MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 491
SENIOR PROJECT DESIGN

Fall 2004

Requirement Analysis Report

WIRTUAL WISION

'Beyond The Sight'

Doğan Sever
Ercan Şahan
Alper Taş
Kubilay Timuçin
Rakıp Yaşar

November 2004

Table of Contents

1. INTRODUCTION

1.1. Purpose of The Document

This document introduces a detailed description of the software that is to be developed in that it explains the system and process requirements to be met while schematizing the whole system through different diagrams. To state the major milestones to be followed within the entire system development progress is the other main purpose of this report.

1.2. System Purpose

The purpose of the software is to provide the PC game players with an easy-to-learn, easy-to-use adventure game with arcade components. Moreover, one of the goals of the system is to present a new computer game with 3D graphics supported by sophisticated properties like lighting and shadowing.

1.3. System Scope

The name of the game software is "Beyond The Sight". The system will support Computer Graphics (CG), Human-Computer Interaction, and Artificial Intelligence (AI) aspects.

The game is based on missions to be completed by the player who will face with several obstacles towards the final target. In particular, the player plays the role of an agent who is assigned to investigate secret information in a military base of a foreign country, rescue the hostages and/or destroy the base.

As stated earlier in the project proposal report, to present the game and to introduce how the game is played to the players, the software will include several multimedia fragments. The software will also enable the users to add new missions when they finished the parts of the game they already have. It will be possible to download new missions from the Internet. This property will make the game more enjoyable in that it will give the players the chance of playing the game in different maps. The map builder will help us generate the default maps delivered with the game software and the ones to be available on Internet. Moreover, to ease the usage of the game, the players will be provided with little tips that appear when necessary. Unlike the ordinary viewing techniques, the game will have an isometric viewing in which camera views are applied instead of a player oriented vision. This will facilitate the performance of the software in that it reduces the calculation of predicted scenes.

One thing to ease the development of the software is to use script codes that reduce the effort to manage the changes made on the source code during the implementation phase. Within the development progress of this adventure game, scripting will be concerned.

To obtain a more adaptable and easy-to-change software, data-driven approach will be applied. To achieve this, some of the configuration variables will be stored externally rather than hardcoding them internally in the source code. For example, control keys can be subject to change in the future and to store the corresponding keys for specific actions is required in such a case.

Language Processing, which was encountered as an aspect to be supported early in the project proposal report, will not be taken into consideration since this might direct us into an undesired route, regarding the purpose of the software stated in the preceding section. In other words, the time period that the software is supposed to be completed does not seem to allow us to add such a complex feature.

Also, in the earlier document it was stated that the game would be a multi-player game. However, it has later been realized that the nature of adventure games does not well conform to the idea of networking since adventure games are mainly the games requiring neither teamwork nor competence with other human players. It is a fact that the game that is to be developed will contain arcade scenes that might seem to be well fitting the idea of multi-player option. However, it still does not seem to be a good idea to implement this feature since it is highly expensive to deal with possible deadlocks that might occur in a real-time system where more than one user interact with the software.

The difficulty level of the game will not be available to be selected by the user. It is going to be a pre-determined level that is to be implemented in the action scenes by using AI algorithms.

1.4. Acronyms, Abbreviations

3D	:	Three Dimensional
AI	:	Artificial Intelligence
CFD	:	Control Flow Diagram
CG	:	Computer Graphics
COCOMO	:	Constructive Cost Model
DD	:	Democratic Decentralized
DFD	:	Data Flow Diagram
IEEE	:	Institute of Electrical and Electronics Engineers
KLOC	:	Kilo Lines of Code
MB	:	Mega Byte
NPC	:	Non-Player Character
OpenGL	:	Open Graphics Library
PC	:	Personal Computer
STD	:	State Transition Diagram
SyRS	:	System Requirements Specifications
VB	:	Visual Basic
VM	:	Virtual Machine
UML	:	Unified Modeling Language

1.5. References

- Project Proposal Report
- Literature Survey Report
- User Survey Report
- IEEE Guide for Developing System Requirements Specifications
- Pressman, R. S., *Software Engineering – A Practitioner's Approach,* 5th Edition

2. GENERAL SYSTEM DESCRIPTION

2.1. System Context

There are two basic software components that are to be developed besides the game software. These are namely the game installer and the map builder (See Figure 2.1).

To provide an installer program instead of delivering all the files needed for the installation of the game seems reasonable since the media on which the software is going to be delivered has limited storage.

The map builder, as mentioned in the system scope section, will facilitate creating new maps and missions. This way will help us generate new gaming environments decreasing the extra effort that has to be made on forming such maps and missions.

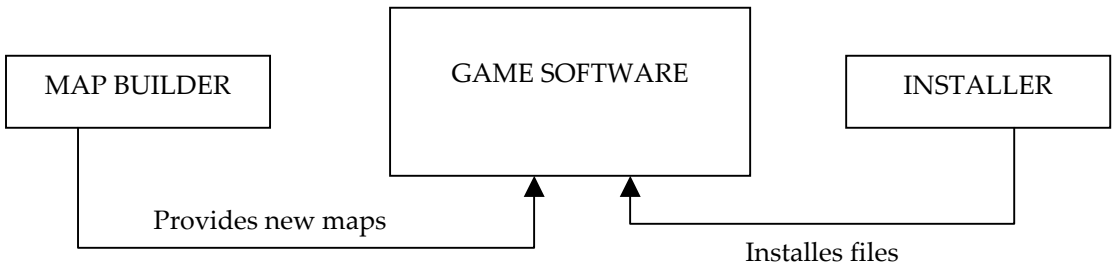

Figure 2.1 – System Context

2.2. System Description

The game software can be examined in six main modules: Menu Management, Game Options Management, Game Save/Load Management, Map (Mission) Load Management, Game Play Mode and Statistical Data Management. These modules have different functionalities giving the players the opportunity to master and play the game effectively.

