Game design and development

Vani Indrani Surangi

GAME DESIGN AND DEVELOPMENT

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Vani Indrani Surangi

December 2006

GAME DESIGN AND DEVELOPMENT

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Vani Indrani Surangi

December 2006

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science

11/12/2006

Date

[redacted]

Dr. Tong Lai Yu

[redacted]

Dr. Kerstin Voigt

ABSTRACT

This project was performed to get hands on experience with present day technology used in games. Most of the project involved research about which tools suit particular game genres, and how to use those tools to develop 3D games. As part of learning these tools, a 3D adventure game called "Adventures of Smiley" was developed using macromedia Director MX and 3D Studio Max. The purpose of this game is to involve children with a fun way of learning using a friendly interface. This game can be extended to any age group based on the difficulty level of lessons, puzzles, etc. The research carried out throughout this project gathered a lot of information about the current game industry, technologies and game genres, and can be used as a reference for the beginner level game programmer. The total research performed throughout the project is documented and also published on the Internet. The documentation can help novice game programmers to find the right tools for their game idea. The game is also published on the Internet, and can be accessed by anyone throughout the world for free.

ACKNOWLEDGMENTS

First, I thank my advisor David Turner, for his continuous support in my project. Dr. Turner was always there to listen and to give advice. He is responsible for involving me in the research about current technologies. Special thanks go to Dr. Tong Lai Yu for inspiring me to start a project in computer graphics and helping throughout my independent study and project. I also want to express my sincere gratitude to Dr. Kerstin Voigt for her wisdom, understanding, and for valuable advice. Besides my Project committee, I would like to thank the rest of computer science department faculty and staff: Dr. Josephine Mendoza, graduate advisor, for guiding me throughout my masters program, Dr. Ernesto Gomez, for having me in the Parallel C compiler research. I am most indebted to Dean Utian, of the University of new South Wales, Sydney, Australia for his great tutorials on Director and helping me throughout this project.

I thank my family: my parents, Mohana Rao Surangi and Nirmala Devi Surangi for giving me life in the first place, for educating me with aspects from both arts and sciences, for unconditional support and encouragement to pursue my interests, even when the interests went beyond boundaries of language, field and geography. I also thank my dear husband Kalyana Mangaraju for his great help and for reminding me that my project should always be useful and serve good purposes, my sister Raja Vani Surangi, my brother-in-law Naresh Yerra for listening to my complaints, frustrations and for believing in me, and my brother Raj Bharath Surangi and my friend Vamshi Kankanala for encouraging me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Game design and development necessitates 3 things: motivation, time, and talent. A game engineer also needs game logic and game content. I always want to use my artistic skills to achieve something special in any area along with technical knowledge I gained through my studies. Specializing in Game Design and Development complements the breadth and depth of knowledge I acquired in my major with a multidisciplinary understanding of the role and basic mechanics of games in contemporary society. After the idea of a project on game design and development came to me, I spent most of the time researching about games and part of the time learning some specific tools. As a byproduct of this project, I am better prepared to get employed in web development, which is another of interest to me.

Purpose

The purpose of this project on game design and development is to get hands-on-experience with present day technology used in games. Most of the project involves research about which tools suit particular game genres, and how to use those tools to develop 3D games.

As part of learning these tools, a 3D adventure game was developed using Macromedia Director MX and 3D Studio Max. The purpose of the game "Adventures of Smiley" is to involve children with a fun way of learning using a friendly. This game can be extended to any age group based on the difficulty level of lessons, puzzles etc. Educational games are chosen because 90% of the age group below 20 years spends considerable amounts of time playing online or video games. Attractive educational games with good sound effects can encourage children to learn while playing. Purpose of these games is to teach skills such as analytical thinking, team building, multitasking and problem-solving under stress.

I chose to research in this area to get myself to learn different tools and 3D modeling where my artistic abilities can be used. I chose Macromedia Director MX to make the game reach a worldwide audience by publishing on the Internet.

Scope

This project can be extended to any age group by adding more content to the game, such as puzzles, mazes, reading comprehension, etc. The project is published on the Internet, and can be accessed anywhere in the world.

The game itself is divided into modules, so the download time is reduced. The programming language used through out this project is Lingo, which is an object-oriented language. With minimal code changes, the game can be easily customized to support new game modules, so further enhancement of the game can be accomplished efficiently. The project documentation can also be used for new programmers to aid them in deciding which tool is best for them. The documentation is going to be published on the Internet, and sub-divided into sections to find the information easily.

Significance

The significance of the project is to involve children with a fun way of learning by using a friendly interface, and improve their concentration level by presenting lessons and articles in an attractive way. The research carried out throughout this project gathered lots of information about the current game industry, technologies and game genres, and can be used as a reference for the beginner level game programmer. When I started the project it took me lot of time to find the right tool for my game idea, and which would help me to get employed in interesting jobs like web design, animation, game programming, etc. So, I determined that 3D Studio Max is the right tool for me to get used to 3D modeling, because it is widely used in game and animation industries.

Shockwave 3D, the player of movie files created in Macromedia Director, is an alternative to the next generation of online 3D games and websites. The drawback of any 3D game is the time to load. Macromedia Director reduces this time by compressing the size of the movie by 60% without compromising the quality of the game. This project used most of the nice features of 3D Studio max and Macromedia Director to create the game "Adventures of Smiley." The CNN article "Not playing around: Scientists say video games can reshape education" [13] points out the potential value of video games for education. It was the intent of this project to make a contribution in this area.

CHAPTER TWO

RESEARCH

Research is the core part this project on game design and development, and this section describes the research I did step by step in finding the right tools for the project as well as about artificial intelligence in games.

Artificial Intelligence in Games

Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. On the player side, Artificial Intelligence in games is usually used for creating the player's opponents. Usually the players want good opponents who can fight really well, but the players get the chance to win most of the time. Here, AI is used not just for creating opponent's steps but also to support the player. Nowadays, quality AI is becoming a selling point to computer games [7].

Classic Games Using Artificial Intelligence Technique

The following table shows some classic games that use artificial intelligence techniques, and gives a brief description of them. The descriptions appearing in this table is taken from [6] [7] [12].

Table 1. Classic Artificial Intelligence Games

Game	Description
Twenty Questions	Twenty Questions is a popular spoken parlour game for two or more the players. It encourages deductive reasoning and creativity.
Master Mind	Mastermind is a simple code-breaking board game for two the players, invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert.
Connect Four	Connect Four (also known as Plot Four) is a two-the player board game in which the players take turns in dropping discs into a vertical grid with the objective of getting four of one's own discs in a line.
Checkers	Checkers is a group of abstract strategy board games between two the players which involve diagonal moves of uniform pieces and mandatory captures by jumping over the enemy's pieces
Qubic (4x4x4 tic-tac-toe):	Cubic plays three-dimensional tic-tac-toe on a 4x4x4 board. Moves are specified as a sequence of three coordinate numbers in the range 1-4.
Chess	Chess is an abstract strategy board game and mental sport for two the players. The object of the game is to checkmate the opponent's king. This occurs when the king is under immediate attack (in check) and there is no way to prevent it from being captured on the next move.
Scrabble	Scrabble is a popular word game and board game in which 2-4 the players score points by forming words from individual lettered tiles on a 15×15 game board. The words are formed across and down in crossword fashion, and must appear in a standard dictionary.
Bridge	Bridge is a trick-taking card game of skill and chance (the relative proportions depend on the variant played). It is played by four the players who form two partnerships (sides); the partners sit opposite each other at a table. The game consists of the auction (often called bidding) and play, after which the hand is scored.

First Step of the Research

After I finished my independent study in computer graphics, I designed a game that involves a fun way of learning. To use the recent technology, I wanted to do the game in 3D, and to publish the game on the Internet. The first idea that came into my mind is to start coding everything in C++ and OpenGL, just like I did for my independent study. But, after some research, I found out that this is not exactly what I wanted, because I need to train myself in 3D modeling and acquire knowledge of current tools that are widely used, so that I could more easily find a job in the fields of animation and website design. So, I started researching about game creation software packed with artistic resources, such as models, textures and sound effects. I also did some research about how game genres are used to differentiate games.

Game Genres

The term 'genre' refers to a particular type or style. Computer games are usually classified into broad genre categories, such as Shooter, Racer, Adventure and Platform games. A game genre is a type or category of game. Game genres are often hard to classify as some games may fit in more than one genre. Here's a list of the most popular genres classified by International Hobo [14] and they have been further researched by me. After this research, I came to know that my game falls under both the Educational and Adventure genres. Then I came up with the funky title "Adventures of Smiley" for my game.

The Purpose of Genres

The purpose of genres is to classify things easily. This makes it easy to describe a game to potential users, by calling it adventurous, educational, etc. If someone played a game and likes it, then it is easy to find similar games based on genre search.

Shooter

Shooter genre involves the players shooting certain objects and escaping from the crossfire. Usually, the player has a number of lives and bonus lives can be obtained by good shooting. Nowadays, most of the selling market of games is within the shooter genre and usually of 3D point of view like first person or second person.

Bat and Ball

The Bat and Ball genre usually comprises of various styles of games but it is outdated now. Here, the player control a paddle and uses it to bounce a ball and blast a number of blocks. The player number of lives depends on how many times they loose the ball. Ping-pong and Breakout are some of classic games in this genre.

