PAGODA SYSTEMS

ULApag

CENG 491

REQUIREMENT ANALYSIS REPORT

Ercan Üret --- 1298045
Çağatay Turkay --- 1298355
Selçuk Tunç --- 1298348
Sinan Mutlu --- 1298025

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

This document is a Requirement Analyses Documentation of the PAGODA group project. Document contains a detailed description of software including main functionalities, work load separation of program. All project requirements, specifications, decisions and estimations are included and explained in this document. The document is also strengthened by the grant chart, user case diagrams.

1.2 SYSTEM PURPOSE

System can be briefly explained as an education tool based on animations and visualizations constructed with 3D computer graphics in the area of assembly. System has been decided, designed and will be developed to be able to create new animations for users in order to educate their customers and show them tutorial like samples in an interactive 3D environment. Our research (see reference [2]) showed that system may be valuable for industrial areas like Lego toys, furniture assembly, etc. but apart from these sectors system will be capable of assembling any mechanical component.

The main purposes of the system, is to be able to visually assemble any mechanical system, make an animation of this assemble to educate people and finalize this animation for end users as a whole product which gives information, and abilities to users to travel in the animation.

One of our target for the system to become very easy both for developers, and their customers – end users - to understand and to use their own environments. And system should be modular to be available for developing, adding new features and patches in the future.

1.3 SYSTEM SCOPE

In the aim of a 3D animation oriented education tool, system has been designed with two main parts; design tool part which is our main program and product that our customers (developers/designers of animation for their system) will use and its output a run time environment (our developers animation product

which will probably be a complete CD product) where the developers' work will run.

Developers will just face with our design tool. Since developers and designers will meet with design tool environment, it is assumed that we have more experienced PC users who are also familiar with some other design tools like 3DS Max, Flash, etc. in this part. With this scope our design tool will use familiar notions like all other design tools do with the same assumption. Timeline usage, screen controls and some objects are designed with this approach. Timeline – frame relations which is a mile stone for animation, screen usage and controls which is a master construction for mechanical assembly, and objects and library – object relations to manage object display, property and creation are all our main functionality we focused on. These functions are explained in Project Specifications chapter in detail and also viewed as diagrams in Use Case chapter.

End users – customers of our customers – will just face with run time environment. So it is important to assume that they are users lack of moderate PC knowledge. In the run time environment end users will view the developers' animation with text and volume extra features and also user have the ability to travel in the animation environment. All functionality is explained in Project Specification chapter and also viewed as a diagram in Use Case chapter.

1.4 ACRONYMS, ABBREVETIONS

GUI : Graphical User Interface
3D : Three-Dimensional
3DS : 3D Studio
RTE : Run Time Environment
DFD : Data Flow Diagram
COCOMO : Constructive Cost Model
PC : Personal Computer
LOC : Lines of Code
FP : Functional Points

1.5 REFERENCES

[Pressman_01] Pressman, R.S., *Software Engineering A Practitioner's Approach,* 5th Edition pg. 128

[Pressman_02] Pressman, R.S., *Software Engineering A Practitioner's Approach,* 5th Edition pg. 129

[Pressman_03] Pressman, R.S., *Software Engineering A Practitioner's Approach,* 5th Edition pg. 133

[Pressman_04] Pressman, R.S., *Software Engineering A Practitioner's Approach,* 5th Edition pg. 311

[1] Pagoda Group, Project Proposal Report

[2] Pagoda Group, Literature Survey Report

[3] "Interactive 3D and Virtual Reality Software from EON Reality - Home Page" www.eonreality.com

2. RESEARCH

2.1 SIMILAR APPLICATIONS AND TOOLS

We have initially searched for similar programs. The best candidate which is similar to our design tool is a Lego Construction program called MLCAD. This program has a library of pre-drawn Lego parts and you can construct big Lego models with using these Lego parts. The tool has four sub windows each displaying the current scene from four different angles. The program doesn't supply any importing or drawing capability, it only recognizes files of a primitive, command line, Lego part drawing tool called Ldraw.

For our run-time environment, we investigated the products of a company called Eon Reality [3]. This company creates 3D advertising solutions for various kinds of companies. Their products only have the capability to play a pre-made animation. In all of their products, they have made a user interface with some

basic modification buttons like rotation, translation and zooming. In one of their product, they were giving the instructions of the assembly process of furniture. During this process, the user can rotate and zoom in/out to the animation. The user is able to watch the animation step-by-step or watch the whole animation from begin to the end. The user interface includes a supporting text box but there is no sound included. Their run-time-environment is made as a browser plug-in, and the animation is played in your browser. The player inspired us very much, and gave us clear ideas of how to implement the RTE.
The two most useful programs which helped us to develop our design are; Macromedia's Flash MX and Discrete's 3D Studio Max. First we cover Flash and then 3DS Max.

Flash is a very successful animation and application development environment which is used worldwide and being more popular everyday. The most useful part of this program in our design is how it handles animation creation process. Flash has a very easy-to-use timeline features. The frame types are set in a very successful and powerful way. Also it provides tweens which makes it very easy to create animations. We will pay great attention to implement these tween features and more in our own program. Also Flash has a well-known player called Macromedia Flash Player (which is a plug-in for browsers), and this player gives us great idea of how a run-time-environment can be. Flash is our main example when designing our project. We are naming our project as "a new Flash MX which works in 3D". But we are thinking to have a limited number of Flash's features of course. Flash's power mostly comes from its very successful scripting language (which allows you to create any kind of effect) and its vector based drawing mechanism (which reduces space drastically). And we can not have these features at this stage.

Our last example program is 3DS Max. It is a very successful and powerful 3D modeling tool used by professionals. We have concentrated mainly on its 3D drawing and rendering capabilities. We have examined how it displays objects on the scene and how it handles modifications. 3DS Max have been a great example for us to design the flow of the program. We also examined the timeline usage of 3DS Max, but as Flash is more powerful in that field, we only got some ideas from 3DS Max.

2.2 TECHNICAL RESEARCH

We had some technical issues to set clear before beginning the requirement analysis. The first problem was to find a graphics library which will provide us more than OpenGL does. We first decided to use Irrlicht Engine which is a free and wide graphics library. But our assistant advised us to use OpenSceneGraph which provides more features, especially a class to import 3DS Max files into our desired format. After realizing that this library will be very applicable for our needs, we decided to use C++ as the programming language and OpenSceneGraph as our graphics Engine. Later on we have decided to use .NET as our development environment because, first of all it is a very successful IDE with VC++ 7.0 integrated into it. Also it provides us easy and a great number of tools to build a user-interface. One other library we will need is related with sound handling. We searched for available libraries and found fmod which is widely used in many of the computer games which can play mp3 and waw files. It also handles sound in a 3D manner and your position effects sound attributes. Lastly we decided to import our objects from 3Ds Max. Firstly, it is very successful and widely-used and secondly, and more importantly, OpenSceneGraph can read .3ds files.