Firstly, the menu management interface enables the user to view and select alternative game preferences. Through the menus that appear both at the beginning of the game and at any time the user performs the corresponding action during the game flow, the player can start a new game, load a previously saved game, load a new map (mission), quit the game, save a game, manage the game options or exit completely. Menu options are generally selected by means of mouse clicks on the corresponding menu items or keyboard actions.

Secondly, Game Options Management provides the users to configure video and audio options besides monitoring the game controls. In the video options section, it is possible to configure the resolution level, anti-aliasing and shadowing. Adjusting these options, the player can adapt the game to his/her hardware settings to obtain a better performance. The menus controlling the audio preferences provide the user with selecting the level of game sounds (i.e. speech sounds, background sounds, keyboard tones etc.). Lastly, it is possible to view the game controllers via the corresponding menus.

The next module is the Game Save/Load Management. Through these modules, the player is capable of saving games and loading previously saved games. Saving a game, the related module requests the user to supply a title for the game to be saved. This name will be used as the name of the configuration, therefore it is expected to be different from the names of the early saved games. Saving option is provided in only certain phases of the game flow. Also, a limited number of game saves are enabled. When a game is saved; date, player nickname, saved game title and the current game stage information are stored. Lastly, loading a game, a menu lists the previously saved games through which the player selects the game to be loaded.

Map (Mission) Load Management is another module in the system structure. This module will give the users the opportunity to load different maps (missions) that are generated by using the map builder. As mentioned earlier, these maps (missions) will be available on the web site of the game and only provided by the game developer group. Loading a map (mission) is possible by selecting the title from a list displayed by the related menu.

The fifth module to be mentioned is the Game Play Mode. In this mode, the player interacts with the environment and the non-player characters exploring the puzzles to be solved. Whenever the player inputs the corresponding actions, the inventory that the main character has at that time and the objectives at that stage are listed. Starting the game, an introductory movie is displayed to make the player get familiar with the scenario of the story. The smooth transition from the introductory movie to the stage where the user takes the control makes the game more realistic and exciting. In between the stages, cut scenes are displayed within the game flow. In the Game Play Mode, some of the scenes contain arcade components where the player tries to defeat the enemies on which some AI features are implemented. These scenes contain the parts where the player has to use his/her reflexes rather than the ability to solve puzzles.

The last module of the system is Statistical Data Management module. By means of the properties of this module, it is possible to store the statistical data of the games, which then can be viewed to score different players or game sessions. The statistical data involves the completing times of each stage, nicknames of players, the number of sessions to complete the stages and/or games, etc.

Both in design and implementation of the game software, these six modules are the essential parties leading us to proceed in the development progress.

2.3. User Characteristics

Developing a game, it is crucial to find out what game players expect from a computer adventure game. To achieve this purpose, a questionnaire survey on the people who are familiar with this kind of PC games was conducted. The questions mainly seek answers to what people think about the current adventure games in the market and what they expect from the future developments. In particular, the features they like or dislike about the current games were one of the concentrations.

The survey includes the answers of 11 computer game players who have been playing computer games for about 10.7 years in average. The most favorite adventure/action-adventure games among the players are Monkey Island, Tomb Raider and Broken Sword.

Most of the answers indicate that the players give much importance to the scenario and its closeness to reality. They mostly enjoy games containing humorous and funny dialogs. Easy-to-use games are preferable among the players. They are generally keen on the games with multi-way solution scenes that do not limit the movements of the player. Challenging puzzles are preferred. Also, some of the players state that they like improvements in the main character's capabilities proceeding the stages within the game flow. It is another preferable feature for the non-player characters to have some intelligence.

The answers also reveal some game characteristics that the players dislike. The difficulty of strolling around, for example, is an annoying incident. Instead, the players prefer to move to the destinations by pointing them with mouse clicks. Another undesirable characteristic is that some games have too long cut scenes.

2.4. Literature Survey

Performing a literature survey is as important and beneficial as conducting user surveys within the development phase of a computer game. This way helps to find out what current technology and market offer to the game players. Regarding this, a comprehensive literature research was held. This research was earlier reported with the title Survey Report.

Having compared the present adventure games in which aspects they differ from each other, it has been decided for the game to have the following characteristics:

- An attractive background and a real-life story,
- Relevant and well-fitting puzzles in problem solving,
- Deterministic progress in the game scenario,
- An optimum difficulty level,
- A linear approach for the target; one final
- An easy-to-use and intuitive interface,
- Common actions not requiring lots of click or key strokes.

Roughly speaking, the game that is going to be developed will support shadowing, lightening, isometric camera view, controls in real-time. Moreover, the interactive environment will contain special characters with which the hero will be able to make conversations. AI features will be implemented on the non-player characters. The performance and the fluency of graphical components will be a core concern for us in developing the game.

3. SYSTEM ANALYSIS MODELING

3.1. Functional Modeling

In this section, the system is represented with a Data Flow Diagram (DFD), which "depicts information flow and the transforms that are applied as data move from input to output" [1]. DFD of the system that is to be developed is composed of three levels in each of which the system is explained in different levels of abstraction. In the first level (See Figure 3.1 - DFD Level 0), the system is represented in an input-output manner. The two input resources to the main system are the keyboard and the mouse, controlled by the player. The two outputs of the main system are the monitor and the speaker.