Racer

Racers genre is also one of popular genre these days. Here, the players need to complete a track in limited time and after reaching they are allowed to play on more advanced track. The players also need to escape from obstacles faced during the racing. These games are mostly done in 3D.

Video Pinball

Video Pinball is a small genre having all table top games. The game play is almost similar to ordinary table top games.

Puzzle

Puzzle games are small but very popular in gaming industry. Here, the players need to solve some kind of puzzles like maze and need to find the optimal way to solve the puzzle. Games fall into this genre were so colorful and addictive. They also challenge hand eye coordination.

Strategy

Strategy games are also the major part of the gaming market. Here, the player needs to use units which are under his in charge and to defeat opponent the player.

These games are usually turn based and the player must make decisions on how to use the units effectively in his/her turn.

Adventure

Adventures games are really popular in 1980's and 90's. Here, the player has to finish some task by solving some intermediate games. The progression of the game is based on sub games. For example, in my game "Adventures of Smiley" the player has to find the key to go to next room in order to finish the game. While playing, the player has to solve puzzles, answer questions based on comprehensions etc.

Video Boardgame

The Video Boardgame genre has the computerized version of board games. Here, the player has to play the original board game on the computer.

Fighting

Fighting game is one of the more popular modern game genres. The highest percentage of players are male. Here, the player has to fight with the opponent and make him unconscious or dead. The player has a number of different moves available and with them he has to eliminate the opponent. The player also needs to think which move is best for the current scene.

Sim

Sim games are usually played by simulating the conditions of a particular environment, real or imagined. Here, the player needs to control multiple resources, and achieve a target.

Computer Role Playing Game

Computer RPG genre is also one of the commercially important genres. Here, the player needs to control a character in its environment. Players also need to interact with other beings and depending on their actions or choice the character is modified.

Platform

Platform games are very popular and started with Nintendo. Here, the game play revolves around jumping between platforms. The player's number of lives is affected by the way he is playing and escaping from certain type of hazards happen in that platform.

Sports

Sports games are so popular and they imitate real or imagined sports. Sports games have become more and more sophisticated over the years and commercial market is also very high.

Arcade Adventure

Arcade adventure games are a combination of adventure games with conventional games. Here, the player is in control of a character and plays, interacts with it in real time.

Rhythm-Dance

The Rhythm-Dance genre is the youngest of all genres. Here the player need to perform actions based on the music.

Game Engines

As a part of my research about game creations I studied different types of game related software tools. Out of them, tools that really attracted me were game engines.

On October, 2006, Wikipedia defined Game engines as below:

> A game engine is the core software component of a computer or video game or other interactive application with real-time graphics. It provides the underlying technologies, simplifies development, and often enables the game to run on multiple platforms such as game consoles and Microsoft Windows. The core functionality typically provided by a game engine

includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection, sound, scripting, animation, artificial intelligence, networking, and a scene graph [12].

A typical game engine structure is shown in figure 1.

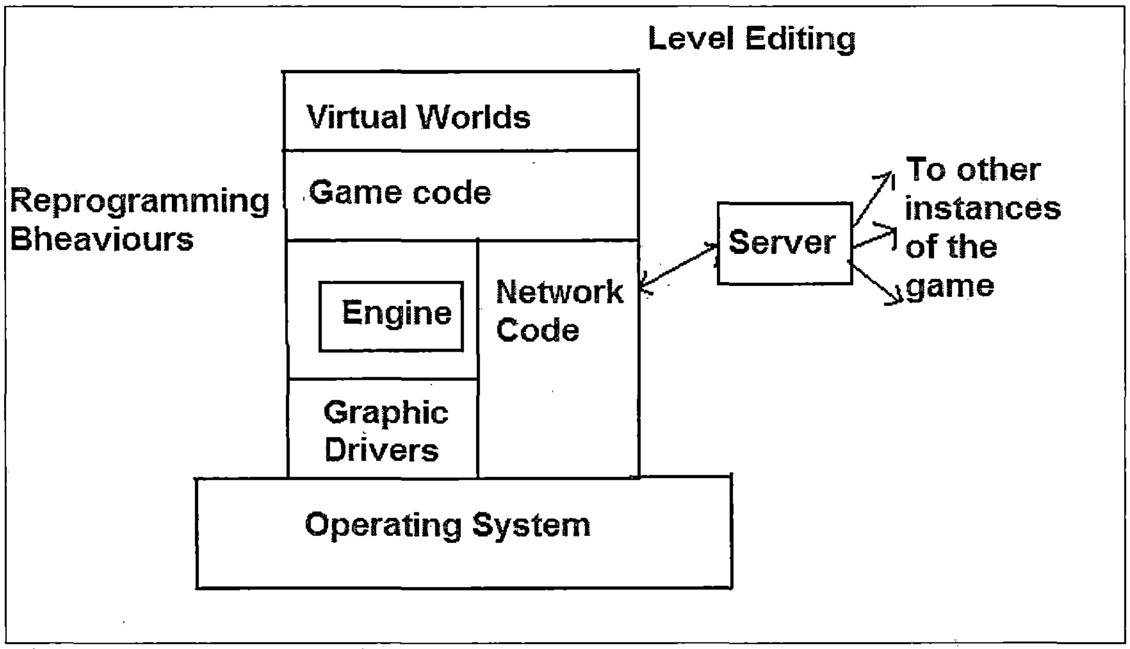


Figure 1. Game Engine Structure

Game Engine Subsystems

Based on article from Gamepipe labs, I observed that typical engine subsystems include [5]:

- Renderer draws the scene with all appropriate objects & effects to the screen.
- Collision detection is used to calculate collisions between objects.

- Physics are used to simulate gravity and other physics related elements.
- Input / output system is used for handling file operations, keyboard, mouse, etc.
- Sound & music system handles playing of sounds & music.
- AI system is used to have computer controlled opponents & allies.
- Network system is essential for multiplayer games played over networks.
- Database system is used storing the data.
- GUI system is used for creating and handling windows, menus, huds, etc.

Table 2 shows details about some of best 3D open source game engines available today [3].

Table 2. 3D Open Source Game Engines

Name	Language	Platform	Graphics	Sound	Networking
OGRE	C++	Windows, Linux, MacOS X	3D via Directx or OpenGL	No	No
Crystal Space	C/C++	Linux, Windows, MacOS X	3D via OpenGL	Yes	No
Irrlicht	C++/.NET	Windows, Linux	3D via DirectX(8,9), OpenGL or various software renderers	No	No
Blender	C/C++, Python	Windows, Linux, MacOS, Solaris, FreeBSD, Irix	3D via OpenGL	SDL	Yes
jMonkey Engine	Java	Windows, Linux, MacOS X	3D via DirectX	No	No
Panda 3D	C++	Windows, Linux	3D via OpenGL	Yes, via FMOD	No
Reality Factory	None needed	Windows	3D via Genesis3D (DirectX)	Yes	Yes
Ray Game Designer 2	None needed	Windows	3D via OpenGL or Direct3D	Yes	No
Jad Engine - C# + MDX Game Engine	C#	Windows	3D via Managed DirectX	MDSound And Vorbis .NET	No
Jet3D	C/C++	Windows	3D via DirectX	No	Yes
NeL	C/C++	Windows, Linux	3D via DirectX or OpenGL	Yes	Yes
NemoX 3D Engine	VB/Delphi /.NET	Windows	3D via DirectX	DirectX	Yes
Raydium 3D	C	Windows, Linux	3D via OpenGL	Yes, via OpenAL	Yes
Revolutio n3D	VB/C++/ .NET	Windows	3D via DirectX	Yes	No
The Nebula Device 2	C++	Windows	3D via DirectX	Yes	Yes
Truevisio n3D	VB/Delphi /C++/.NET	Windows	3D via DirectX	DirectX	Yes
Visual3D. NET	.NET 2.0 (C#)	Windows, Xbox 360, Windows Mobile devices	3D via DirectX or XNA	Yes	Yes, via Suva3D

Object-Oriented Graphics Rendering Engine

The game engine which got rave reviews and tops the open source engines list attracted my attention, and thought of using it for my project. So, I did further research about the engine and this section include some features of OGRE. On Feb, 2005, the developing team of OGRE described OGRE as:

> OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes [8].

Some of the features of OGRE have been categorized in the following table.