3. REQUIREMENTS

3.1 HARDWIRE REQUIREMENTS

3.1.1 USER SIDE

- Pentium IV-1.4 GHz / AMD 1.5 GHz Processor
- 128 MB of Memory
- 32 MB of Video Card (Display Adaptor) with 3D Support
- Free Disk Space for Installation
- Sound Card

3.1.2 DEVELOPER SIDE

- Pentium IV-2.0 GHz / AMD 2.0 GHz Processor

- 256 MB of Memory
- 64 MB of Video Card (Display Adaptor) with 3D Support
- Free Disk Space Depending on the Needs of the Project
- Sound Card
- Internet Connection

3.2 SOFTWARE REQUIREMENTS

3.2.1 USER SIDE

- Windows 98-2000-XP Operating Systems (for both end users and animation developers)
- 3D Modeling Tool (for the animation developers and our software will import files from 3d Max.)

3.2.2 DEVELOPER SIDE

- fmod library

  It has a library that can be used with C++. In addition we will use it to play sound that tells to the end user the work being done at that time.
- Open Scene Graph

Based around the concept of a Scene Graph, it provides an object oriented framework on top of OpenGL freeing the developer from implementing and optimizing low level graphics calls, and provides many additional utilities for rapid development of graphics applications.

- 3DS Max
- Windows XP Operating System
- C++ Compiler (Microsoft Visual Studio .NET)

4. PROJECT SPECIFICATIONS

Our project has two main parts: the design tool and the run-time-environment. The design tool environment is the major part of the program which is used by the assembly animation developer. And we have a run-time-environment which is capable of playing our design tool's output and it is used by the end user. We will cover these in subheadings and in detail in this section.

4.1 THE DESIGN TOOL

This is the main part of our program which will be used by the animation developer. It will enhance the developer with various tools to develop an animation sequence easily. We can inspect the program in a number of components and by investigating these components separately; we will give a clear idea of how the program will work and which functionalities it supplies. Here are the components of the main design tool:

4.1.1 THE SCENE AND GRAPHIC EDITING

The scene will form the main window which the user will use generally. The scene will be partitioned into four. These partitions will be four different camera views, like in most of the 3D modeling tools. These camera views will be top, left, right and a perspective camera which will be used as a free camera. Only in the perspective camera window, which we will call the main sub-window later in this section, the objects will be drawn filled. In the other three windows the objects will be in wireframe mode. All of these scenes will have a grid map drawn on them. This grid map will be modifiable, i.e. the user can choose the grid size. The user will be able to rotate the main map and zoom in/out to it. When the map is rotated, all the objects placed on it are rotated as well. When it is zoomed in, we are zooming all the available objects.

The program will have some states which the program is in and the user inputs are handled with respect to the mode selection. The selection will be made with buttons on the main toolbar. The buttons will remain pressed in order to acknowledge the user about the current program state. The available graphical editing modes are briefly as follows:

- Add Mode: When the user wants to add a new object, this mode will be chosen.
- Select Mode: The user will be able to select one or more objects at a time. It will have different modes which enhances the selecting process.
- Select & Move Mode: The user will first select the object or objects and with the aid of the helping arrows which will show the x-y-z directions to the user as an aid in moving the objects.

- Select & Rotate Mode: The user will first select the objects and will be able to rotate it afterwards with the aid of helping lines which will appear on the object to indicate x-y-z directions and the user will rotate by clicking on the helper lines and moving the mouse up–down or right-left.
- Select & Scale Mode: The user will be able to scale the object in this mode. Helper scale lines will appear on the object and the user will scale the object with values higher than one by moving the mouse-up movement (with left mouse button clicked) and scale with values less than one with mouse-down movement.
- Make Composite – Remove Composition: The user can bind one or more objects to each other. From now on they will be treated as a composite sole object. To remove this composition there will be a remove composition mode. The composition capability is quite crucial, because, as the program will be mainly used for assembly instructions, already assembled parts will have to move together.

To elaborate the usage of the graphical editor, we go into detail of functionalities that it supplies. First we will cover how to create a new object. The user will first select the add–mode. Afterwards he/she will choose an object from the library (which we will cover in detail later in this section) and put it in a desired position on the scene with just clicking at a spot on the main sub-window. The other three sub-windows will be displaying the objects from different, predefined angles. After placing the object on the scene, the user can modify it. In order to modify a specific object, the user has to select the object. The available selection methods will be:

-Clicking on the object. If the user wants to select multiple objects, selecting more objects with the button configuration CTRL + left click will be available.

- Using area selection tools. This tool will be similar to area selection tools in all common modeling and drawing programs. It will select all the objects which lie inside the selection area the user defines. It will have two modes the rectangular selection tool and the very useful polygon lasso tool which will let the

user to select in any shape desired. In this lasso tool, the user will draw a generic, connected shape which will be the selection area.

After making the selection the user has a number of choices to apply on the selected object. These modifications are all done in one of the modes considered above. The first modification is translation. When the user selects the move mode, helper movement directions will appear. The user translates the objects with simple mouse dragging. If more than one object is selected they will move altogether. If one of the composite objects declared by the user is selected, the ones connected to that object are also selected and the translation will be applied to all of these composite objects. The other modifications; namely, rotation and scaling are applied in the same manner, there will also be helper lines and the modifications will behave the composite objects in the same manner. If the user enters one of the move, rotate and scale modes without selecting any object, he/she will be able to first select and then do the modifications.

One other capability will be making composite objects. The user will first choose the make composite mode and afterwards select one or more objects and after the selection is over, some kind of apply composition command will be passed and now the selected objects are a whole. If later on, the user wants to add more objects to this composition, again composite mode is selected afterwards one of the objects which is already a member of the composition (which means selecting all the elements of the composition) is selected and then the new objects to join the composition are selected. If the composition is planned to be torn into its pieces, the remove composition mode is selected and then the objects that will be removed from the composition are selected. If all the objects of the composition are selected (with the rectangular selection tool for example) there will be no two objects left connected.

One last manipulation to be done on an object is deletion. When an object is selected, a stroke on the DELETE key will be enough to delete the object.

Lastly, the user will be able to add lights to the current scene. As light is very important on the appearance of the models, we will pay great importance on lighting quality and its usage. There will be two lighting modes available: the spot light and the skylight. The spot light will give a more direct light whereas the skylight will provide a smoother and wide – spread lighting. These lighting modes

can be used together on the same scene. The user can create more than one spot light and only one skylight. These lights will have their own attributes like; light color, light intensity and light radius. The lights will be treated same as the objects, i.e. they can be translated, rotated, animated and deleted.

4.1.2 LIBRARY

The library will be the main tool for the user to manage all the objects created on the scene. There will be two internal parts of the library: Current Library and Shared Library.