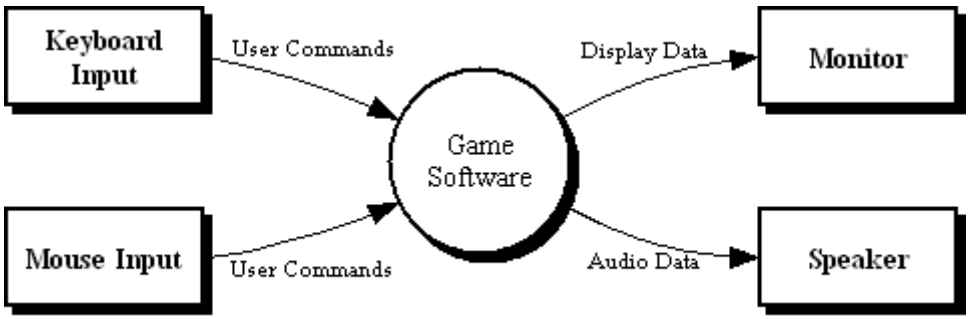

Figure 3.1 - DFD Level 0

User issues the commands to the system via the input resources and gets the system response, which are visual and audio data in this case, from the monitor and the speaker, respectively.

DFD Level 1 (See Figure 3.2) focuses on the main system, named Game Software in DFD Level 0. The system is composed of six modules and the data flows among them. Also, the files containing external data on the disk participate to the data flow. The six modules are: Physical Engine with Setup Manager, Event Listener & Handler, AI Engine, Graphical Engine, Sound Engine and Data Pre-Process.

[1] Pressman, R.S., *Software Engineering – A Practitioner's Approach*, 5th Edition, p. 311

The whole physical environment that accommodates all the interactive and non-interactive objects is contained in the Physical Engine Module. This module also manages the global variables of the system. This module directly accepts the user inputs. Interacting with all the other modules, Physical Engine plays a central role.

Event Listener & Handler Module, as the name implies, listens and handles the internal events that occur within the game flow with the impact of the internal interactive objects. It reacts the Physical Engine informing the necessary actions to be performed.

Who manages the decision-making process of the "intelligent" non-player characters is the AI Engine Module. The Physical Engine informs it about the environmental changes and the AI Engine determines the actions of the intelligent characters.

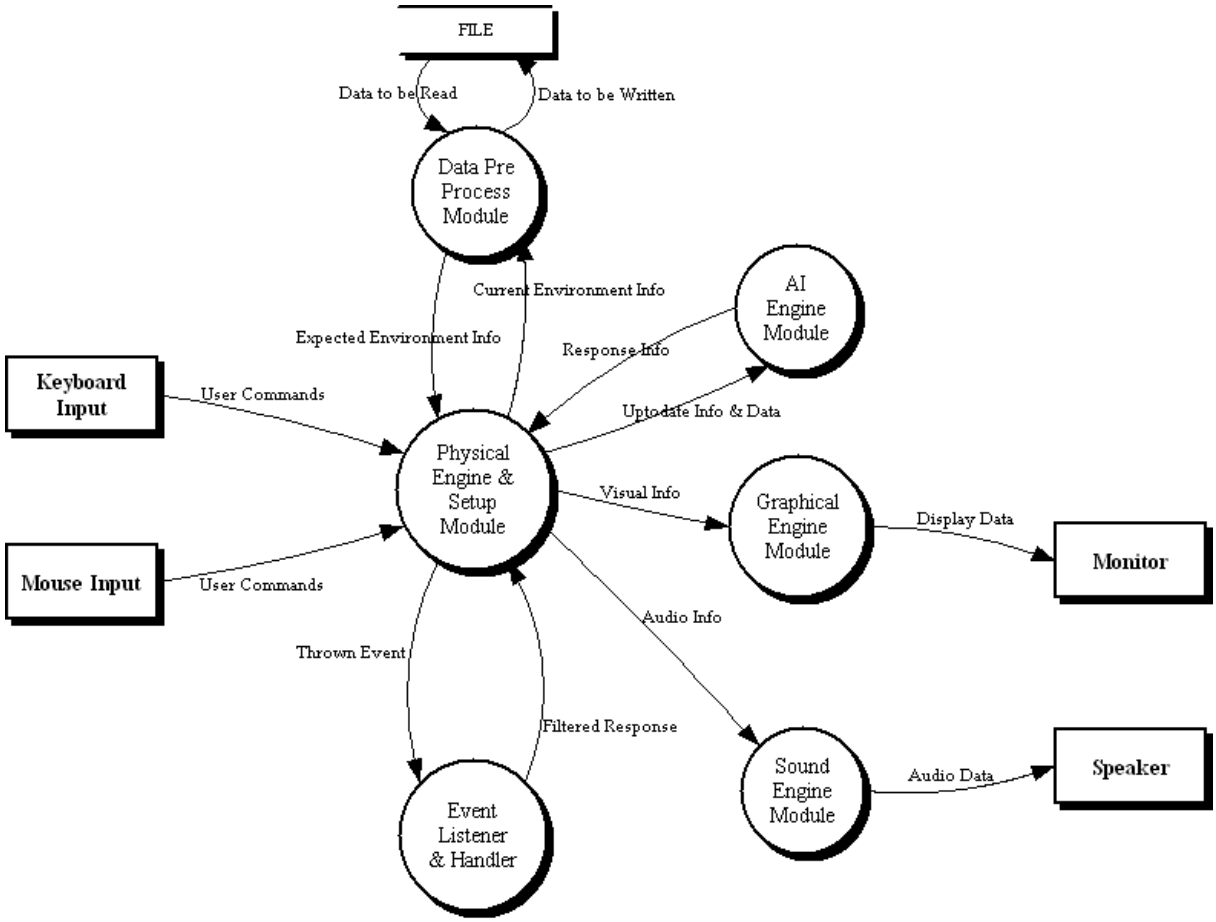

Figure 3.2 - DFD Level 1, Game Software

Data Pre-Process Module manages the load/save operations. It is also responsible for loading a new map (mission). To do these, it reads the files on the disk, fills necessary data structures and transmits these data to the Physical Engine. On the other hand, it writes the game save data to files. It is also the Data Pre-Process Module's duty to handle all file read/write operations of the whole system.