Table 3. Object-Oriented Graphics Rendering Engine

Description

Features	Description
General features	Object-Oriented Design, Plug-in Architecture, Save/Load System: • Simple, easy to use OO interface designed to minimize the effort required to render 3D scenes, and to be independent of 3D implementation i.e. Direct3D/OpenGL. • Flexible plug-in architecture allows engine to be extended without recompilation • Clean, design and full documentation of all engine classes • Support for ZIP/PK3 for archiving
Scripting	• Scripted material language allows you to maintain material assets outside of your code • Scriptable multipass rendering
Physics	Basic Physics, Collision Detection, Rigid Body: • Controllers allow you to easily organize derived values between objects • Includes bindings for multiple 3rd party collision / physics systems (ODE, Novodex and Tokamak)
Lighting	Per-vertex, Per-pixel, Lightmapping: • Can have an unlimited number of lights in the scene • Supported through vertex and fragment programs.
Shadows	Shadow Mapping, Shadow Volume: • Techniques supported: modulative stencil, additive stencil, modulative projective • Multiple stencil shadow optimisations, including vertex program extrusion, smart light and shadow caster culling, integration with mesh LOD, zpass and zfail methods, 2-sided stencilling, scissor region clipping • Texture shadows fade out at far distance
Texturing	Basic, Multi-texturing, Bumpmapping, Mipmapping, Volumetric, Projected: • Projective texturing automatically links with texture unit to a Frustum instance • can register external texture sources in order to feed texture data in from another source • Supports PNG, JPEG, TGA, BMP and DDS image files

Features	Description
Shaders	Vertex, Pixel, High Level: • Supports vertex and fragment programs (shaders), both low-level programs written in assembler, and high-level programs written in Cg or DirectX9 HLSL, and provides automatic support for many commonly bound constant parameters
Scene Management	General, BSP, Octrees, Occlusion Culling, LOD: • Highly customisable, flexible scene management, not tied to any single scene type. Use predefined classes for scene organisation if they suit or plug in your own subclass to gain full control over the scene organization • Hierarchical scene graph • Scene querying features
Animation	Inverse Kinematics, Skeletal Animation, Animation Blending: • Skeletal animation, including blending of multiple animations, variable bone weight skinning.
Meshes	Mesh Loading, Skinning, Progressive: • Hardware-accelerated skinning • Flexible mesh data formats accepted • Export from many modeling tools including Milkshape3D, 3D Studio Max, Maya, Blender and Wings3D.
Surfaces & Curves	Splines: • Biquadric Bezier patches for curved surfaces
Special Effects	Environment Mapping, Lens Flares, Billboarding, Particle System, Motion Blur, Sky, Water, Fog: • Particle Systems, including easily extensible emitters and affecters (customisable through plugins). Systems can be defined in text scripts for easy tweaking • Support for skyboxes, sky planes and skydomes, very easy to use

Features	Description
Rendering	Fixed-function, Render-to-Texture, Fonts, GUI: • Scriptable multipass rendering • Material LOD • Supports the complete range of fixed function operations such as multitexture and multipass blending, texture coordinate generation and modification, independent color and alpha operations for non-programmable hardware • Support for multiple material techniques • Transparent objects automatically managed • Font system: TrueType fonts and precreated textures • 2D GUI System with Buttons, Lists, Edit boxes, Scrollbars, etc.

GNU Lesser General Public License

As I found out that OGRE is licensed under LGPL I did some research on it. The GNU Lesser General Public License (formerly the GNU Library General Public License) is a free software license published by the Free Software Foundation [8]. The GNU Lesser General Public License was written in 1991 by Richard Stallman, with legal advice from Eben Moglen. If a product is licensed under GNU Lesser General Public License the following conditions apply:

- Release any modifications made to the source
- Pass on the source to the product authors with all the copyrights intact
- Declare any customizations

Why not Object-Oriented Graphics Rendering Engine?

There is no doubt that OGRE is one of the best available open source game engines. But, after spending more than 2 weeks researching about all the features of OGRE I questioned myself whether this engine is really suitable for this project. The first problem encountered is, I cannot have the ready-to-go online version of the game and it is going to be tedious process to prepare the online version. The next thing is time frame, I have mere 2 months to learn the tools to produce my project and OGRE is definitely going to take at least 4 months to get started on an own project. The final problem I found is the complexity of the source code, That is the main drawback of OGRE actually and of the little time I have, it is too hard to get used to that! Then, I started my quest for an easier but compact tool for my project.

3D in Flash

So, while I am searching the internet for the right tool for my project I came to know that we can do 3D in Flash. That is a delighted news for me as Flash is one of the best tools that web designers use nowadays. So, having 3D effects in Flash would be more appealing. I spent almost 4 weeks of time finding information about how to do the 3D in Flash and learning the tools involved in the process [9].

<u>Prerendered 3D Animation</u>

So, I thought of going on the first way to prerender 3D animation as it gives me the opportunity to learn a 3D modeling tool. By this time I have already started on learning 3D Studio Max on side to get the best use of time. So, I wanted to use models I have created using 3DS Max and find a way to import them into Flash. Swift3D is that kind of intermediate software to render out 3D scenes we build within programs which are then exported into .swf files, movie files such as .avi and .mov, or some other format like sequences of images which can then be imported and manipulated within Flash [15].

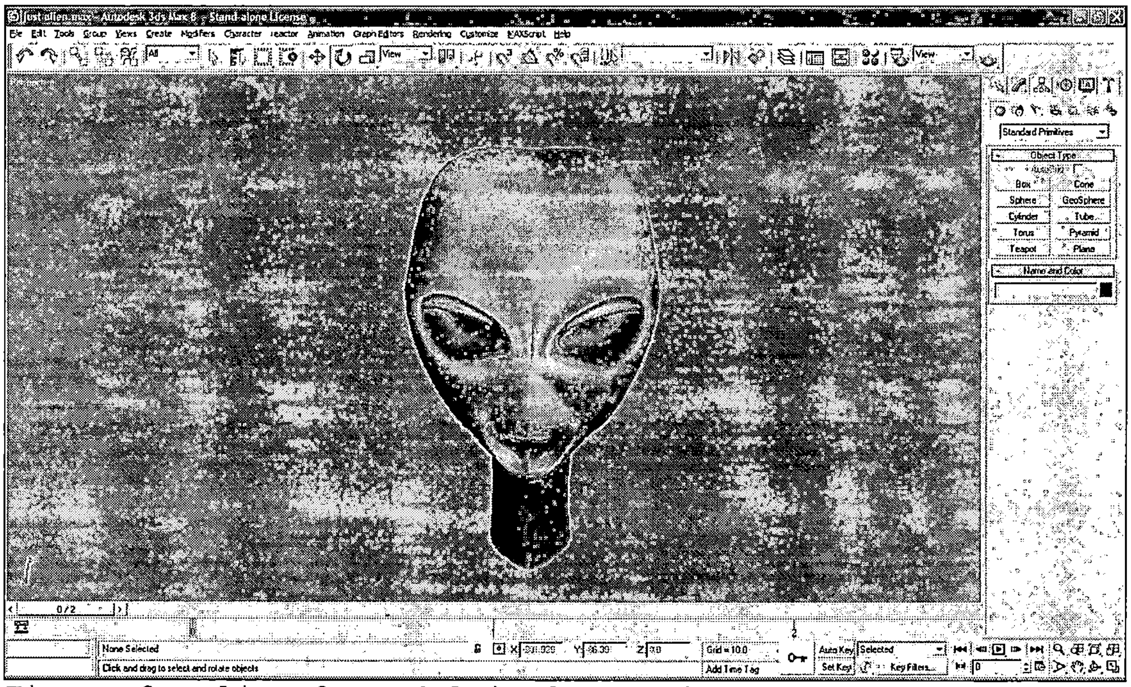
<u>3D Studio Max to Flash Using Swift3D</u>

From the developing team of Swift3D, it is described as below on Aug, 2004:

> Swift3D is the only 3D application to directly integrate with Macromedia Flash and provides unrivaled vector rendering quality and output style options. Swift3D's toolset and interface allow anyone to quickly create 3D content while providing a full set of advanced tools to grow into. With both vector and video export capabilities, Swift3D provides the

entire motion graphics design industry with a powerful, easy to use 3D solution that delivers high quality results for an affordable price [16].

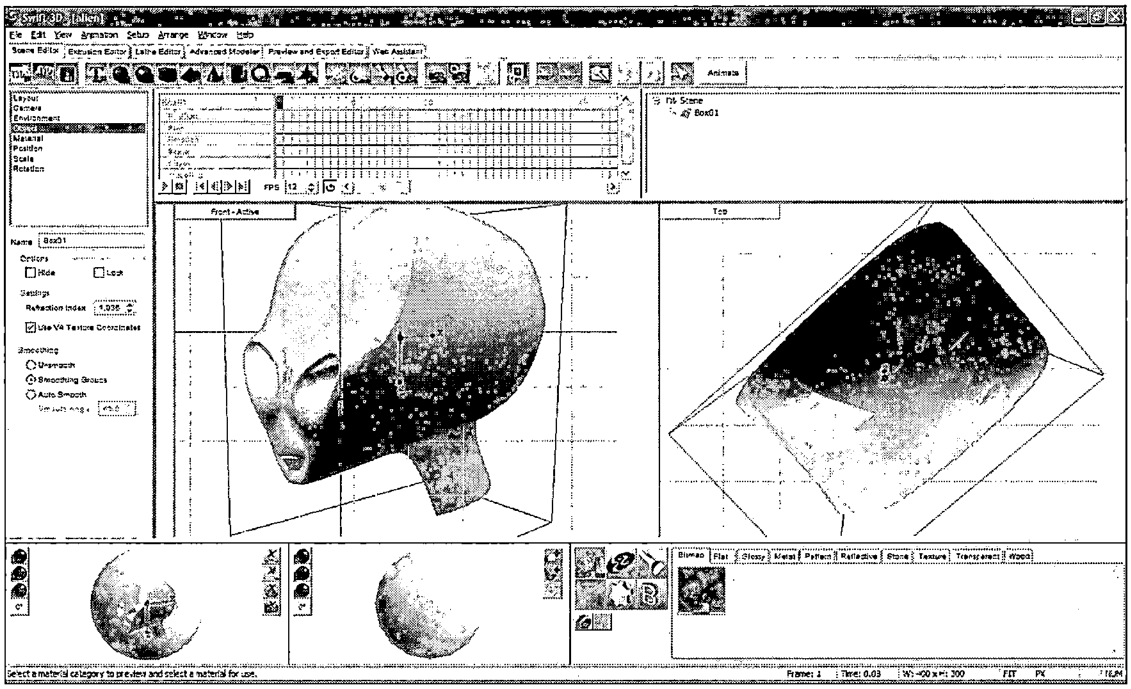
The following figure shows the alien model I created using 3D Studio Max.



just alien.max - Autodesk 3ds Max 8 - Stand-alone License
File Edit Tools Group Views Create Modifiers Character reactor Animation Graph Editors Rendering Customize MAXScript Help
Standard Primitives
Object Type
Box
Cone
Sphere
GeoSphere
Cylinder
Tube
Torus
Pyramid
Teapot
Plane
Name and Color
0 / 2
None Selected
Click and drag to select and rotate objects
Grid = 10.0
Add Time Tag
Auto Key
Selected
Set Key
Key Filters...