Current Library will hold information about all objects created on the current scene. When a new file is opened to work on, the library will be empty as there is no object crated on the scene. To add an object in the current library the user will either import from a 3D Studio Max file or add an already imported object from the shared library. There will be an import button on the library panel which opens up a file menu to locate the file to be imported. The imported file will appear on the library and the user can assign a name to it. Importing an object to the library doesn't mean creating an instance of it on the scene. The user will have to first enter the add-mode and a select an object from the library and afterwards click on the main display window to create an instance of the object. The newly created instance of the object will be displayed in the library under the objects name, the library will look in a tree structure fashion. We can see the imported object as a class and the objects on the scene as instances of that class. The instances of the original object will each have their own attributes, namely; name, x – y – z coordinates, rotation amount and composition properties. Another way to select an object on the screen will be selecting it from the library. Also another way to delete an object from the current scene is deleting it from the library. Deleting an imported object (i.e. a class) will also delete all the instances of it.

The Shared Library is an extra feature for the library. It will include very frequently used objects that have been already imported from a .3ds file into our internal representation (their extensions will be .ulp, we can use xml instead). The aim of this feature is to remove the cost of the very expensive importing process. For example, the user has modeled screws in a modeling tool and imported them into the current file's library. If he/she is thinking to use it again in another

assembly animation project, the screws can be added to the shared library and they will now be available for all the newly created projects. When the user wants to add an object from the shared library to the current library, he/she will simply drag and drop the chosen object from the shared library to the current library without waiting for the object to be imported again. Also the shared library will be a historical database for the user. If the user wants, the shared library can be carried to another computer without any modeling tool and our program can work properly on that machine without needing any extra programs. When the user wants to import a file to the current library he/she will be asked whether to include it into the shared library. If the user decides to add an object to the shared library afterwards, he/she can add with just dragging it from the current library to the shared one.
Lastly, the library will have a small display on top of it, which shows a small image of the object selected in the library.

4.1.3 IMPORTING & EXPORTING

Our program will not have any drawing or modeling ability. There are very complicated and successful modeling tools available for professional usage. As we are planning on an easier and amateur usage, we will not deal with the modeling part. Instead we will enable the user to import from the modeling tool's files into our file format. Our object file format will be .ulp and it will contain all the necessary information about the object modeled in the modeling tool. It will mainly contain; vertex coordinates; texture coordinates; if there is a map applied on the object, all the essential data about it like normal vectors and map files. You will get what you model in the modeling tool. We are now only thinking of importing from Discrete 3D Studio Max. The reason for this is 3DS Max is very popular in current industry and also OpenSceneGraph provides tools to import 3DS Max files.

After the user is pleased with the animation and wants to finish the current project, the last step to take will be exporting the project into our output file format (called .pgd). This file will be played by our run-time-environment. The format of this file is not clear at the moment but the capabilities the run-time-environment supplies are clear and will be covered in the upcoming sections. But

to mention one thing about the output file, the user will be able create a skin for the run-time-environment, i.e. the user will be able to add a picture, probably the logo of the company, and determine the color of the user interface of the run - time – environment.

4.1.4 ANIMATION CREATION PROCESS

The animation creation process is the most critical part of our program. As the user's main aim to use this program will be making assembly tutorial animations, this feature of our project must very easy to use and its tools must be very satisfying. So we are paying this part great attention. The most critical part in this process is the timeline feature and its usage. We are now covering it in detail:

4.1.4.1 TIMELINE

The animation is all about timing and the best way to control time in a editor program is using a timeline. Timeline is a structure which consists of little rectangles each of which representing a frame in the current animation. By making suitable changes in shape, location and rotation of objects in consecutive frames, the user can create successful animations. There will be a header which indicates the current position on the timeline. The user can advance the header to the desired frame by selecting that frame on the timeline.

4.1.4.2 FRAMES

There will be three types of frames on the timeline. These are:

- Key Frames: Key frames are the breakpoint frames of the animation. The animations take place between two key frames. The user sets a frame as key frame and places the objects to some position in the main scene, after that, user defines another key frame and move, scale or rotate the objects. The animation is created via some tweens between these two key frames. A tween is; making an average behavior with considering the start and end values. Tweens are explained in more detail later in this section.

- Normal Frames: Normal frames are the standard frames which lay between key frames. They are generally used in tweens between key frames.
- Step Frames: Step frames are special frames to be used in the run – time – environment. The RTE will provide the user to investigate the assembly process in step-by-step manner and somehow the RTE must know where a step in the output animation begins and where it ends. A frame can both be a key frame and a step frame at the same time.

4.1.4.3 TIMELINE USAGE

The next thing to consider here is how the tweens work. Tweens are defined between two key frames. Consider you have a first key frame and an object is placed on the current frame of the scene. Then you add some (let's say five) normal frames and after that five frames, you add another key frame. In this last key frame, an attribute of the object in your scene is changed, namely; it is moved to another position. If you apply motion tween between these two key frames, the object will follow a path from its initial position in the first frame into the position in the last key frame. In each of the five normal frames between the key frames, the object advances one fifth of the path which yields to its final position. Our available tween modes will be; motion tween, rotation tween and scale tween. Each of these three tweens are applied in the same fashion as the above tween explained. Motion tween will create a motion animation for the objects, the rotation tween will create a rotation animation, i.e. the object will rotate in partial angles to reach a final rotation angle, and lastly the scale tween, which simulates the scaling of an object in specified steps.

To extend the usage of tweens, we will define light sources as objects and tweens can be applied to them as well, this will enable the user to animate the motion of light sources and create dynamic lighting effects.

One last effect that can be created using this tween mechanism is the camera motion. Our main sub window, which is the one with the free perspective camera, will be our main source for the animations. We can think ourselves holding the camera in our hands and looking through it to the scene. When we rotate the map or zoom in/out to the perspective map, it is moving the camera or adjusting

the lens of the camera. If our viewing angle changes (means that we have rotated the map) or if our field of view (means that we have zoomed in/out to the map) has changed between two key frames, it means that we have adjusted our camera and the program will reflect this change in the camera attributes as a smooth sequence and this creates a pan effect in the animation and creates a cinematic approach. The smooth sequence of camera movement and adjustment is achieved by the same tween mechanism described above. Because the camera is also treated as an object here.

One other facility of the tween mechanism is that they can be applied altogether. If at the same time you rotated, scaled and moved an object and you also moved the camera and even the light, all these tweens will be applied simultaneously to the objects. The user can choose to cancel some of the twins between some key frames and these changes are simulated.

The user will be able to set a frame per second (fps) value which determines how fast the animation will be displayed. We will set a limit of 25, because higher frame rates will be hard for our graphic engine to draw and higher frame rates are not that essential.

Working with the timeline will be quite easy. The user begins with a clear timeline first with all empty frames, the output animation at this state will be of length zero. The user first begins with adding an object to the scene and now the current frame is a non-empty normal frame. If the user wants to create an animation with tweens, he/she adds another key frame after a number of frames later. The content of the first key frame is copied to the newly created key frame and all the normal frames between these two key frames have the same content as the first key frame. Any change in the first key frame is reflected to all the frames up to the second key frame. And then the user modifies the content in the second key frame and indicates that he/she wants some kind of tween between these two key frames and all these normal frames display the parts of the animation between these two frames. The content of these normal frames; which some tween operations are applied; are not modifiable, because their content only depends on the change of the object attributes between the key frames they lie between.