Graphical and Sound Engine Modules manipulate and transmit the visual and audio data that they receive from the central Physical Engine to the output devices, respectively.

In DFD Level 2 (See Figure 3.3), inner structures and the data flows of the Physical Engine & Setup Module and Data Pre-Process are the consideration.

Setup Manager deals with the whole configuration of the game managing the changes in the graphical, audio and control configurations.

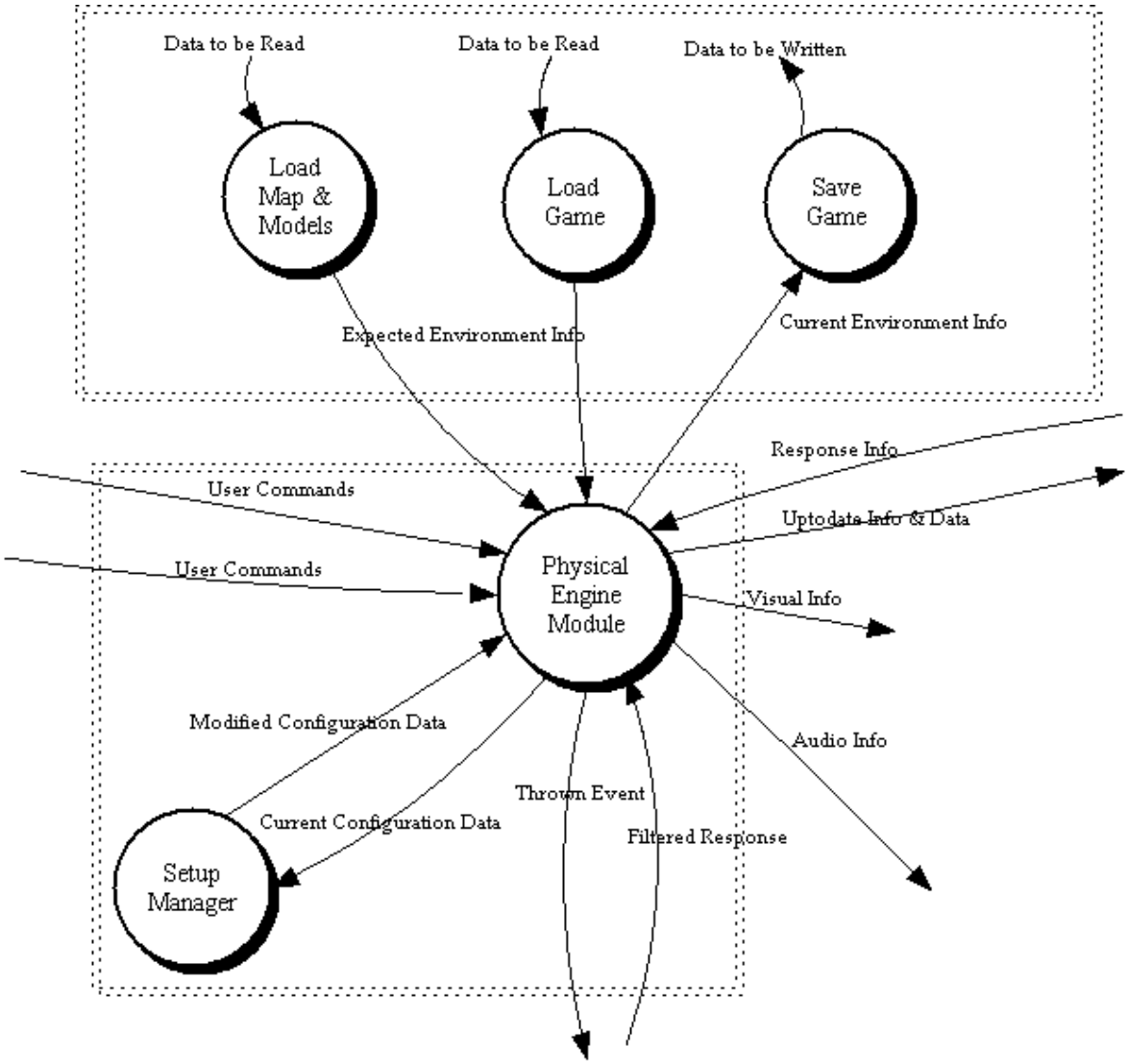

Figure 3.3 - DFD Level 2, Physical Engine

As a whole, these diagrams give an idea of the sub-structures of the whole system and the data flows in between them.