Figure 2. Alien 3D Model in 3D Studio Max

Then I exported the model into .3Ds file format using File>Export> options. Using the trail version of Swift3D I used one of the features of the tool, Import from a 3Ds file. The model imported using Swift3D is shown in the below figure. As you can see the model is not as crisp as it was in 3D Studio Max.



Name: Box01

Options: Hide, Lock

Settings: Refraction Index 1.035

[✓] Use V4 Texture Coordinates

Smoothing: Unsmooth, [✓ Smoothing Groups], Auto Smooth

Select a material category to preview and select a material for use.

Frame: 1 Time: 0.03

Figure 3. Imported Alien 3D Model in Swift3D

Then, using rasterized rendering option in Swift3D I exported the model to valid .swf format that Flash can recognize and the software was able to achieve exactly like the model it has imported. The Flash file is shown in the below figure.

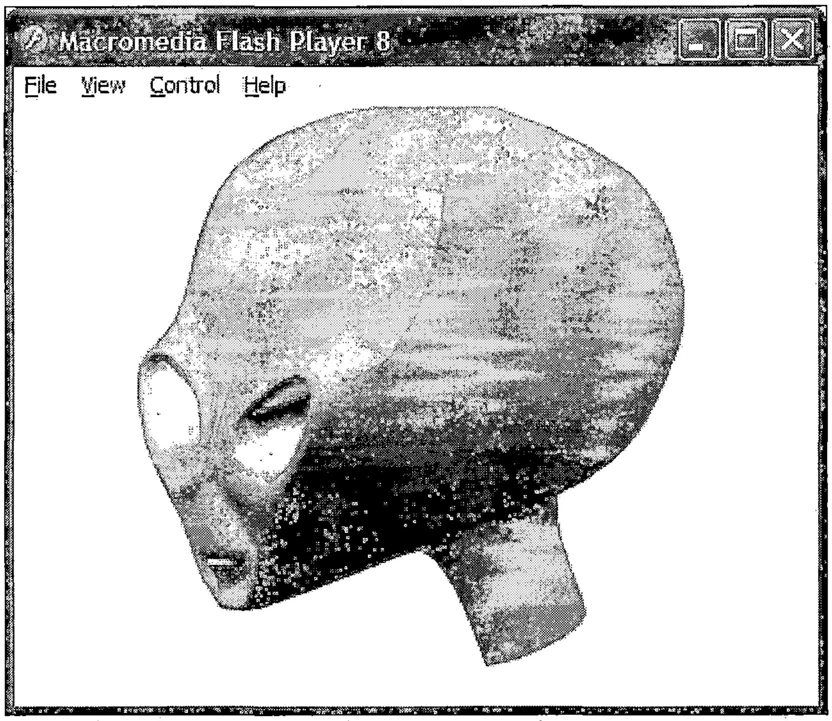


Figure 4. Alien 3D Model in the Desired Flash Format

Isometric Movement in Flash

Though I was able to import the 3D model in Flash I could not make much progress as it is too hard to make the model interactive, which is later explained in detail. This section gives a brief overview of how can we achieve isometric movement in Flash. Isometry is of two types [9].

- Static isometry: Here, most of the objects are fixed to the floor that cannot be moved.
- Dynamic isometry: Here, objects are not fixed to the floor. All objects can be pushed and moved on the floor. Collision detection is done by comparing object's position with all other objects

through looping. The depth level of the object can be changeable.

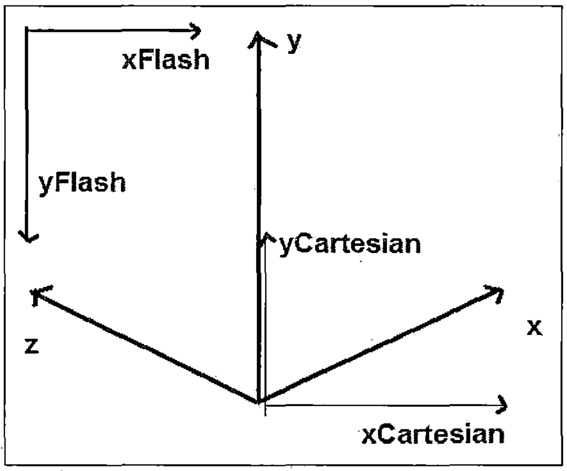


Figure 5. Isometric, Cartesian and Flash Coordinate System

Based on the coordinate system in the above figure, function are written to move a 2D object which looks like a 3D object, using isometric projection as shown in the following figure. Note that the perimeter of the 2D drawing is almost like a regular hexagon, all the black lines are of equal length and all the cube's faces are the same areas.

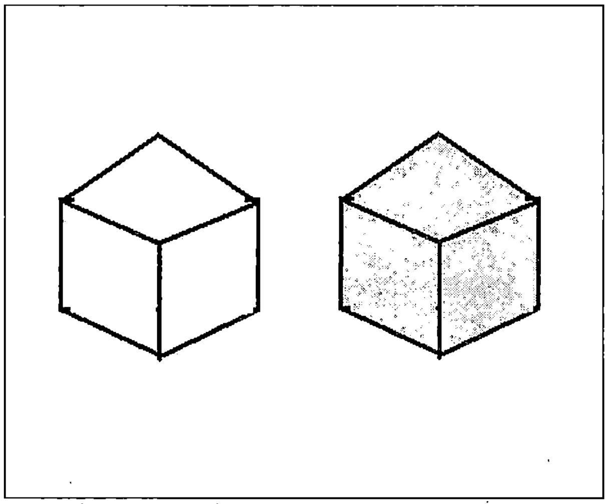


Figure 6. Isometric Drawing of the Box

Why not Flash?

As you see the difference in the screenshots I provided above, the quality of the model is being reduced considerably from 3D Studio Max to Flash. I was not satisfied with this result and also found some other problems I faced while trying to make the model interactive in Flash using the action script. Though Swift3D does the whole work of importing and exporting models into a valid format, it makes the interactivity hard. In other words, there's little I can do with that kind of 3D once it starts playing.

Everything is pre-rendered and therefore could not be changed while playing in the Flash player. The next problem is too much work to get the 3D model to Flash, as the model count goes up the whole work is going to be tiresome. Though I was able to achieve isometric projection and the 3D model in Flash I am not satisfied with the result. So, I am on search for some other tool like Flash which can help to get my project working.

Why Choose Director?

After a little while of research I found about Director which is also a product from Macromedia, the inventor of Flash. I was in dilemma most of the time whether to choose Director or Flash. Finally, I went with Director. So, to support my selection of Director for the project, I gave the comparison table of Flash and Director.

Overview of the Comparison

Flash is better suited for certain types of projects than Director. For instance, interface design for a full site is much more efficient in Flash. But, game development and complicated interface programming is much better in Director. Director has a much stronger timer than Flash. This is essential for games. Director gives far better control over user interaction and stage manipulation [11].

