MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 492
SENIOR PROJECT IMPLEMENTATION

Spring 2005

Test Specification Report

WIRTUAL WISION
'Beyond The Sight'

Doğan Sever
Ercan Şahan
Alper Taş
Kubilay Timuçin
Rakıp Yaşar

May 2005

Table of Contents

1. INTRODUCTION

1.1. PURPOSE OF TEST SPECIFICATION REPORT

The purpose of this document is to describe the scope, approach, resources, and schedule of the testing activities that will be carried out within the one-month period following the first release of Beyond The Sight's beta version. Test specification report also identifies the modules, items and features to be tested, testing tasks to be performed and personal responsibilities for each task, module, item or feature.

1.1. SCOPE OF THE DOCUMENT

Along with this piece of writing, one first finds the testing strategies that will be applied during the testing activities. Then, the game components are discussed from the view point of testing activities. Lastly, planning and requirements of testing is covered in the later section.

Reading this document, it will be helpful to refer to the product's requirement analysis report and detailed design report to achieve a good understanding of the testing activities that are supposed to ensure both functionality and durability of the game in that the software is tested in accordance with these aspects.

2. TESTING STRATEGY

In a simplistic view, testing is to identify bugs found in the software, so the problem can be removed. To do this, there are two basic testing strategies that will be applied during the testing activities of Beyond The Sight. Namely, these are **black-box** and **white-box** testing.

In black-box testing, group members will simply play the game knowing how to observe all the functionality issuing keyboard and mouse controls. Since black-box testing focuses only on the functional or the playability aspects of the game, what is to be tested in here are the selection menus, the use of buttons, the "look and feel" of graphics, sounds and animations, overall what we call as "actual game play".

In white-box testing on the other hand, group members will focus on the architecture and integration aspects of the game software. The interaction and integration of software modules will be tested in code-detail. The game loop will be debugged with all possible usages of software deploying sample game maps. Since Beyond The Sight has been designed and implemented according to the idea that all game data is loaded from external files to the game engine, it is possible to first generate little game maps that contain the level information and than load the sample map for testing purposes.

Next section clarifies the game components that will be under consideration of testing activities.

3. GAME COMPONENTS TESTING

In this section, the components to be tested are mentioned under two approaches. As mentioned earlier, the features will be tested according to the black-box strategy by just playing the game. Then, the modules will be considered when it is time to test the software code according to the white-box testing strategy.

3.1.TESTING GAME PLAY

The features to be tested within the game flow are as follows:

- Light,
- Shadow,
- Reflection,
- Overall rendering,
- Camera views,
- Hero states,
- Hero movements,
- Hero interactions,
- Object interactions,
- Collisions,
- Non-Player Character (NPC) states, movements.
- Main menu buttons,
- In game menu buttons,
- Configurations.

Firstly, the correctness of the selection of right light position will be tested against the location of the main character, because the light positions in Beyond The Sight are determined dynamically according to the location of the hero in that each room in the game has a light and moving room to room it is expected to observe different lightening. Observation of light position, however, is only possible by inspecting the positions of shadows of the hero and other objects.

Secondly, shadowing will be tested against the light. This might seem to be ambiguous since lights are said to be tested against shadows, but here what we test is only the correctness of the selection of right light position. In other words, shadows will be tested again against the location of hero moving within the room.
Thirdly, reflection of the objects on the ground will be tested against the positions of objects in 3D environment. For example, moving an object upward, the reflection of the object on the ground will be tested.

In the game, rendering unit only renders the current environment and the objects in it. Since the game environment is divided into rooms, it is expected from the game to display only the current room and not to render other rooms. This will be achieved by looking at the other rooms using the dummy gates prepared for this sort of testing.

Camera positions are also selected dynamically according to the hero positions. There might be more than one camera in a single room and isometric views change to achieve the pre-defined camera position according to the position of the main character. Also, one of the cameras can be set to first-person-view so that it changes to first person view when the hero is in the corresponding location matched with that camera. So, cameras will also be tested against the hero position.

Furthermore, the hero states and the correctness of transitions between them are also to be tested according to the pre-defined transition rules. This includes the animation and sound synchronization of each state of the hero. What is to be tested is the compliance of hero reactions to all possible inputs. Idle to walk, run to idle, pick to idle, use to idle are some of the state transitions whose correctness are tested with respect to the correct animation and sound.