As in all assembly instructions, the assembly animations here can be considered to flow step-by-step. The user will want to categorize these steps and

form the animation with that knowledge. The run-time-environment needs to know where each step begins and ends. So the users have to set some frames as ‘Step Frames’. The first frame is always a step frame because it is always the beginning of the first step of an assembly animation.

4.1.4.4 ADDITIONAL FRAME ATTRIBUTES

The frames can also hold additional attributes like text and sound. The user will be able to add any supporting sound and text to the key frames. When clicked on a frame a dialog box will appear indicating the attributes of the frame. And if it is a key frame, there will also be fields for attaching sound and text to the frame. When the user attaches sound to a frame, when the animation is playing, the attached sound (this can possibly be a speech or some sound effect like screwing or knocking) is played at the time the specified frame is being played. The user will be able to load .mp3 and .waw files and can choose the sound play in a loop or for only once. The loop playing routine will be applicable for attaching background sounds. Also there will be a volume choice for the sound being loaded. The background volume can be set to a lower decibel not to disturb the instruction speeches. In addition to the helper sounds, the user may want to add acknowledgement texts to the animation. The user can enter the text to be displayed in the text box which will be found in frame’s attribute window. This text will appear in a separate part in the user–interface of the run-time-environment. Here the user will be given font and color choices for the text.

4.1.4.5 QUICK VIEW

The user will always be dealing with animation and he/she will want to see how the process is going on. One way to see it will be exporting and finalizing it and then viewing it from the RTE, but this very expensive in terms of time and performance. Instead we will add a quick view facility to see how the process is going on; it will have play, stop and pause buttons. It can also be used to traverse between step frames. The play operation ends when the full frames are over. The speed at which the frames will flow is determined by the fps value which is already determined by the user. In this quick view mode, the objects are not rendered and

they animate with the look they have on the drawing window. The static three window views will animate with wire framed models.

4.1.4.6 SHORTCUTS

All of the successful modeling and drawing tools employ lots of keyboard shortcuts for all kinds of operations. Professional users do not want to use the mouse because it makes them slower. Therefore, we decided to have all sorts of shortcuts to make usage easier and faster.

4.2 RUN TIME ENVIRONMENT

Run Time Environment is dedicated to end users who, as previously mentioned, are assumed to be users lack of moderate PC knowledge. Due to users' any possible level of knowledge RTE is decided and designed in a very simple logic that constitutes of a screen which 3D animations of assembly visualized, a text area that developer may want to view some instructions to users in each step, a volume on/off control and control buttons which enables user to travel through animation in step-by-step mode.

Apart from its simple logic design, our run time environment will run in two modes namely; assembly mode and step by step mode.

Since RTE is an output of our design tool, it is decided as a complete product with its software files (dlls, executable files, etc...) and animation file. At each animation, a new animation file will be created but all other software files will remain same as a predetermined format.

3D Screen

3D Screen is a 3D environment designed scene. Whole animation of developers will flow from this scene like a film according to their determined frame per seconds in assembly mode. And in step-by-step mode user has the right to stop animation at any frame and can view from any dimension she wants with help of control buttons.

Text Area

Texts will appear in this area according to what seems in the scene at that time. As men above texts are fixed to key frames. User has no right to intervene to this area.

Volume On/Off

The RTE has the capability to manage an audio file and gives the user to make on or off this file according to her will, and no matter with the audio file is fixed the whole animation or fixed to key frames.

Control Buttons

There will be Zoom In, Zoom Out, Rotate and Translate buttons on the RTE. This buttons can be just used in step-by-step mode which enables the user to travel around the 3D environment and get view of any aspects.

Assembly Mode

Assembly Mode is a mode for RTE works in a way of animation flow which user just watches the developers' preparation and can not intervene the scene or mode, control buttons do not work. In this mode user can control volume properties. Text area flows continuously. It is like a cinema watch for user which she has no interactive rights on the screen even if she wants to travel around the scene or get a view from a different angle. If she wants to some rights on the scene she has to select step-by-step mode for a intervention.

Step-By-Step Mode

In step-by-step mode, user has the rights for inspecting the assembly animations step-by-step. At this mode user can intervene to screen, control buttons are enabled, and user can rotate, translate the scene, and zoom in/out any object she wants. This is a mode of which on can say interactive. And also volume controls are available for user. Text area is fixed to the current step.

5. USE CASE ANALYSES

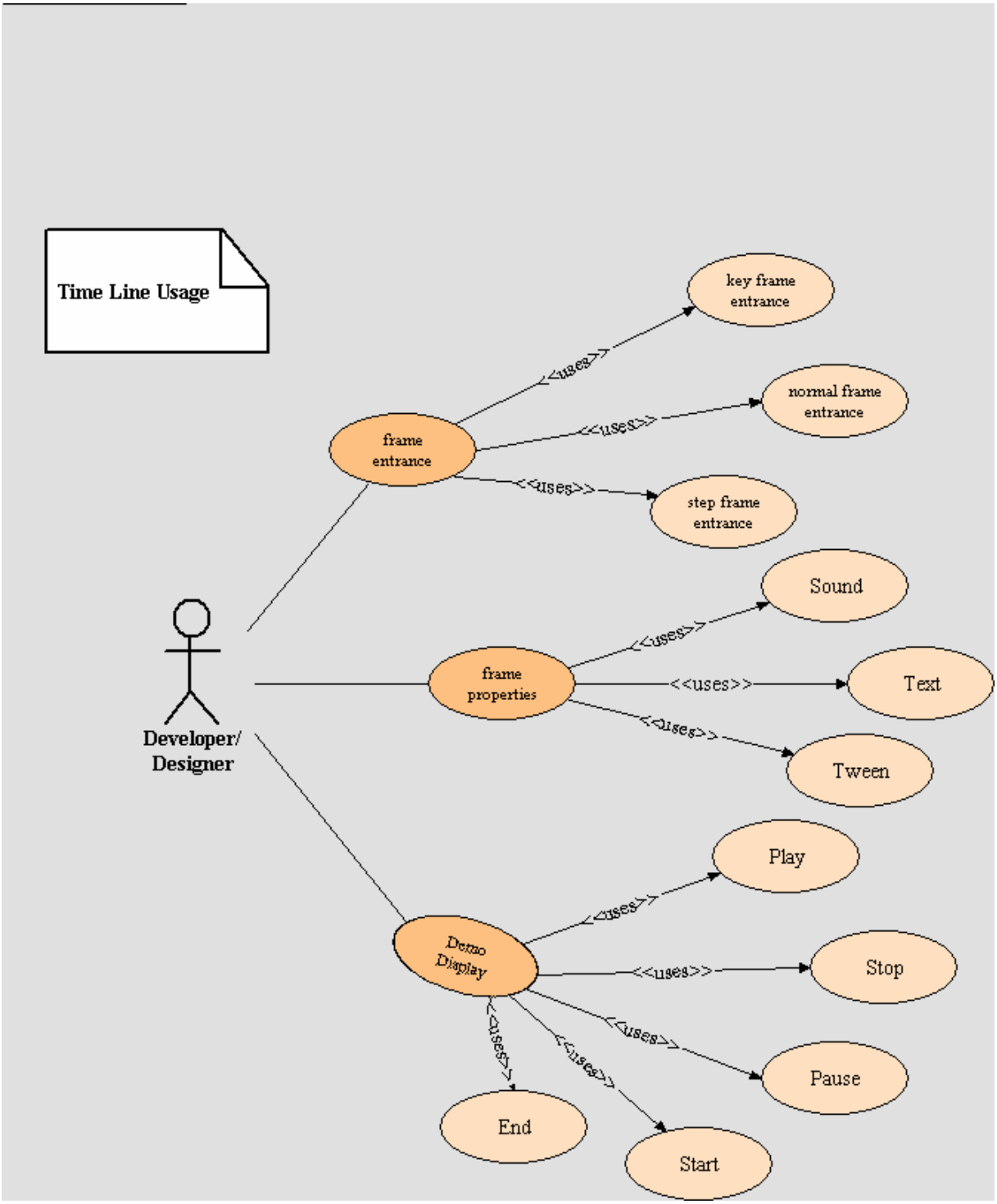
For our two main parts of program there exist two main users namely; designer/developer of the assembly animation who works with the design tool part of our program, and the end user who views and travels in the 3D animation environment in a run-time environment that is an output of design tool. It is possible to group functionalities of the program which users would expectedly face with.