Table 4. Flash versus Director

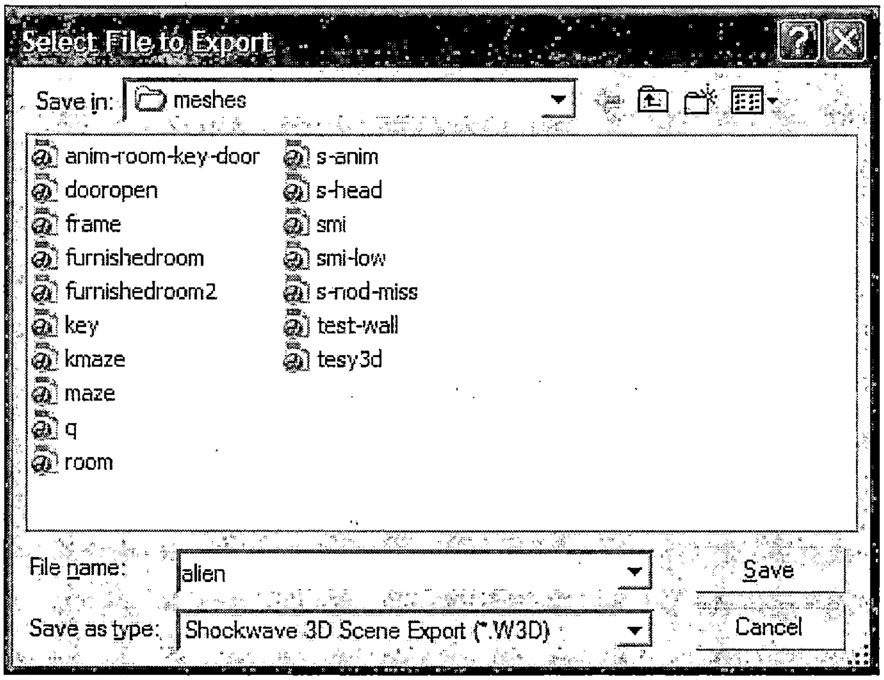
Feature	Flash	Director
Graphics	Vector based • Very good handling of vector based artwork • Exports vector art framework in SWF format • The editor for the vector shapes is accurate and easy to use • Bad way of handling bitmaps • Sometimes, same bitmap look different at each frame	Bitmap based • Not as good as Flash in handling vector artwork • Very good handling of bitmaps like pixel to pixel editing, changing depth, making background transparent etc. • No change in image quality after compile. • Good compression of images without losing quality.
Plug-in	Flash • Simple and fast • Streaming is built in and easy to setup • Much smaller file size due to vector graphics	Shockwave • More intensive and take time to initialize • Very complex setup for streaming • Very big when compared to Flash due to more use of bitmap images
Work Environment	• Almost same as Director except for Keyframe editor • Can edit only one window at a time	• Almost same as Flash, and the timeline is referred as score • Multiple windows can be edited at same time
Keyframe editor	• Too much work for even simple editing. • Multiple resources can not be edited at same time • Complex feature editing of sprites • Multiple objects on one key frame which make editing complex. • Different resource controls like movieclip, button, graphic etc.,	• Much smarter and easy to use • Multiple resources can be edited at same time • Very easy editing of features of sprites • Only one object on one keyframe, which limits the confusion and adds clarity to the animation • Same level of control for all objects

Feature	Flash	Director
The Stage	• Hard to access the stage • Masking is much better and easy to do	• Stage Is like a floating window • Masking is a bit hard than Flash but it is as good as Flash
The Cast and the Library	Cast • List that can be divided into subfolders • Hard to find a cast member	Library • List, graphical representation with no hierarchy • Easy to identify a cast member
Movieclips and FilmLoops	Movie Clips • Easy to do and not even a single problem • Whole Flash movie can be converted into movieclip and can be used in other Flash movie.	Film Loops • Easy to handle but some problems with that like o Hit test o Cannot rotate, blend with Lingo o Improper execution of Lingo from inside movieloops
Playback Controller	• The only way to test a movie is by generating Flash files, and this creates lots of unused Flash files.	• When testing in the workspace, the programs works 100% the way when it is compiled.
Stability	• More likely to crash • Abrupt crashes and corrupts the workfile.	• Less likely to crash • Stay open longer and gives an option to save movie. Though, it sometimes save the project, most of the time it corrupts it.

Feature	Flash	Director
Programming	Action script • Procedure Oriented and syntax close to C. • Easier to make code into movieclip and play it when necessary • One code window at a time • Point and click code, easy for non programmers • Only two choices to attach the code, when the playhead enters a frame, or actions on buttons. • Manual compiling	Lingo • Object oriented and syntax close to visual basic • Call functions and procedure at anytime from anywhere • Multiple code windows at a time • Point and click supported but multiple windows will open. Instead having all the code at one place is easy and more suitable to programmers. • Attach code to objects using a few different methods like, write one behavior and add it hundreds of objects on the stage. • Automatic indenting, formatting and coloring the code. • Automatic compiling when code is saved or play button is hit.
Debugging	• Impossible to find the value in the debugging window	• Watcher allows seeing the value of multiple variables in debugging window.
Variables	• No support for data structures. • Only way to access the variable position is by integer • Hard to use global variables, as we need to specify the right path to it always	• better array handling than Flash • create full data structures of object types so we can access them by name instead of list position • Easy to use global variables
Xtras	• Complex to use • Fairly low number of Xtras or Extensions.	• Easy to use • More number of Xtras available

From 3D Studio Max to Director

3D Studio Max has a plug-in called Shockwave 3D Scene Exporter, .W3D which is a valid format for Director to import the 3D Model into it. The following set of figures show how the exporting in 3DSMAX is done and you can see the model after the import too.



Save in: meshes

anim-room-key-door
dooropen
frame
furnishedroom
furnishedroom2
key
kmaze
maze
q
room
s-anim
s-head
smi
smi-low
s-nod-miss
test-wall
tesy3d

File name: alien

Save as type: Shockwave 3D Scene Export (*.W3D)

Figure 7. Alien 3D Model Exporting from 3D Studio Max

The following figure shows the alien 3D model after imported from W3D file in Director MX. As you can see the model is exactly the same as in 3D max and able to import with all details in modeling, NURBS meshes and animation etc.

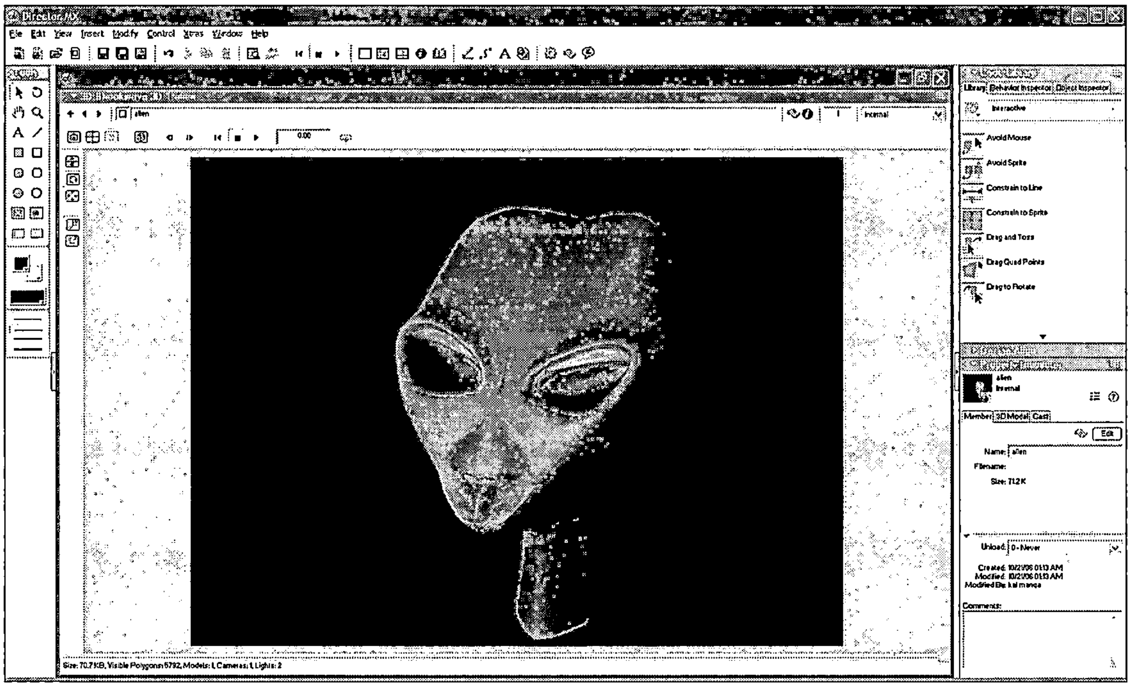


Figure 8. Alien 3D Model in Macromedia Director

Final Decision with Director

The perfect 3D model importing is the first reason that I decided to go with Director. The second reason I choose is the programming language, I always like object oriented approach as it eases the task of programming and increases reusability. The programming language of Director is Lingo, which is object oriented. Last but not least what I like about Director is the possibility of publishing the game on the Internet. It is so easy, quick and even the size of the movie is compressed by the tool without reducing quality. So, it is easy to publish on the Internet and fast to download and play 3D games on the Internet.

CHAPTER THREE
GAME DESIGN AND DEVELOPMENT

This project is an extension of what I have done in my independent study. The game I designed and developed for my independent study is called "Game of Nim." It is programmed with artificial intelligence techniques and graphics using SDL. The other project is a basic Christmas Eve animation and graphics are done through openGL. The current game I developed for this project is "Adventures of Smiley" and uses Macromedia Director MX and 3D Studio Max.

Game of Nim

In this 2D version of NIM, two players alternatively take blocks from a pile of blocks. In each move, a player chooses how many blocks to take. The player must take at least one but at most half of the blocks. Then, the other the player takes a turn. The player who takes the last block looses. In this game, you have to play against computer. Without knowing the winning strategy it is impossible to beat the computer.

The program uses object oriented techniques that allow easy expansion and modification of the program for future use. This program is mutli-threaded, cross-platform, re-usable, user friendly and robust. 90 percent of the code is reusable. Error handling of the game is done efficiently. If it has any problem with loading files or libraries, it just prints an image and closes the game. This program uses Simple DirectMedia Layer (SDL) to generate graphics. SDL is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video frame buffer. The working environment of this game is:

- Language: C++
- Graphic Library: SDL
- Operating System: Cross-platform

  The game is developed on Linux. With few changes it will work on Windows and other platforms too.