3.2. Behavioral Modeling

To model the reactions of the software to the external events, the State Transition Diagram (STD) is used. In this modeling technique, the STD "represents the behavior of a system by depicting its states and the events that cause the system to change the state. In addition, this STD indicates what actions (e.g. process activation) are taken as a consequence of a particular event." [2]

[2] Pressman, R.S., *Software Engineering – A Practitioner's Approach*, 5th Edition, p. 317

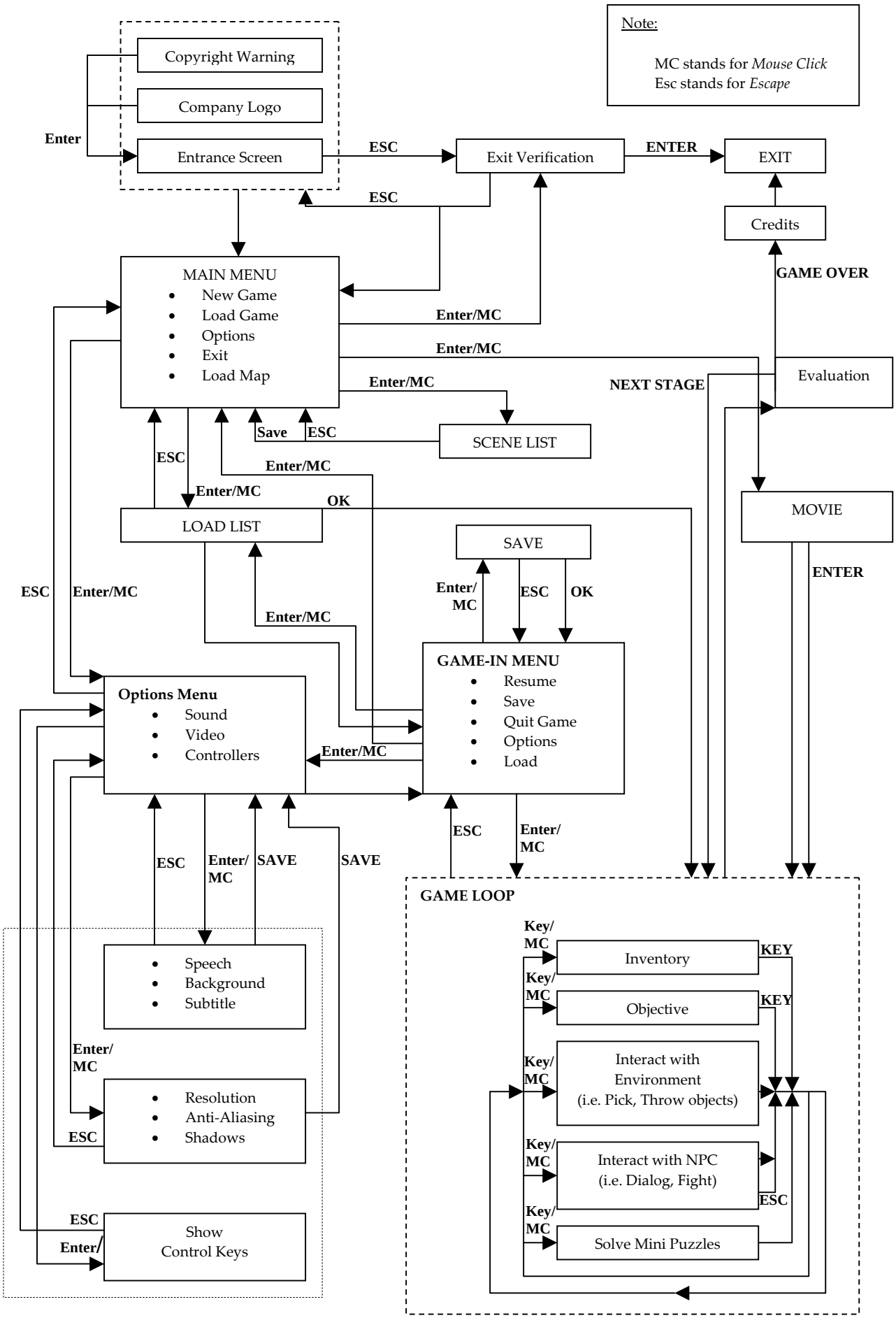

Figure 3.4 - State Transition Diagram

The STD of the software to be developed (See Figure 3.4) indicates the states and transitions among them. The menu interfaces and the game loop are the two main parts.

As can be seen in the diagram, Esc (Escape character) is generally used for return to the previous state. Enter and the other corresponding key characters are used for submitting and going one step further in the system flow.

To understand the game flow more extensively, it is very useful to examine the use cases within the game flow.

Possible use cases for the system activities follows:

1) Actor activates the *.EXE file of the game. At the start of opening sequence, copyright information and company logo/animation will be shown. During this demonstration, actor presses ENTER key on the keyboard. Sequence is cut and sequence jumps to main menu of the game.

2) Actor is at the main menu. Actor should select NEW GAME option, and following that action, game starts with the introductory movie/scenes. Introductory movie can be cut by ENTER key interrupt from actor side.

3) Actor is at the main menu. Actor should select LOAD GAME option, and following that action; actor is presented with a load list. Actor chooses a saved game if applicable and confirms. Actor starts from the loaded point in the game. If a load is not applicable, i.e. no saved game exists; actor should need to go back to main menu by choosing CANCEL option in the load list window.

4) Actor is at the main menu. Actor should select OPTIONS option, and following that action, actor is presented with OPTIONS MENU. Actor chooses SOUND option here. Actor turns on /off SPEECH, BACKGROUND sounds and/or SUBTITLES. Actor confirms the changes or cancels, and goes back to OPTIONS MENU. Actor goes up to main menu by pressing ESC.

5) Actor is at the main menu. Actor should select OPTIONS option, and following that action, actor is presented with OPTIONS MENU. Actor chooses VIDEO option here. Actor changes RESOLUTION here. Actor turns on /off ANTI-ALIASING and/or SHADOWING. Actor confirms the changes or cancels, and goes back to OPTIONS MENU. Actor goes up to main menu by pressing ESC.