Developer of a 3D animation will meet three fatal functionalities in the design tool; Time Line, Screen, Library each of which is exhibited as diagrams below.

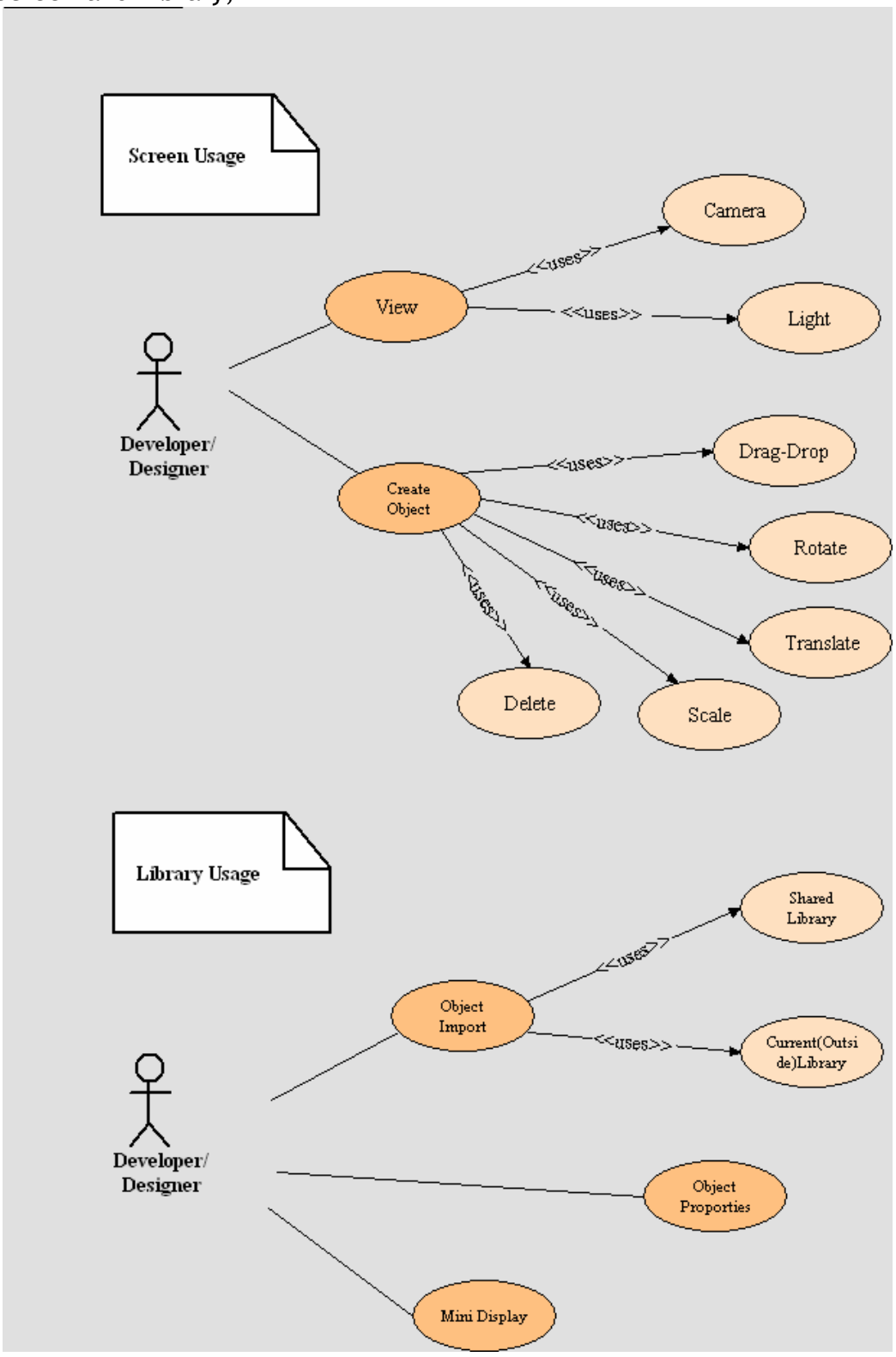
End user, the user of run-time environment, will face with three functionalities which will we held as only one diagram below.