Nim is an artificially intelligent game where we have to follow a specific strategy to win the game. Whatever the number of blocks on the table, you have to leave the number of blocks as two power k - 1, for a suitable k. For example, if you have 20 on the board leave 15(2^4-1). This game actually generates random number of blocks from 10 to 100. So, best options is to leave 3, 7, 15, 31, 63 number of blocks on the table. This is the winning strategy of the game. For the interactive play, I haven't added the option like computer can start first. Since, if computer starts there is no way that user can win the game.

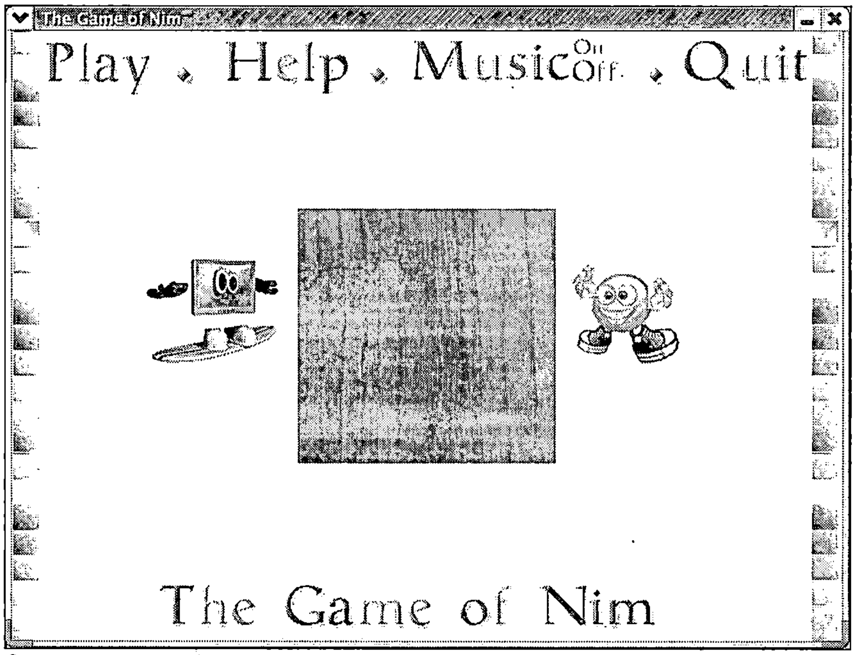


Figure 9. Screenshot of the Game

I have edited and created all images being used in the game with paint program. As you can see the background image is loaded first and later smiley images along with wood board are added to the screen. This is shown in the screen shot of the game at starting position.

After 'p' button is pressed or by clicking 'Play' on screen, actual game starts. I made a little animation with those figures by displaying different images on the screen with a delay of 0.020 seconds. Actual images being used for smiley animation are given below.

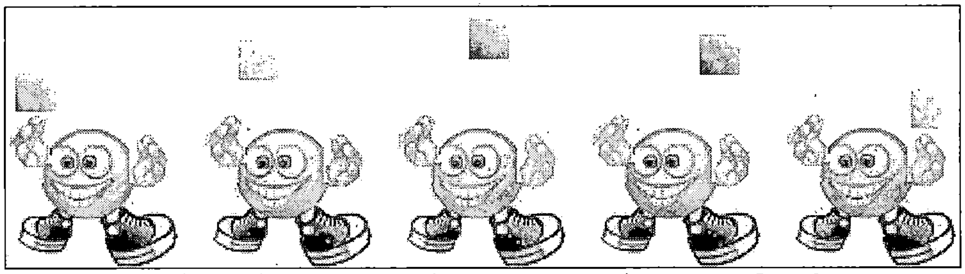


Figure 10. Animation of Smiley Handling a Block

I have created different expressions of both computer and smiley just for fun. People whoever played this game, love those expressions. Some of those expressions are shown below:

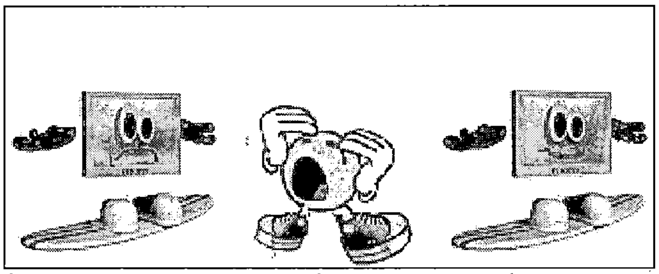


Figure 11. Different Expressions of Smiley and Computer

Blocks on ground are arranged based on the dimension of the wood board. The wood board can hold up to 100 blocks which is basically is the game design. Here is a screenshot of the game while being played.

3D Christmas Eve Animation with OpenGL

The goal of Christmas Eve project is to implement beautiful and realistic 3D graphics with OpenGL and C++ using object oriented techniques. The Scene of Christmas Eve has lot of snowmen, puppets & gift box with good music. All Graphics for this project were done using OpenGL and SDL is used for music. I want to implement most of OpenGL features learned during my study such as display lists, lighting, and reflection etc., Then comes the idea of designing Christmas Eve. Because, it is so colorful and I can add whatever I want to the scene.

This project uses object oriented technology for reusability and expendability. So I choose to code in C++ and created graphics with OpenGL .OpenGL (Open Graphics Library) is a specification defining a cross-language cross-platform API for writing applications that produce 3D computer graphics (and 2D computer graphics as well). The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. It is very popular in the video games industry where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL). OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization and video game development. The development environment of this game is:

- Language : C++
- Graphic Library: OpenGL
- Operating System: Cross-platform

  The game is developed on Linux. With few changes it will work on Windows and other platforms too.

I will explain how I have used OpenGL features for the Christmas Eve animation with screenshots of the animation. The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and the Utah teapot. GLUT even has some limited support for creating pop-up windows.

I have used GLUT shapes for creating most of characters in the project like snowman, gift-box. GLU is used for generating quadratic objects like cylinder for creating Puppet. All mouse and key board events are handled by GLUT functions.

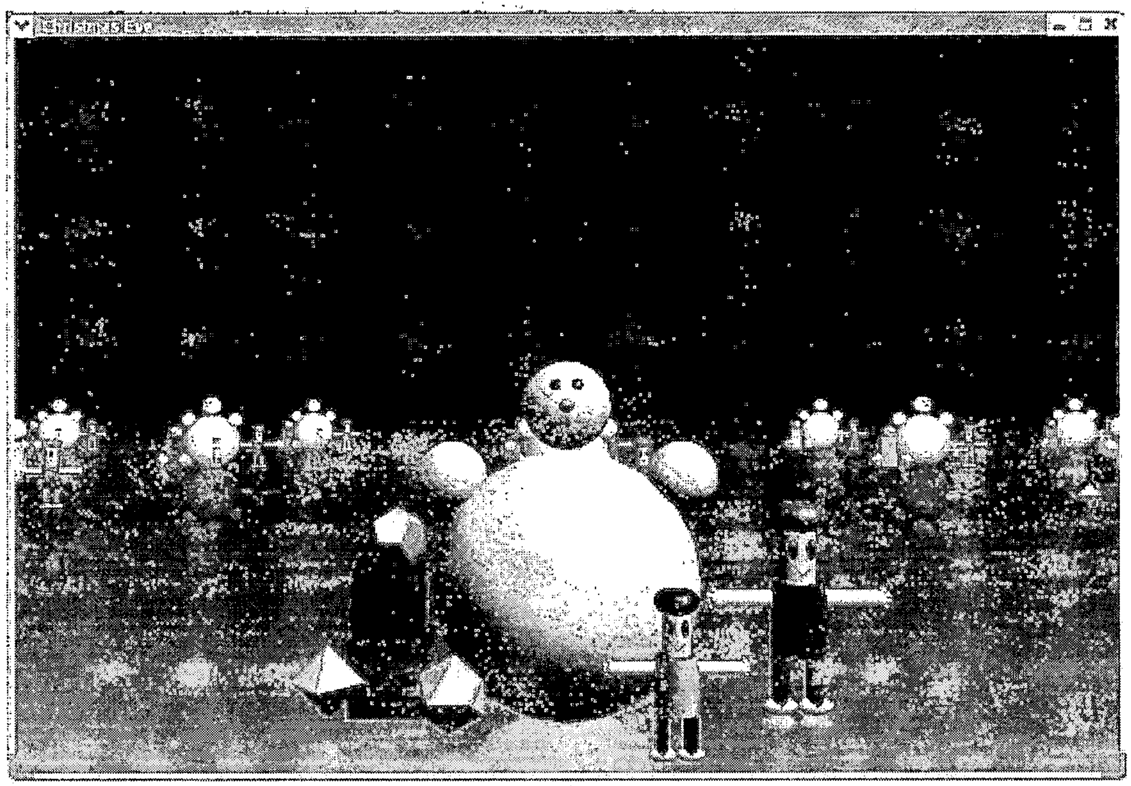


Figure 12. Screenshot of the Animation when Started

glLight is used for lighting which gives actual depth to the scene and made it more realistic too. Using different parameters with glLighting I was able to achieve different variations in lighting such as ambient, diffuse, specular, spotlight etc., Light can be moved by holding and moving mouse on the screen.

The reflection of the scene is achieved by making the ground transparent and drawing whole scene on top of the ground at the bottom of ground by flipping it using glScalef() on negative Y-axis. Transparency of the ground is achieved by using GL_BLEND and glBlendFunc().