6) Actor is at the main menu. Actor should select OPTIONS option, and following that action, actor is presented with OPTIONS MENU. Actor chooses CONTROLLER option here. Actor observes key functions in the game.

7) Actor is at the main menu. Actor should select Load Map option, and following that action; actor is presented with map options. Actor chooses desiring map, confirms, and goes back to main menu.

8) Actor is at the main menu. Actor should select EXIT option, and following that action; actor is presented with confirmation sub window. Actor confirms and exits, or cancels and goes back to main menu.

9) Actor is in the game. Actor presses ESC, and following that action, actor is presented with Game-in Menu. Actor chooses RESUME option, goes back to game.

10) Actor is in the game. Actor presses ESC, and following that action, actor is presented with Game-in Menu. Actor chooses SAVE option, and following that action, actor is asked for confirmation in case a free slot is available. If there is no remaining free slot for saving, actor is asked for overwrite on the least recently saved game. Confirmation grants the process in both cases.

11) Actor is in the game. Actor presses ESC, and following that action, actor is presented with Game-in Menu. Actor chooses QUIT GAME option, and following that action, actor is asked for confirmation. On confirmation, actor goes back to main menu. Current game is not saved.

12) Actor is in the game. Actor presses ESC, and following that action, actor is presented with Game-in Menu. Actor chooses OPTIONS MENU. From that point, cases (4-6) apply.

13) Actor is in the game. Actor presses ESC, and following that action, actor is presented with Game-in Menu. Actor chooses LOAD option. From that point, case 3 applies. Current game will not be saved.

14) Actor is in the game. Actor presses "a specialized key", and following that action, actor is presented with INVENTORY. In the inventory screen, which occupies some small area, there are two slot sets. One of these sets consists of 2 slots and the other one does 15 slots. In the 15-slot set, actor keeps items collected around. Actor may takes two items from the 15-slot set and put them into 2-slot set and combines them, if possible.

15) Actor is in the game. Actor presses "a specialized key", and following that action, actor is presented with INVENTORY. Actor may pick an item from its 15-slot set and use it by clicking at a location in the environment, if possible.

16) Actor is in the game. Actor presses "a specialized key", and following that action, actor is presented with Objectives Screen. Actor sees here the current objectives.

17) Actor is in the game. Actor completes a stage. Actor is presented with an Evaluation Screen. Actor presses ENTER and passes to the next stage in the game.

18) Actor is in the game. Actor passes to the next stage. Actor is presented with a stage-integrating movie/scene. Actor presses ENTER key movie/scene will be interrupted. Movie/scene moves on unaffected from the other keys.

19) Actor is in the game. Actor completes final stage. Actor is presented with Credits Information. Actor's key presses do not interrupt Credits flow. Credits play unaffected. When Credits play is over, actor goes to main menu.

20) Actor is in the game. Actor clicks on a NPC, and following that action camera view changes as taking the actor and the NPC in the same frame. Actor continues conversation by further mouse clicks. Actor may click on "Leave" button and escape conversation.

21) Actor is in the game. Actor clicks on a place in the screen. Actor moves to that point. Actor makes explanation about the place under concern if applicable.

22) Actor is in the game. Actor clicks on an object in the screen. Actor moves near the object. Actor makes explanation about the object if applicable. Actor makes a second click on the object. Inventory screen appears. Actor may pick the object or leave it.

23) Actor is in the game. Actor opens the inventory screen. Actor may get explanations about the objects in the inventory by left click on the inventory.

3.3. Structural Analysis

The data flow among the different modules of the system is displayed in the DFD. These flows are controlled by several mechanisms that assure the consistency and integrity of data flowing from one module to another. The data flows across the modules of the game to be developed are also controlled in some cases. These controls are demonstrated in the Control Flow Diagram (CFD) in Figure 3.5. This CFD is a dual of DFD level 1 in that the modules are represented in the same level of abstraction.

As seen in the diagram, there are three control structures. Firstly, there is a control in the flow of graphical data from physical engine module to graphical engine. This control corresponds the activation of some of the graphical features. For example, through the corresponding configuration menus provided to him/her, the player can optionally disable shadowing. The other control mechanism is the sound control. The flow of audio data is checked against the player's decision on whether to switch on or off the game sounds. Lastly, as stated earlier, saving a game is enabled in only certain phases of the game flow. For this reason, there must be a control point to check if the current phase allows a game save.

