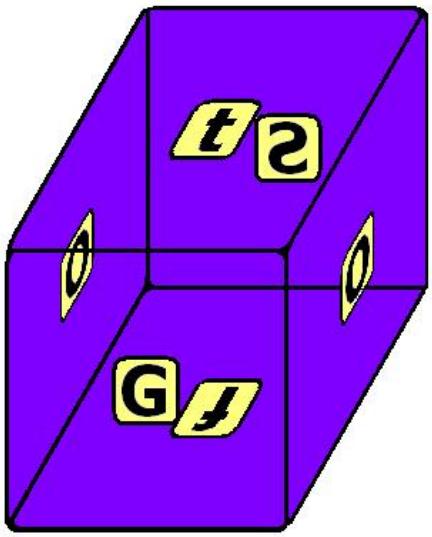
CENG 490
Senior Project

"A 3D – Massively Multiplayer Online Game"
The Ma3e-3D

Test Specifications Report

by



Ömer Akyüz e1347079
Önder Babur e1347186
Süleyman Cincioğlu e1347277
Güneş Aluç e1462670

Table of Contents

1. Introduction

1.1. Goals and Objectives

The Ma3e is a massively multiplayer online game supporting realistic 3D graphics. Core functionalities and the game's expected behavior have clearly been outlined in former documents (e.g. Final Design Report). Although the testing activities span the lifetime of the project development and implementation, the items addressed in this document generally refer to the tests conducted during later stages of development and implementation. Taking into account restrictions on timing, resources and budget; the goal of the tests will be to guide the developers for a product that satisfies the following:

- In terms of server functionalities, the product should offer reliable, fast and error-resistant services;
- Game state consistency should be established as not to interfere with the user's perception of the game world. Specific to our scenario, the following objectives can be extracted:
    - Player movements (e.g. walk, jump, change orientation) within a single room should be highly synchronous,
    - Multi-room events (e.g. join room, exit room, change room) should be consistent as viewed from all players,
    - Player-object interactions that involve a chain of events should be consistent among the players in the same room;
- Assuming some minimum requirements for the underlying hardware/software, scene rendering should be fast, realistic and smooth enough as not to disappoint the target audience.
- Both in technical and commercial aspects AI functionalities should be seamlessly integrated into the platform.

1.2. Document Scope

The following items give some brief information about the contents of this document. It must be noted that the document was prepared inline with the unique aspects of the Ma3e project, therefore distinct testing features specific to a massively multiplayer online game are also addressed.

As previously mentioned, testing spans the whole lifetime of development and implementation of the Ma3e. In this respect, some of the tests mentioned in this document have already been conducted by developers during former stages. These tests will also be addressed in this document, with a clear distinction of their completion.

On a broad scale, the document contains information on the:

- Methods by which the game has been/will be tested,
- Procedures that have been/will be followed in relation to a specific testing activity,
- Resource and time allocation plans for conducting the testing activities in parallel with project development and implementation.

1.3. Testing Plan Scope

Various types of testing activities exist for fostering the product to meet its predefined/expected requirements: unit tests, integration tests, validation tests and some high-order tests such as performance tests, stress tests, alpha and beta tests. These types of tests can be applied in almost any software engineering practice. The following items describe how each testing activity has been tailored to meet the specific needs of the Ma3e:

- ***Unit Testing:*** Based on the specific needs of each module and trade-offs between performance and resources available, the tests conducted separately for different modules of the Ma3e (Graphics, Network, Puzzle Deployment, AI, User Interface) are described.
- ***Integration Testing:*** The testing activities under this title are inline with the different integration stages the product goes through during its development. They are to justify that the components – as they collaborate – work as predicted. The integration stages can be outlined as:
    - Integrating the Graphics and Network modules to produce a game with multi-player functionalities,
    - Integrating the User Interfaces on top of the previously established core to provide high-level interaction capabilities to users (menus, chat, etc.)
    - Integrating the Puzzle Deployment module so that the predefined scenarios are realized.
- ***Validation Testing:*** These tests are performed to monitor and examine the weaknesses of the product during its different stages of integration, so that the goals set in 1.1Goals and Objectives are met.
- ***Performance Testing:*** Performance of the Ma3e can be examined in two different categories: server reaction/processing promptness and rendering performance. Tests will be conducted to monitor and enhance product performance in each of these categories.
- ***Stress Testing:*** Load performance and game state consistency will be monitored as the number of virtual players in the game is increased.
- ***Alpha and Beta Testing:*** Within the scope of this coursework, no beta testing will be conducted. However, to get a sense of how the target audience will react when it is first exposed to the product and to determine more bugs, Alpha tests will be carried out.