The correctness of hero movements is another item to be tested. This will also be tested by exposing the hero to all possible keyboard and mouse inputs and observing the correctness of the movement. For example, clicking on a point on the ground or an object, the hero is supposed to find the shortest path to that position and arrive at there. Doing this hero is expected also not to crash into the objects in the environment. Also, moving the hero using keyboards arrow buttons, tester will ensure the directional and positional compliance of the hero.

The interaction between the hero and the environmental objects is yet another feature to be tested. Hero is expected to interact with the objects in the environment when they are clicked with the mouse. As mentioned in the preceding paragraph, the hero will first get close the object clicked. If it is already close enough, the object is supposed to disappear from the environment and appear in the inventory slots if it is an inventory item and no action is taken otherwise. To illustrate, a screwdriver can be interacted but a sofa may not.

The interaction between objects is supposed to be achieved when the player decides to use one of the items in the inventory slots. Suppose a screwdriver is selected and the player issues the use command on a screw. It is expected from the screwdriver to trigger the screw from steady state to another state, let's say rotation state. This way, all pre-defined object relations are to be tested. Testing key-door relation may seem to be clearer to understand what is meant by testing object relations in the environment.

Since an external library, Open Dynamics Engine (ODE) is used for handling the collisions, this feature will not be tested in code-detail, but with respect to the durability of the software. Particularly, different numbers of geometrically and physically defined objects will be exposed to collisions to find an approximate collision overload that the software can tolerate.

Moreover, NPCs' states and their transitions will be tested by observing the animations and sounds performed during an interaction with the main character. An intelligent enemy character, for example, is supposed to change its state to follow the hero once the hero falls into the field of vision of him. Changing the state, the correct animation and sound should be observed. NPCs are supposed to detect the hero, follow him along with the shortest path to the hero and kill him when they get close enough.

Lastly, the functionalities of menus (main mane, in-game menu, inventory menu) are to be tested. The correctness of the game actions against menu commands will be checked. Arranging game options through the menus will be tested. Enabling/disabling shadow and/or reflection and increasing/decreasing sound volume will all be tested.

3.2.TESTING GAME ARCHITECTURE

Testing the game architecture, the debugger tool of MS Visual Studio is the key item to perform a most satisfactory, consistent and efficient testing in code-detail. The methodology to be applied is to input the software and debug all modules and thus code segments line-by-line.

Firstly, the data pre-processor module that is responsible for loading the game map will be tested if it reads the map file correctly and fill the corresponding data structures without any fault. Object inspectors of debugger will be helpful to observe the fulfillment of loading the map.

The loader of 3D models will not be tested since it is an external resource imported and assumed to load the models correctly.

The graphics engine will be tested by inspecting the render loop against different game states to assure that it goes into the correct condition and renders the desired screen according to the game state and configuration.

Sound generator module will only be tested simply listening the sound against correctness, quality, length, and loop condition. This might somehow seem to conform with only black-box approach, however, developing the software an

external library (SDL) is used and thus code-wise inspection is not reasonable in this case.

The game engine will be inspected against input handling actions. Again, debugger will be used to go over the lines of input handler to test if the game reacts correctly by running the correct code segments when an external input issued.

Next section gives the test plans of each feature, item or module of Beyond The Sight in time and person.

4. TESTING PLANNING

BLACK BOX TESTING		
Test Task	**Time**	**Person**
Light	11 Mayıs 05	Ercan Şahan
Shadow	13 Mayıs 05	Doğan Sever
Reflection	15 Mayıs 05	Doğan Sever
Rendering	17 Mayıs 05	Alper Taş
Camera	19 Mayıs 05	Ercan Şahan
Hero states	21 Mayıs 05	Alper Taş
Hero movements	23 Mayıs 05	Kubilay Timuçin
Hero interacitons	25 Mayıs 05	Rakıp Yaşar
Object interactions	27 Mayıs 05	Rakıp Yaşar
Collisions	29 Mayıs 05	Kubilay Timuçin
NPC states, movements	31 Mayıs 05	Ercan Şahan
Menus, configuration	2 Haziran 05	Doğan Sever

WHITE BOX TESTING		
Test Task	**Time**	**Person**
Debugging Loaders	11 Mayıs 05	Ercan Şahan
Debugging AI Manager	13 Mayıs 05	Kubilay Timuçin
Debugging Renderer	15 Mayıs 05	Doğan Sever
Debugging Input Handler	17 Mayıs 05	Alper Taş
Debugging Sound Generator	19 Mayıs 05	Kubilay Timuçin
Debugging State Transitions	21 Mayıs 05	Rakıp Yaşar
Debugging Animation Manager	23 Mayıs 05	Doğan Sever