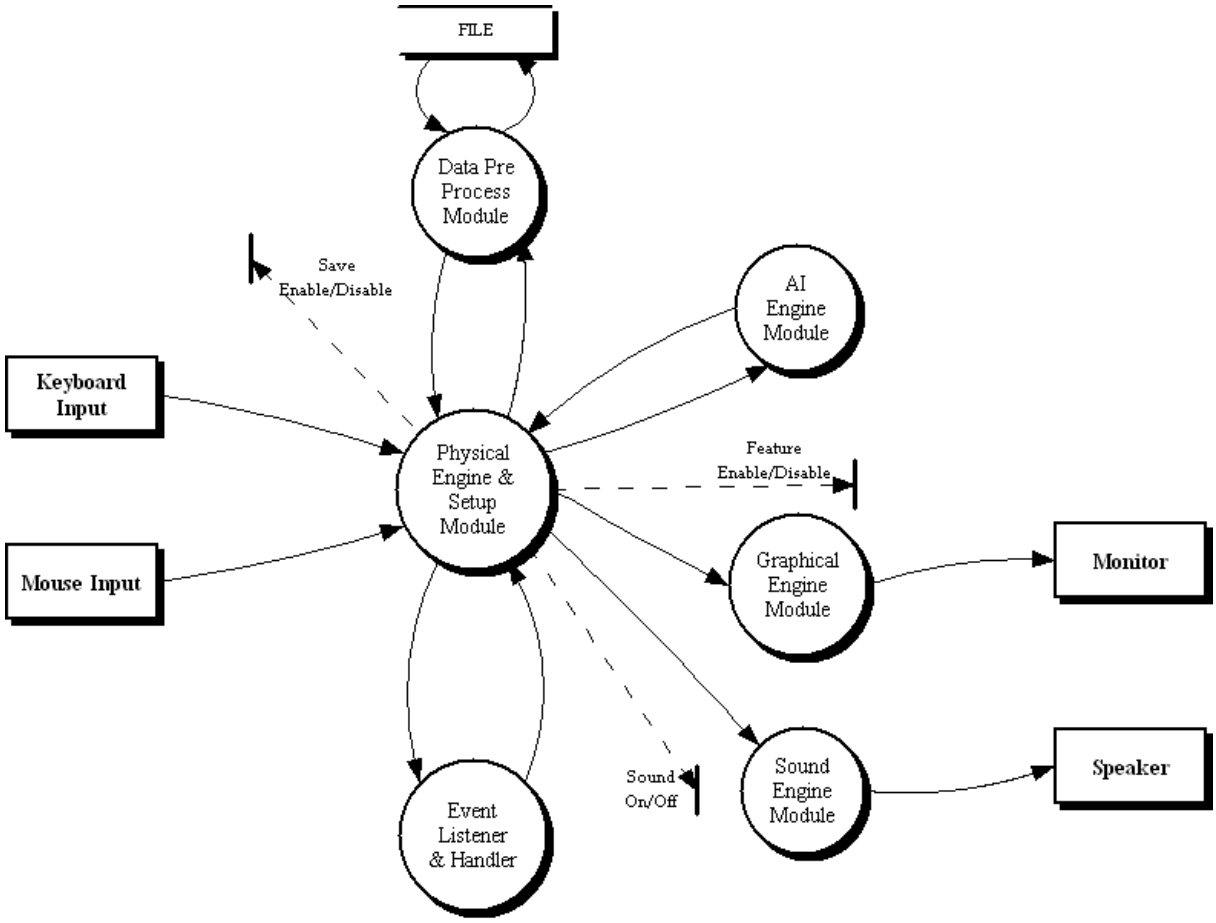

Figure 3.5 - CFD Level 1

4. GENERAL REQUIREMENTS

This part discusses the process requirements that must be met within the development progress of the software and the system requirements that determine the necessary environment for the software to be installed and run.

4.1. Process Requirements

This subsection involves both the description of the platforms on which the system is to be developed and a brief comparative analysis of the alternative platforms.

4.1.1. Software and Hardware Platforms

The software will be developed on Windows XP Operating System using C++. Microsoft Visual Studio will be the tool for compiling, executing and debugging the C++ codes. In order to create 3D graphical environments, OpenGL will be used. Besides, multimedia content will be created using auxiliary programs. As script languages, Python or VB are the most probable candidates for game scripting.

Software engineering tools (i.e. Microsoft Visio) will help us create design documents such as UML diagrams and time schedules. For the sound management, a sound editor, named GoldWave, is thought to be useful. The graphical tools (i.e. 3Dmax, Adobe Photoshop, etc.) to draw images are much probable to be used.

Pentium 4 computers are the hardware on which the system is going to be developed. Internet connection is also needed to perform necessary research activities.

4.1.2. Development Platform Analysis

Regarding the market needs, it is a very reasonable idea to develop the game for Windows environments since most of the end users who are potentially computer game players use Windows operating systems on their PCs. Therefore, developing the system on Windows is mandatory.

Deciding on the programming language, the performance of the software was the basic consideration. To provide the consistency of the hierarchical structures of data objects that base the software system and the modularity of the program, an object-oriented language was quite feasible. As an object-oriented language, Java seemed to be disadvantageous in that it makes the systems considerable slow due to the nature of its Virtual Machine (VM). Since it offers the powerful features of C and it facilitates object-oriented structures, C++ is the best choice to meet the performance and modularity requirements.

There were two alternatives for the development environment for C++ coding: Microsoft's Visual Studio and Microsoft's .NET platform. Visual Studio overweighed .NET in that .NET has extra features like garbage collection which is highly performance decreasing.

For managing the 3D graphical environment, it was agreed on OpenGL as the graphical library. The alternative was DirectX. The basic reason behind this choice was that the group members are familiar with OpenGL rather than DirectX. Also, the traceability of DirectX seemed more complex. Yet, it is a fact that to reach OpenGL tutorials on Internet is much more easier than to do those of DirectX. Even though OpenGL facilitates a powerful utility tool, GLUT, which provides built-in methods to create and animate graphical objects easily, we preferred not to use its benefits to obtain a purer code that is more desirable in that it is faster than implementing with ready-to-use methods.

4.2. System Requirements

To obtain a satisfactory game performance, the software should be installed on a PC with a certain configuration. These specifications are mostly satisfied by the current PCs in the market. Following is an example of such kind of configuration with minimum requirements.

Central Processor	:	Intel Pentium III
Graphic Adapter	:	32 MB with 3D accelerator and OpenGL support
Main Memory	:	64 MB
Sound Card	:	Any compatible type with an installed driver
Mouse	:	Any compatible type with an installed driver

5. PROJECT MANAGEMENT PLAN

In this project, the organization of group members can roughly be described as Democratic Decentralized (DD) since among the group member there is no one permanent leader and tasks are assigned for short durations and the task organization is not strictly described, rather, each member can be appointed to different functions. Decisions on problems and approach are made by group consensus. Communication among team members is horizontal.