Animation of Christmas is done by moving the legs of big puppet and rotating the small puppet. When you animate I was able to achieve moving red, green and blue spotlights which gives effect of combination of multiple colors and make it look more festive. The screen shot of animation is shown below.

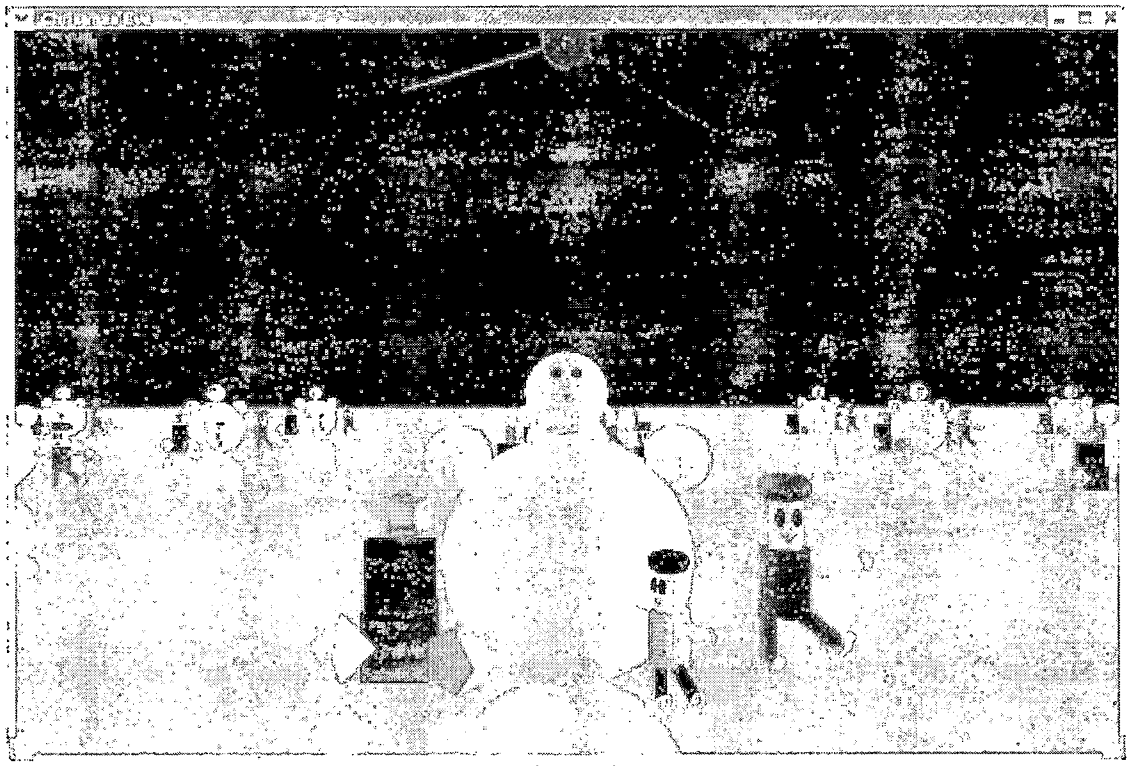


Figure 13. Screenshot of Animation

The camera angle is controlled by arrow keys of key board which uses function gluLookAt(). The above is a screenshot of all snowmen by moving cameras far away from the scene.

A display list is a group of OpenGL commands that have been stored for later execution. When a display list is invoked, the commands in it are executed in the order in which they were issued. One common use is creating a display list for an object that is to be drawn more than once. I have used a display list for creating these snowmen with all those figures. Because I am drawing them again and again it gave me a better frame rate and looks better.

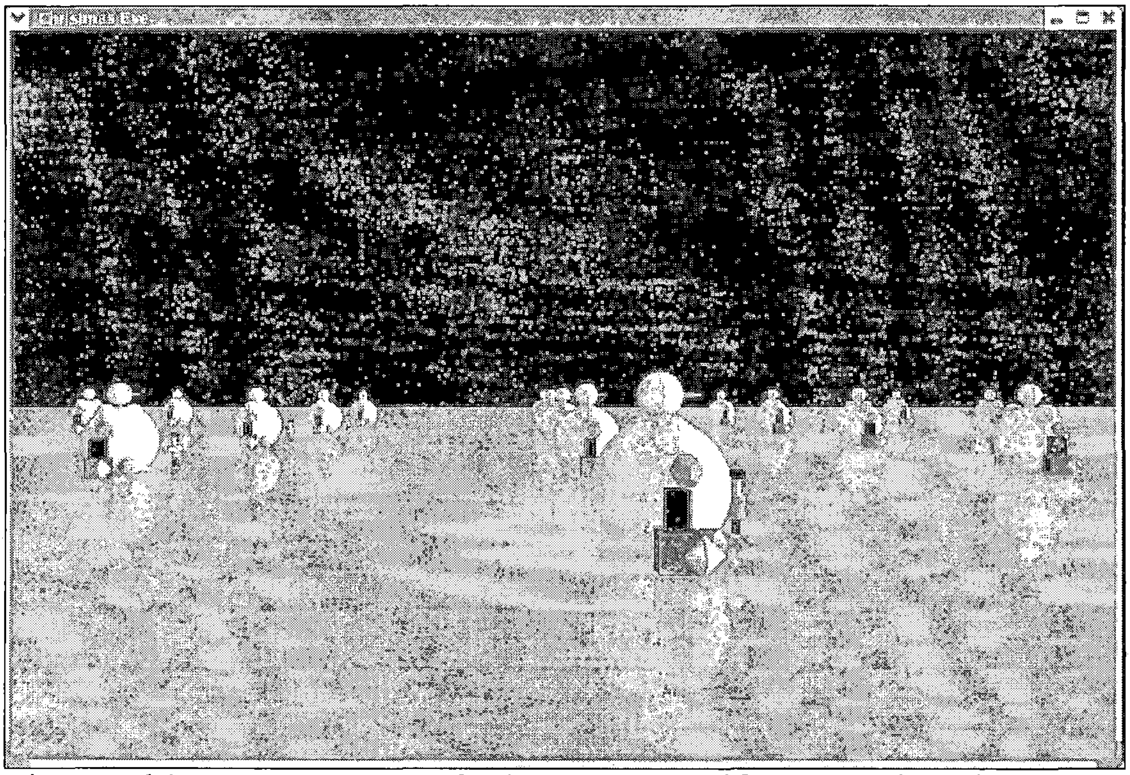


Figure 14. Screenshot of the Scene while Rotating in Y-Direction

You can also rotate the entire scene around X, Y, & Z axis by using keys x, y, & z respectively. This is handled by using glRotatef() function. The camera can also be moved when you are rotating the whole scene. This gives a very good effect to the scene.

Adventures of Smiley

After these two small projects I became so interested in graphics and want to learn further 3D modeling and some other programming tool. As my other field of interest is Web design and development, I thought of choosing some good tool where I can design and develop games as well as publish on the Internet. That experience with those kinds of tools is definitely going to help when designing web content. I am going to explain how the game "Adventures of Smiley" is developed from scratch to finish in the following sections.

Game Idea

The basic idea of the game is to help children learning in a fun and interactive way. The first version of the game idea is to find a key in a room by answering questions to get to the next room and the user has to pass a series of rooms to get to the destination room. Each room has different features and the user is faced with simple AI games, tricky questions, and hints. For example, a room may have different wall colors, carpet, lighting and the character advances and explores what's behind the objects. The user can view characters from all angles. The characters will be developed in 3D Studio Max and later imported to the project using some tools. The characters are planned to be funny looking with minimal movements. So, after the user finds the answer to the question, a key will appear. The user must approach the key in order to take it into his or her hand. The user uses the key to unlock the door, and goes into the next room.

For preschool children the game/questions inside the room can be like:

1. Identify missing letters.
2. Choose the right shape from a series of shapes after you hear that name of the shape.
3. Memory game: show an object and in the next screen gave a set of objects and let the user to choose the right one that he/she saw before. For example show an apple, and give a set of fruits.
4. Counting objects like flowers, puppies, and other objects that children usually like.

Then when the game starts a series of rooms is generated based on how many questions were there in that file. Each room will have randomly placed objects, furniture and other things that children can explore. The hint option is also there, but if the user requests the hint, he/she is going to lose points.

The whole game idea above is written in the idea of using Game engines as the basic tool. But after some research as I explained in the previous chapter I shifted to Director. This selection of Director has some advantages as well as some disadvantages but for my time frame this was the best tool to go with.

3D Modeling Using 3D Studio Max

The first phase of the project is to model some 3D objects, main character of the project. After some research I decided to do a smiley on skating wheels. The first reason I choose smiley is because everybody loves smiley, and the second reason is the limitations of Director. I will explain that in detail in the section "problems faced."

About 3D Studio Max

On 2006 Oct, Wikipedia has an entry describing 3D Studio Max:

> 3D Studio Max is one of the most widely-used off the shelf 3D animation programs. It has strong modeling capabilities, a ubiquitous plug-in architecture and a

long heritage on the Microsoft Windows platform. It is mostly used by video game developers but can also be used for pre-rendered productions such as movies, special effects and architectural presentations [12].