1.4. Constraints for Testing

In the following sections, the constraints that were taken into consideration when determining the scope of the testing activities are outlined.

1.4.1 Time

Up to now; unit tests and various integration and validation tests have been performed to assess product maturity. Due to the approaching deadlines, and relatively heavy still-remaining-schedule in development and implementation, testing will not be conducted on a thorough scale. However, in order to tackle this difficulty and reduce risks associated with “last-day found” defects; the devised test scenarios that respect the future workload, have been and will be carried out in parallel with the development efforts. A detailed plan can be found in section 4 Testing Schedule.

1.4.2. Staff

GOSOft consists of 4 members. Due to the unique nature of the Ma3e (dependence on human interaction/observation); except for some stress tests, software automated approaches are not practical. Otherwise, efforts in developing software automated tools may exceed the total effort spent in developing and implementing the product. Therefore, for the testing activities apart from a restricted subset of the stress tests, the participation of all team members is necessary. Consequently, the tests specifications and the nature in which the tests are conducted should highly be restricted.

2. Testing Strategy and Procedures

In this part of the test specification document, the procedures, ways and strategies that are going to be followed during the testing will be mentioned. First we will introduce the test items that we will test and then we will mention the tests that will be performed.

2.1. Test Items

Our project “Ma3e” consists of many modules, and all of the modules that the project consists of will be tested according to their individual functionalities and overall interactions of these modules with eachother.

The main modules of our project are listed below:

- Graphics Module
  - Static Graphics
  - Animation Module
- Network Module
- Puzzle Deployement Module
- AI Module
- User Interface
  - I/O Module
  - Chat Module
  - Menu Management

2.2. Unit Testing

In our project “Ma3e” unit testing is considered in two main development areas which are module tests and data. But most of the tests about unit testing have been completed during the development of the project, so from now on we consider to do unit testing only when we believe that the results will be really useful and beneficial. In such a case we will write test cases for the related classes and their functionalities on its own. Inheritence is also considered in unit testing.

2.2.1. Module Tests

One of two major parts of the unit testing is module tests. Our project consists of 5 major modules and some submodules. Most of the modules are very near to be finished, but also some of them needs more time to be completed. Module tests are conducted in the development phase of the related module. And unit testing is performed on the modules:

- Which have critical requirements for their interior processing
- Whose processing results can be observed easily also while working independent of the other modules.

So far we performed unit testing on our Network Module in the beginning of the second semester and first semester. We have used black box testing strategy for it. We have tested the reliability of the network module by this test.

Graphics Module’s unit testing was basically based on the data it contatins so we didnt need to test it by unit testing.

AI Module is embedded in the Graphics and Network Module so much that its testing can only be done in the Integration Testing.

User Interface is also very important module of our project “Ma3e”. It consists of 3 submodules: I/O Module, Chat Module and Menu Management. I/O Module and Chat Module’s module tests are completed. We use black box testing for their classes and functions. Menu Management is still being constructed but its testing will be intergration testing.

Puzzle Deployement Module is also under construction, and it will also not be tested in unit testing. It will be similar to AI module because Puzzle Deployement Module will also be very embedded to Network and Graphics Module, and it is very difficult to test it individually. Also time consideration is another issue about this.

2.2.2. Data Tests

There are some data items to be tested in our project. All of them are related to the Graphics Module. They are listed below:

- **Static Models:** Static models will be tested for their integration to the program. Since during the exporting step of the model problems related to the material properties, textures occur, each model must be examined for correctness of these items.

- **Animated Models:** In order to avoid problems in defined animations of each Model, texture problems, material properties related problems; all animated models must be examined for each model that is going to be in the final package.

2.3. Integration Testing

Our aims for using integration testing are as follows:

• Check whether modules work together properly or not
• Find out the defects that could not be identified by the other testing strategies

The modules are constructed independently, and then when all the functions and fields are prepared the module is integrated to all ready related modules. Before the integration step all required test steps are prepared and performed. In that way there will be no need to do extra tests after the integration up to new module added to the system. As a result, at the end of the project we are going to have fully integrated and tested software in terms of module integration. By means of integration testing, which is broad story-level tests, we are able to verify the interactions between the various actions supported by the application across all controllers.