5.1 DEVELOPER/DESIGNER

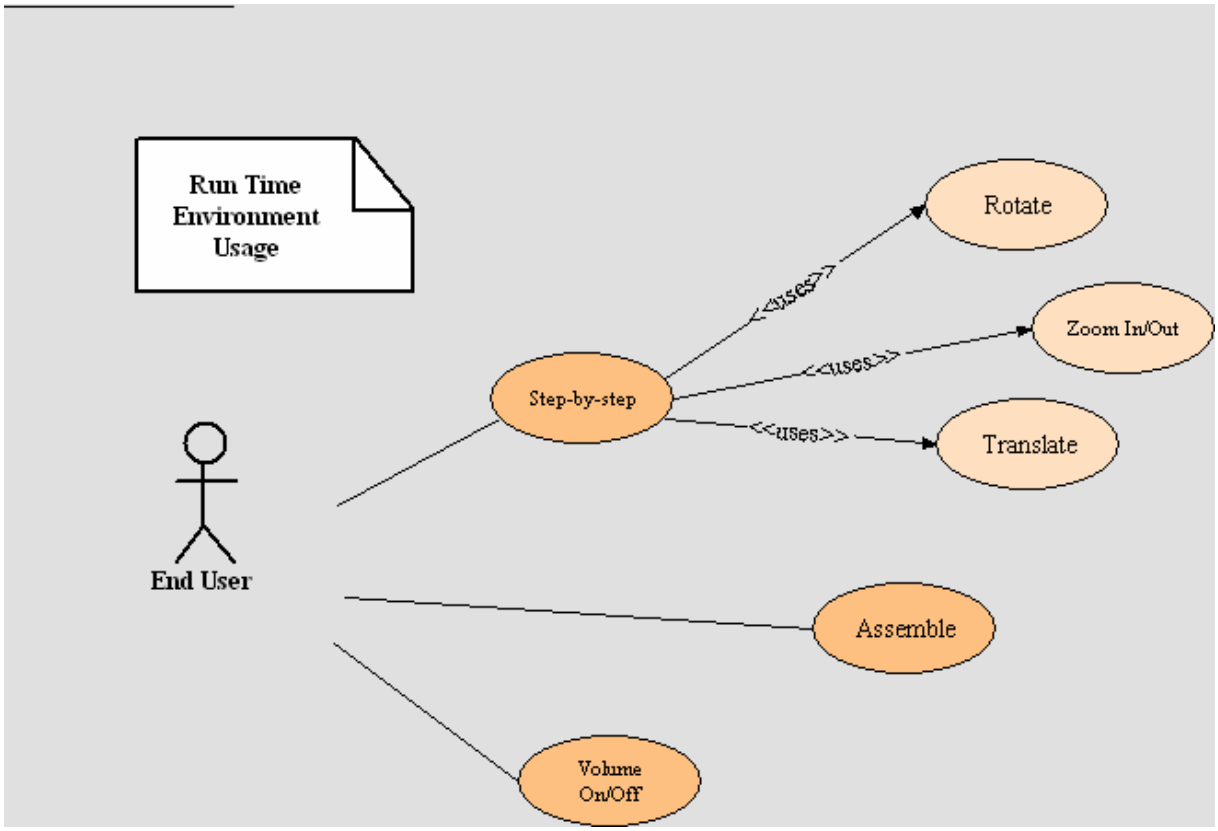
Timeline Usage;



Screen and Library;

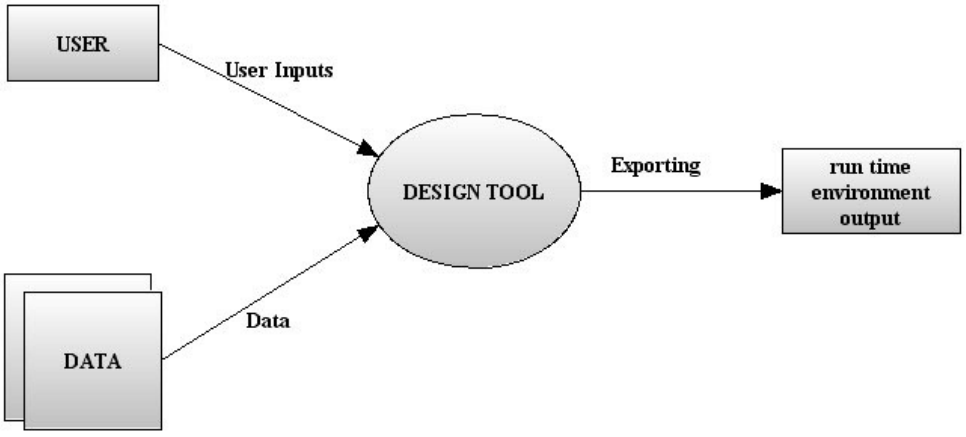


5.2 END USER

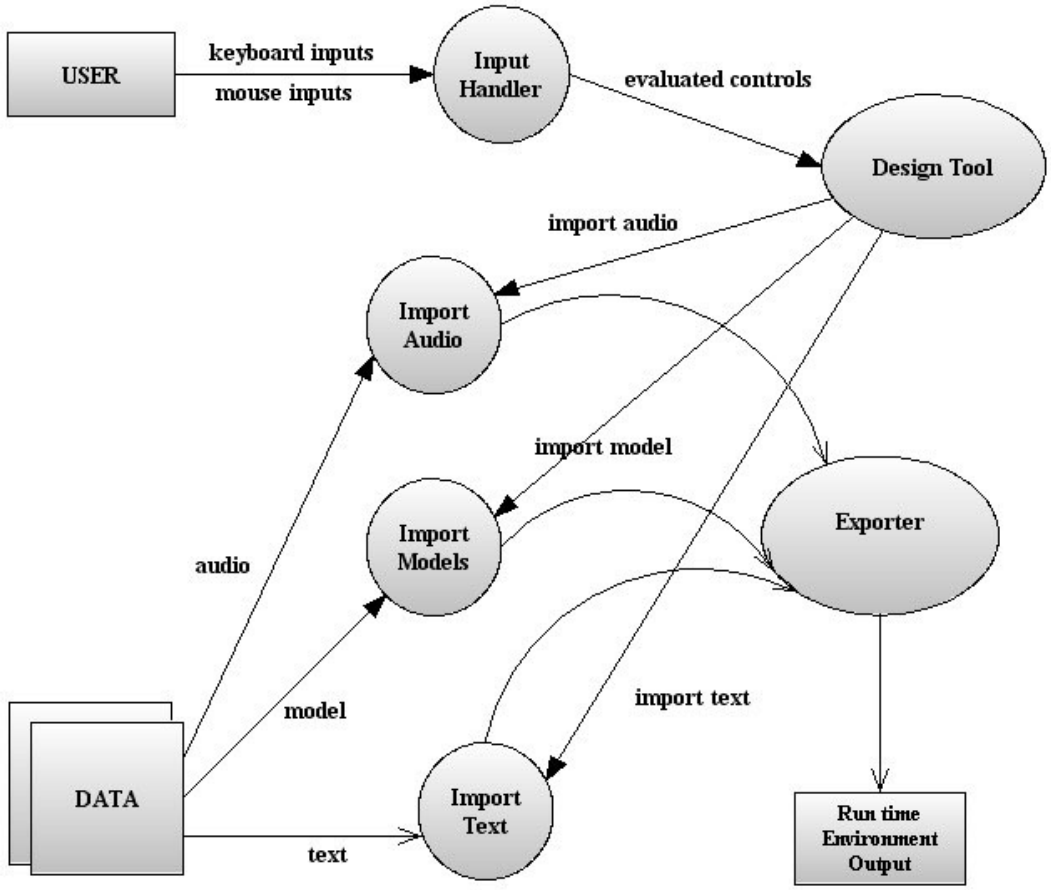

Run Time
Environment
Usage
Rotate
<<uses>>
Zoom In/Out
<<uses>>
Step-by-step
<<uses>>
Translate
End User
Assemble
Volume
On/Off

6. DATA FLOW DIAGRAMS

6.1 LEVEL -0 DFD



6.2 LEVEL-1 DFD



7. RISK MANAGEMENT

7.1 SCOPE AND INTENT OF RISK MANAGEMENT ACT

We want the software that we will develop to be free of any errors or defect. We are aware of that it is impossible not to face with any problems or defects in this project. These problems may cause latency for the completing the project. To be safe a risk management plan should be formed to develop a strategy to deal with any risk that can be encountered at any stage of the project. So with the risk management plan process of the project may be impact less and we can reduce the rate of the defects and errors.