The way that the group is organized has been preferred to be as explained above since the project has a limited development time and it can be considered to be a difficult project. The details of the project development cycle are explained in the following sub-sections in more detail.

5.1. Approach and Methodology

A Linear Sequential Model is being followed within the development life cycle of this software. This model "suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support." [3]

The definition of the problem scope has been followed by the phases of linear sequential model. As the first step, the requirement analysis has been achieved through the research and survey activities. The results of this phase are already being introduced with this piece of report. Afterwards software requirements will be analyzed and the design phase will begin in which the requirements are translated into models. Then a prototype will be developed. The prototype will help seeing the probable problems that can be encountered in the implementation phase. Since the prototype will have basic features of the system, there will be the chance to see the big picture of the project. Having completed the design phase and prototyping; implementation of the project will start. Finally an extensive testing policy will be followed on the system to make the system as much bug free as possible.

An object-oriented approach will be applied in the project development progress. The system is intended to have a high modularity that increases the adaptability and reusability whereas it decreases the complexity of the system. Also object-oriented programming provides several concepts and features to make the creation and use of objects easier and more flexible.

[3] Pressman, R.S., *Software Engineering – A Practitioner's Approach*, 5th Edition, p. 28

5.2. Major Milestones

The development phase of the adventure game has ten major milestones. During the development of project the following milestones are followed:

- Determination of Project Scope: The scope of the project is determined. The boundaries of the project are specified. The approach and methodology that will be followed are decided.
- Gathering Requirements: Examining the current adventure games and their features in the market, interviewing with the computer game players, requirements are gathered.
- Analysis of Requirements: Listing the advantages and disadvantages of the existing games, deciding on new features and analyzing the most efficient and useful components that can function in an ideal adventure game; the requirements are analyzed in this particular report.
- Architectural Design: The necessary data structures will be modeled. Data types and classes will be specified; the relationships and the hierarchy among them will be determined. These models will be used in the implementation phase for the game engine development.
- Multimedia Design and Production: In this stage, the 3D environments and objects will be modeled. Videos and cut scenes will be determined. Images constituting the texture will be produced. Also, audio components will be built.
- Developing Prototype: A prototype will be developed which demonstrates the main features of the system.
- Implementation: All coding activities, embedding and combining all previously produced multimedia and related components will be held in this stage.
- Testing: Both white box and black box testing will be applied. It is planned to spend much time on testing to deliver a more consistent product with as few bugs as possible.
- Documentation: User manuals will be produced.

5.3. Project Estimations

In the calculations of the estimations, the Basic COCOMO Model is applied for this project. It computes software development effort as a function of program size. The program size is measured with the number of lines of code to be written. The system to be developed is considered to be organic since it is not hardware dependent or managing hardware. Therefore, the formulas used in the computations of effort (E) and duration (D) estimations and the coefficients in the formulas are:

$E = a_b * (KLOC)^{b_b}$, $D = c_b * (E)^{d_b}$
$a_b = 2.4$, $b_b = 1.05$, $c_b = 2.5$, $d_b = 0.38$, KLOC(estimated Kilo lines of codes) = 15

The results of the calculations are as follows:

E = 41.22 person-month, D = 10.27 month

Since this is a 5-member group, the duration of the project is expected to be approximately 8 months.

Estimated Duration = Calculated Effort / Number of person => D = 41.22/5 = 8.244 months.

5.4. Quality Assurance, Risk Management and Testing

Developing a project as concerning project quality and continuity does stand as a undeniable property leading to the constructed projects perfection. According to that idea, though we are in the design phase of the project, we make analysis for what we can do in order to maximize the project that we plan to construct and publish in future. To aim at highest-level quality, we shall detect possible aspects which are potentially against product development and product itself, and avoid those aspects as we learned from the "Software Engineering" course. From that point, we can state future risk holding aspects and techniques to avoid them in such sentences.

One possible risk we may encounter is the end-user hardware system. The project does need to be working on different hardware systems. Predicting that hardware variability is a phase which is very hard to handle. For that reason, in order to keep the situation under control as much as possible from the developer's respect, end-user is presented with minimum requirements integrated with the product. In that way, it is planned for the customer gladness to be higher.

Moreover, while hardware system is supporting the minimum requirements, it may be the case that system does not support the OpenGL library made use of in the project. Such risks actually are adjusted by Graphical Adaptor providers to a minimal level. That kind of a risk does come from the lack of OpenGL driver support in the Graphics Cards of the systems. Fortunately, these days almost all cards do support corresponding driver.

One another risk and restriction is that the time for the project development is strictly restricted. It is calculated in the time estimation of the required time amount for the project to be completed. That estimated time is later than the project delivery date. Such time restrictions are meant that things are needed to be done more quickly. Quick implementation comes hand-to-hand with potential arise of bugs in the software.

Additionally, one another obstacle is lack of experience. Same as in the time limitation example, lack of experience also does comes with potential bugs in the software.

Arise of bugs is a risk which does shake the project quality reliance deeply. Arise of these unintended bugs and risk groups does seem inevitable. Bugs are needed to be taken care of, and the finest approach to such a point is testing.

Testing the software and in this way solution of such bad points is one of the project developments crucial steps. By testing, it is aimed that project comes to the point the end-user wishes to see it, and have the quality maximized. For the testing process, both white box and black box testing will be applied. White box testing will be done by group members. However, there will be other testers for black box testing.

5.5. Schedule

To set a time schedule for the project, those of previous year projects and the ones from Internet have been examined and a flexible time schedule, which may be subject to change time to time, has been prepared. The time schedule is attached to the end of this document.

END OF DOCUMENT