We have considered our project's integration test in 3 phases:

- **Graphics-Network Integration Test:** Graphics and Network Modules are the core modules of our game. By integrating two of them we developed a 3D multiplayer online environment. The integration test of this core part was completed. Two modules are consistent with each other.

- **User Interface Integration Test:** The first add-on to the core part of the game is the user interface. By means of user interface 3 sub modules have been developed which are: I/O Module, Chat Module and Menu Management. 3 of them are much related to both Graphics and Network Modules. By these sub modules we will have a much player oriented game. The integration tests of these sub modules are almost finished.

- **Puzzle Deployment and AI Modules Integration:** Last layers of our game are Puzzle Deployment and AI Modules. These modules are needed for an enjoyable game play. Both of these modules are under construction and their integration testing will be held just after their completion.

2.4. Validation Testing

2.4.1. Requirements Validation

The purpose of these tests will be to ensure that the product, in its current stage of integration, meets the predefined requirements. These requirements have been identified in section 1.1 Goal and Objectives. For convenience, they are listed again in this section, with an associated list of checklist items.

Requirement Description	*In terms of server functionalities, the product should offer reliable, fast and error-resistant services*
	• Does the server ever get into an error-state? • Is there a need to restart the server to reestablish game state consistency? • Is there any significant lag in received messages due to a busy server? • Does the server suffer from unpredicted connection-related-behavior due to a client application (e.g. connection close)?

Requirement Description	*Game state consistency should be established as not to interfere with the user's perception of the game world.* *a) Player movements (e.g. walk, jump, change orientation) within a single room should be highly synchronous*
	• Are the players in their reported positions when idle? • Are the players in their reported positions when in motion? • After a long period of continuous motion, are the players in their reported positions? • Are jumps perceived as they appear on the generating application? • When the players change their orientation, is it perceived as a smooth animation on recipient applications?

Requirement Description	*Game state consistency should be established as not to interfere with the user's perception of the game world.* *b) Multi-room events (e.g. join room, exit room, change room) should be consistent as viewed from all players*
	• As perceived on a recipient application, does the join-room event of a player occur simultaneously with that of the actual? • As perceived on a recipient application, does the exit-room event of a player occur simultaneously with that of the actual? • During the change-room event of a player (i.e. after exit-room and before join-room), can the player be observed in any of the rooms? • During the change-room event of a player (i.e. after exit-room and before join-room), are there any irrelevant event messages sent to the player? • Does the player that joins a room receive information on all the players currently in the room?

Requirement Description	*Game state consistency should be established as not to interfere with the user's perception of the game world.* *c) Player-object interactions that involve a chain of events should be consistent among the players in the same room*
	• When a player joins a room, is the state of all objects (i.e. current progress made on the puzzle) faithfully transmitted to the player? • When a player interacts with an object, is the change-of-state reported to other players in the room? • If for some reason the recipient loses/does not process an event in the chain, are the states of objects synchronized by the next event(s)? • Does the application take precautions as to disallow concurrent modification on the state of an object?

Requirement Description	*Assuming some minimum requirements for the underlying hardware/software, scene rendering should be fast, realistic and smooth enough as not to disappoint the target audience.*
	• Does the camera respond smoothly to mouse motion? • Is one's own motion consistent with expected speed and underlying physical rules (e.g. walking, jumping, etc.)? • Is the recipient able to generate the realistic animation of other players' motion? • Does the recipient get a vivid image of textures while a) idle b) in motion? • Do the user interfaces interfere with normal game playing? • Are the transitions from one menu state to the other smooth?

Requirement Description	*Both in technical and commercial aspects AI functionalities should be seamlessly integrated into the platform.*
	• Does the game support basic random motion with path finding support algorithms for the AI players? • Do the AI players provide unidirectional chat-based support to players in the group for puzzle solving? • Can the AI players move around the Cube just like any other player? • Does the presence of AI generate any observable quality/performance degrade in any of the checklist items present in this/preceding sections?

2.4.2. Design Validation

As the project does not allow an incremental approach, to obtain the maximum output from the project resources; many of the features outlined in the project design need to be followed during development and implementation. Furthermore, if found necessary, changes made on the design should be discussed among the GOSOft members until consensus is reached. This approach is critical to ensuring that the changes are taken in the right direction.