7.2 RISK MANAGEMENT ORGANIZATION ROLE

Everyone associated with the software has the responsibility of managing the risk. If all the members of project pay close attention to all the details of the project from the beginning of the project, many risks can be avoided. By checking the project schedule, implementation size and required cost the project group can avoid risks. In addition by getting all the details of the equipment that are provided or accessible, the project team can avoid the risks. The negotiations with the customer can help to avoid the risks if the customer clearly states all the information about the project when the project development started.

7.3 DESCRIPTION OF THE RISK

In this part includes the descriptions of the risks that may be encountered during the development of the project.

7.3.1 PRODUCT SIZE

It includes the risks that are associated with overall size of the software to be built. There is a time limitation so if the project includes too much details and includes extra options with the necessary ones, so this may cause not finishing some parts of the project.

7.3.2 CUSTOMER RISK

This is the risk about the customers' motivation and willingness in helping the software development team. But during the development we will not have any customers so we will not have a risk like this.

7.3.3 DEVELOPMENT RISKS

It includes the risks associated with the availability and quality of the tools to be used to build the product. In the early phase of project if the team has not done enough research on the tools which are chosen may cause some difficulties or in the case if they have chosen tools that they are not enough experienced will cause huge problems.

7.3.4 EMPLOYEE RISKS

This is totally depends on the experience, willingness and ability of the team members to develop the project. If the members of the group are not experienced enough to use the tools or applications, which were chosen for developing the project, this may cause latency in completing the project. In addition if the members are not put all their efforts on the project this may cause fail of the project.

7.3.5 PROCESS RISK

Process risk involves risks regarding product quality. If the developed product does not meet the standards set by the project team, it is a failure. This may be happen because of the failure of the team to understand or describe the project. It also includes misjudgment needs of the project like total employees and proper equipments.

7.3.6 BUSINESS IMPACT RISK

This is the risk where concern is that of the not being able to come up or produce he product that has impact on the client's business. Today there are more preferable programs, developed by big companies, doing the same. So some clients may not find this software preferable. Moreover this program will not be too big but it will support all the needs of the end users. In addition it may not support all the needs of users who are developing animations since today there are programs that are doing more complicated works then this one. So it may achieve it goal by supporting end users needs, and also supports the main functions (which are sufficient for creating animation for end users) for the developer users.

7.3.7 TECHNOLOGY RISK

This includes the technology (that we used in developing the project) that is already or in future will be useless. If there is a kind problem in a short period during the developing the project it will be too hard to go on. But the tools and technology we will use will not be useless in a short period.

7.3.8 RISK TABLE

Category	Risks	Probability	Impact
Product Size	Misjudgment of size estimate	%35	3
Customer Risks	No Risk	%0	-
Development Risk	Insufficient resources	%20	1
Employee Risk	Lack of experience and willingness	%40	1
Process Risk	Low product quality	%25	2
Business Impact Risk	Not having any impact on clients	%25	3
Technology Risk	Changes in technology	%10	4

Values	Description
1	Catastrophic
2	Critical
3	Marginal
4	Negligible

8. PROJECT SCHEDULE

8.1 TIME & EFFORT ESTIMATION

8.1.1 FUNCTION POINT ESTIMATION

	count	weight	FP COUNT
Number of inputs	10	4	40
Number of outputs	2	10	20
Number of inquiries	4	7	28
Number of files	5	5	25
Number of external interfaces	0	0	0
Count Total			113

The total count of Fi's ,(i=1 to 14) “complexity adjustment values” that based on responses, is calculated as 40.
FPest= count total * [0.65 + 0.01 * ∑(Fi)]
FPest= 113 * (0.65 + 0.01 * 40) = 118,65

LOC/Function Point is found as 130 . Then;
LOC= 113*130 = 14690By considering the below equation and substituting the found value;

Effort = 1.4 * thousands-of-lines-of-code
Effort = 1.4 * (14690/1000). 20 person months
The team consists of 4 members.
20/4=5 months.

8.1.2 COCOMO MODEL

We will apply the basic COCOMO Model in the calculation of the estimation.It computes software development effort as a function of program size.The formulas used in the computations are:

Effort=a*(LOC/1000)^b
Duration=c*(E ^ d)

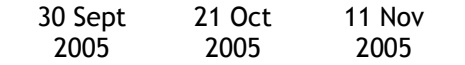
Function	**Estimated LOC**
User Interface Design	3000
2D Geometric Analysis	1500
3D Geometric Analysis	1500
Computer Graphics Display Facilities	3000
Peripheral Control	1000
Estimated lines of code	10000

LOC=10000

a = 3.1
b = 1.05
c = 2.5
d= 0. 35

Effort= 34.1 person-month
Duration=8.4 months
N=E/D=4 people

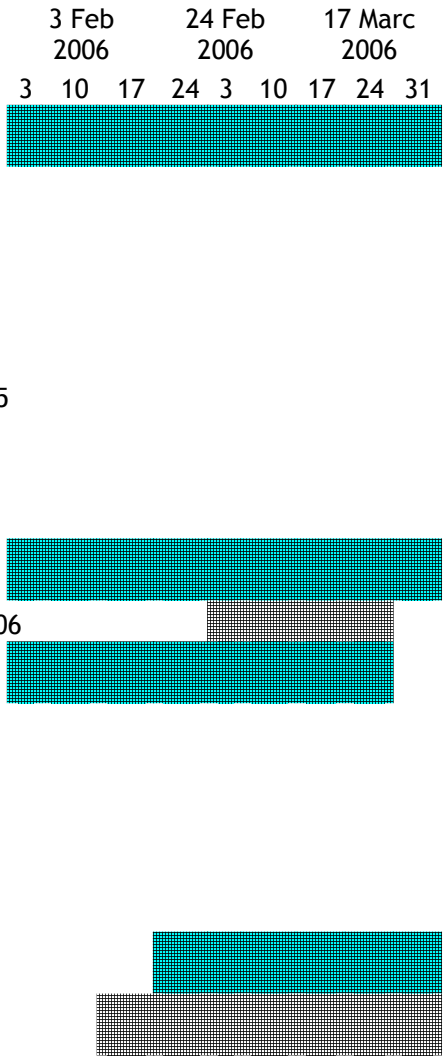
8.2 GANNT CHART



ID	TASKNAME	DURATION	START	FINISH
1	Project Management	150 Days	30 Sept 2005	22 Ma 2006
2	Proposal	5 Days	30 sept 2005	7 Oct 2005
3	Requirement Specification	17 Days	7 Oct 2005	25 Oct 2005
4	Initial Design Report	20 Days	21 Oct 2005	6 Nov 2005
5	Detailed Design Report	30 Days	18 Nov 2005	16 Dec 2005
6	Researching Tools	10 Days	21 Oct 2005	4 Nov 2005
7	Technical Analyse	20 Days	4 Nov 2005	25 Nov '005
8	Design Tool Development	90 Days	27 Jan 2006	21 Apr 2006
9	Controling Objects	30 Days	3 Feb 2006	24 Marc 2006
10	Camera Movement	35 Days	27 Jan 2006	24 Marc 2006
11	Light Addition	20 Days	7 Apr 2006	21 Apr 2006
12	Importing Objects	10 Days	21 Apr 2006	05 Ma 2006
13	Importing Audio	5 Days	5 Ma 2006	12 Ma 2006
14	Library Development	10 Days	7 Apr 2006	21 Apr 2006
15	Animation Creation Process	80 Days	24 Feb 2006	19 Ma 2006
16	Timeline Development	50 Days	17 Feb 2006	12 Apr 2006
17	Menu Design	5 Days	5 Ma 2006	12 Ma 2006
18	Help Menu Design	3 Days	28 Apr 2006	5 Ma 2006
19	Tutorial Preperation	5 Days	28 Apr 2006	5 Ma 2006
20	Testing Design Tool	30 Days	7 Apr 2006	5 Ma 2006
21	Finilazing the Design Tool	10 Days	5 Ma 2006	19 Ma 2006

ID	TASKNAME	DURATION	START	FINISH
1	Project Management	150 Days	30 Sept 2005	22 Ma 2006
2	Proposal	5 Days	30 sept 2005	7 Oct 2005
3	Requirement Specification	17 Days	7 Oct 2005	25 Oct 2005
4	Initial Design Report	20 Days	21 Oct 2005	6 Nov 2005
5	Detailed Design Report	30 Days	18 Nov 2005	16 Dec 2005
6	Researching Tools	10 Days	21 Oct 2005	4 Nov 2005
7	Technical Analyse	20 Days	4 Nov 2005	25 Nov '005
8	Design Tool Development	90 Days	27 Jan 2006	21 Apr 2006
9	Controling Objects	30 Days	3 Feb 2006	24 Marc 2006
10	Camera Movement	35 Days	27 Jan 2006	24 Marc 2006
11	Light Addition	20 Days	7 Apr 2006	21 Apr 2006
12	Importing Objects	10 Days	21 Apr 2006	05 Ma 2006
13	Importing Audio	5 Days	5 Ma 2006	12 Ma 2006
14	Library Development	10 Days	7 Apr 2006	21 Apr 2006
15	Animation Creation Process	80 Days	24 Feb 2006	19 Ma 2006
16	Timeline Development	50 Days	17 Feb 2006	12 Apr 2006
17	Menu Design	5 Days	5 Ma 2006	12 Ma 2006
18	Help Menu Design	3 Days	28 Apr 2006	5 Ma 2006
19	Tutorial Preperation	5 Days	28 Apr 2006	5 Ma 2006
20	Testing Design Tool	30 Days	7 Apr 2006	5 Ma 2006
21	Finilazing the Design Tool	10 Days	5 Ma 2006	19 Ma 2006

ID	TASKNAME	DURATION	START	FINISH
1	Project Management	150 Days	30 Sept 2005	22 Ma 2006
2	Proposal	5 Days	30 sept 2005	7 Oct 2005
3	Requirement Specification	17 Days	7 Oct 2005	25 Oct 2005
4	Initial Design Report	20 Days	21 Oct 2005	6 Nov 2005
5	Detailed Design Report	30 Days	18 Nov 2005	16 Dec 2005
6	Researching Tools	10 Days	21 Oct 2005	4 Nov 2005
7	Technical Analyse	20 Days	4 Nov 2005	25 Nov '005
8	Design Tool Development	90 Days	27 Jan 2006	21 Apr 2006
9	Controling Objects	30 Days	3 Feb 2006	24 Marc 2006
10	Camera Movement	35 Days	27 Jan 2006	24 Marc 2006
11	Light Addition	20 Days	7 Apr 2006	21 Apr 2006
12	Importing Objects	10 Days	21 Apr 2006	05 Ma 2006
13	Importing Audio	5 Days	5 Ma 2006	12 Ma 2006
14	Library Development	10 Days	7 Apr 2006	21 Apr 2006
15	Animation Creation Process	80 Days	24 Feb 2006	19 Ma 2006
16	Timeline Development	50 Days	17 Feb 2006	12 Apr 2006
17	Menu Design	5 Days	5 Ma 2006	12 Ma 2006
18	Help Menu Design	3 Days	28 Apr 2006	5 Ma 2006
19	Tutorial Preperation	5 Days	28 Apr 2006	5 Ma 2006
20	Testing Design Tool	30 Days	7 Apr 2006	5 Ma 2006
21	Finilazing the Design Tool	10 Days	5 Ma 2006	19 Ma 2006



ID	TASKNAME	DURATION	START	FINISH
1	Project Management	150 Days	30 Sept 2005	22 Ma 2006
2	Proposal	5 Days	30 sept 2005	7 Oct 2005
3	Requirement Specification	17 Days	7 Oct 2005	25 Oct 2005
4	Initial Design Report	20 Days	21 Oct 2005	6 Nov 2005
5	Detailed Design Report	30 Days	18 Nov 2005	16 Dec 2005
6	Researching Tools	10 Days	21 Oct 2005	4 Nov 2005
7	Technical Analyse	20 Days	4 Nov 2005	25 Nov '005
8	Design Tool Development	90 Days	27 Jan 2006	21 Apr 2006
9	Controling Objects	30 Days	3 Feb 2006	24 Marc 2006
10	Camera Movement	35 Days	27 Jan 2006	24 Marc 2006
11	Light Addition	20 Days	7 Apr 2006	21 Apr 2006
12	Importing Objects	10 Days	21 Apr 2006	05 Ma 2006
13	Importing Audio	5 Days	5 Ma 2006	12 Ma 2006
14	Library Development	10 Days	7 Apr 2006	21 Apr 2006
15	Animation Creation Process	80 Days	24 Feb 2006	19 Ma 2006
16	Timeline Development	50 Days	17 Feb 2006	12 Apr 2006
17	Menu Design	5 Days	5 Ma 2006	12 Ma 2006
18	Help Menu Design	3 Days	28 Apr 2006	5 Ma 2006
19	Tutorial Preperation	5 Days	28 Apr 2006	5 Ma 2006
20	Testing Design Tool	30 Days	7 Apr 2006	5 Ma 2006
21	Finilazing the Design Tool	10 Days	5 Ma 2006	19 Ma 2006