Of course, compliance with the design considerations cannot be assessed unless some measurements are collected. The results of the design validation tests performed at each stage of integration (as outlined in sections 1.3 Testing Plan Scope and 2.3 Integration Testing), will be used merely for this purpose. The white box testing strategy will be utilized during this process. This approach will enable the developers see whether or not the class and module interactions are in synchronization with those in the detailed design documents.

2.5. High-Order Testing

In addition to the tests above we also plan to perform some high order tests. You can find more information about them below.

2.5.1. Performance Tests

Performance of the Ma3e can be examined in two different categories: server reaction/processing promptness and rendering performance. Tests will be conducted to monitor and enhance product performance in each of these categories.

2.5.2. Stress Tests

Load performance and game state consistency will be monitored as the number of virtual players in the game is increased. We have also some useful scenarios for stress test.

- Load the game with as more players as we can and move these players. We will check the game consistency and frame rate.

- Try to change the room with lots of players in the same moment and check if the game crashes.

- Try to load the chat module as much as we can and check if it crashes or not.

- Load the game with as more AI players as we can and check the game consistency and frame rate.

- Load the game with as more rooms as we can and check their integrity.

- Load the game as more items (objects) as we can and check the frame rate.

2.5.3. Alpha and Beta Tests

Being a game project, the project “Ma3e” does not have a specific customer. On the contrary its possible customers are in fact anybody who can play games. This situation makes alpha and beta testing very important (i.e. more than any other project) for “Ma3e”. But we have only 5 weeks left and still have modules to be finished. So we are only planning to conduct alpha testing. For alpha testing we will use 2 or 3 volunteer players. These numbers can change according to time But we will not do beta testing due to time constraint and the requirement of development environment.

3. Resource Allocation Plan

The resources to be allocated at all stages of testing are limited to the following:

- Group members,
- Test automation software to be developed for a subset of stress tests.

Unless these resources are allocated in an efficient way, testing cannot be conducted in parallel with the ongoing development and implementation activity. Consequently, the group members have decided to follow the following plan for each activity. The corresponding schedule is available on section 4 Testing Schedule.

- ***Test Plan Delivery:*** after discussions among group members and when a consensus is reached on the plan to follow, the associated “Test Specifications Report” will be prepared according to which all testing activity shall be conducted.
- ***Unit Tests:*** The unit tests of all modules except for the Puzzle Deployment module have been completed. The remaining activity will be carried out by the sub-team responsible for the development of this module prior to its integration into the framework.
- ***Integration Tests:*** Most of Phase-1 and Phase-2 integration tests have been completed. Further tests will be performed in Phase-2 as the developers fix the bugs associated with Phase-2 integration. With the integration of the Puzzle Deployment module, Phase-3 integration test activity will begin. Tests preferably require the presence of 3 group members, 2 being the minimum.
- ***Validation Tests:*** Informal validation tests have been conducted at each integration phase. However, the group members will meet twice before the deadline for completion of this activity (23.05.2007) to formally conduct these tests. These meetings require the full attendance of the group members. Tests will be carried out based on the checklists provided in this document (see section 2.4 Validation Tests). One group member will be responsible for documenting all deficiencies. Formal documentation will be avoided due to time constraints.
- ***Stress Tests:*** One week before the deadline, the automated test application will be developed and integrated into the framework. Two independent team members will individually run the application and record their observations. These observations will informally be discussed among the group.
- ***Alpha Tests:*** As soon as the product is delivered as an alpha release, with all expected functionalities implemented and associated tests completed, alpha tests will begin. The tests will be conducted on a daily basis for a duration of 7-

days and at least 2 hours/day. Participation of at least 3 group members in each test session is mandatory. Observations will informally be recorded and analyzed by the participants by the end of the session. Necessary modifications that do not hinder the timely delivery of the product and that do not further decrease the quality of the software will be made in this period. The remaining hours on each day will be allocated to correction. Validation tests MAY be repeated at this stage, if found necessary by the group members.

4. Testing Schedule

TASK	DEADLINE
Test Plan Delivery	07.05.2007
Unit Tests	07.05.2007(completed)
Integration Tests	23.05.2007
Validation Tests	23.05.2007
Stress Tests	23.05.2007
Alpha Tests	27.05.2007
Correction	01.06.2007