Polygon modeling is more common with game design than any other modeling technique as the very specific control over individual polygons allows for extreme optimization. Also, it is relatively faster to calculate in real-time. Usually, the modeler begins with one of the 3Ds max primitives, and using such tools as bevel, extrude, and polygon cut, adds detail to and refines the model. Versions 6 and up feature the Editable Poly object, which simplifies most mesh editing operations, and provides subdivision smoothing at customizable levels.

Smiley Modeling

The first step of modeling smiley is to have a picture of smiley ready. So, I drew a picture with both side view and front view. Though I am almost perfect with the drawing both views didn't match exactly and it may create problems while modeling. Then, I used Photoshop CS to edit the picture and achieved the similarity. Following figures show the smiley drawings, before and after editing in Photoshop.

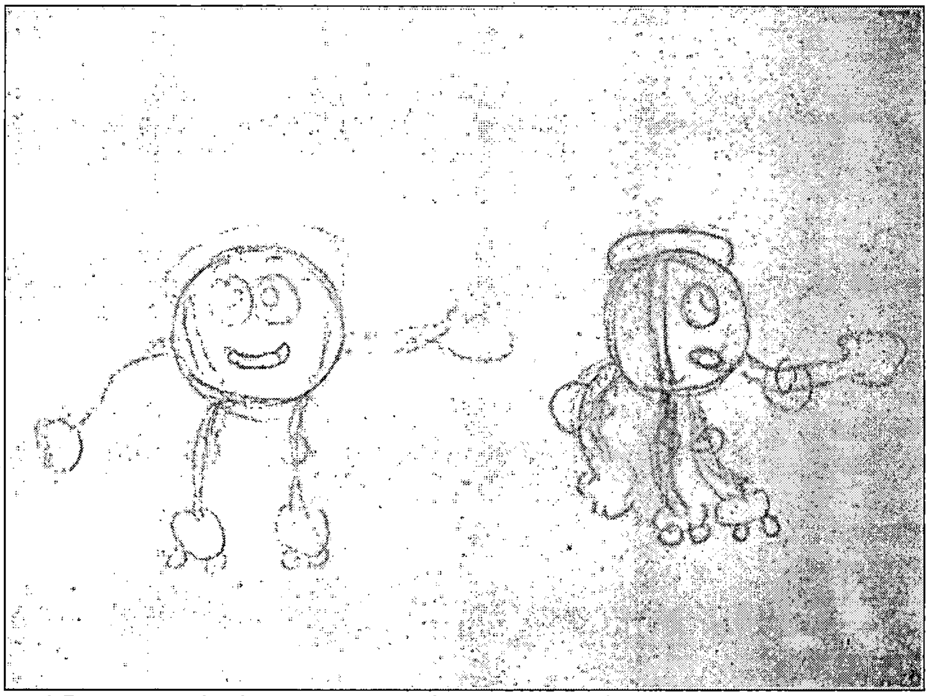


Figure 15. Preliminary Drawing of Smiley

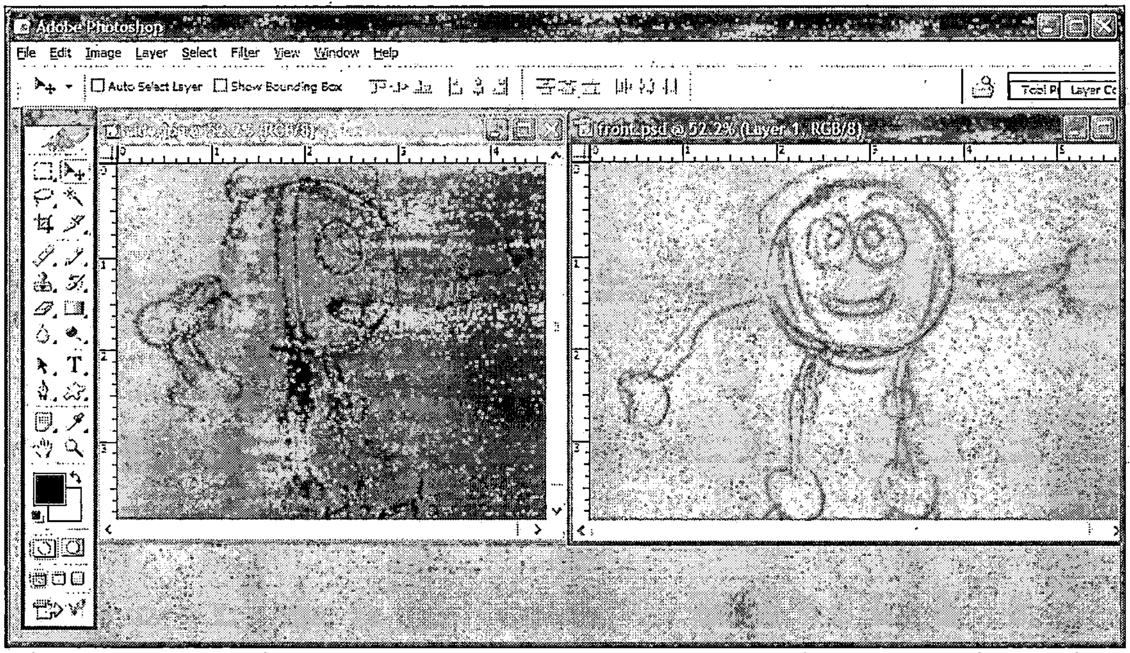

Adobe Photoshop
File Edit Image Layer Select Filter View Window Help
Auto Select Layer
Show Bounding Box
front.psd @ 52.2% (Layer 1, RGB/8)

Figure 16. Edited Drawings to Match the Size of Smiley in Both Views

The next step is to import them into the 3D Studio Max and create smiley using the pictures as references in all view ports. The picture of finished smiley along with matching drawings is show in the below figure.

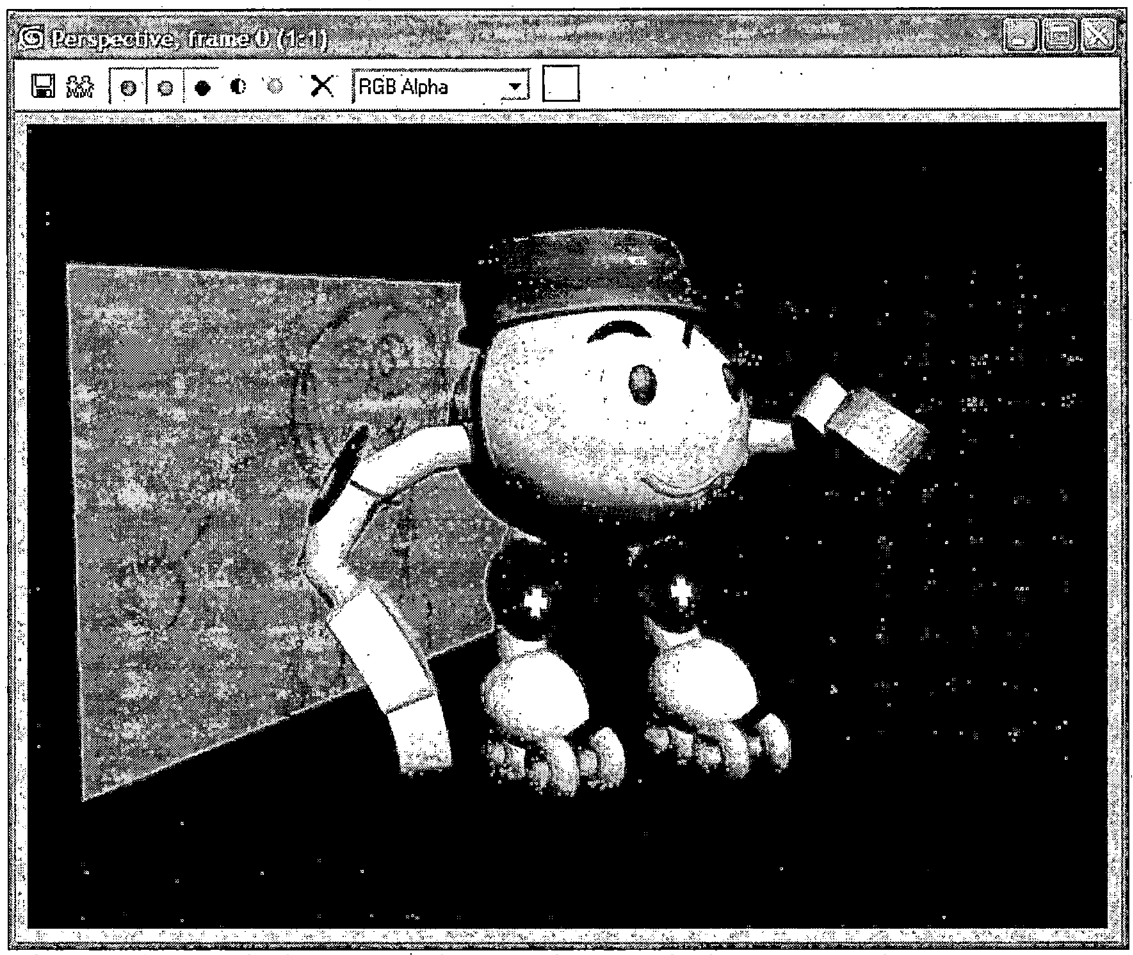
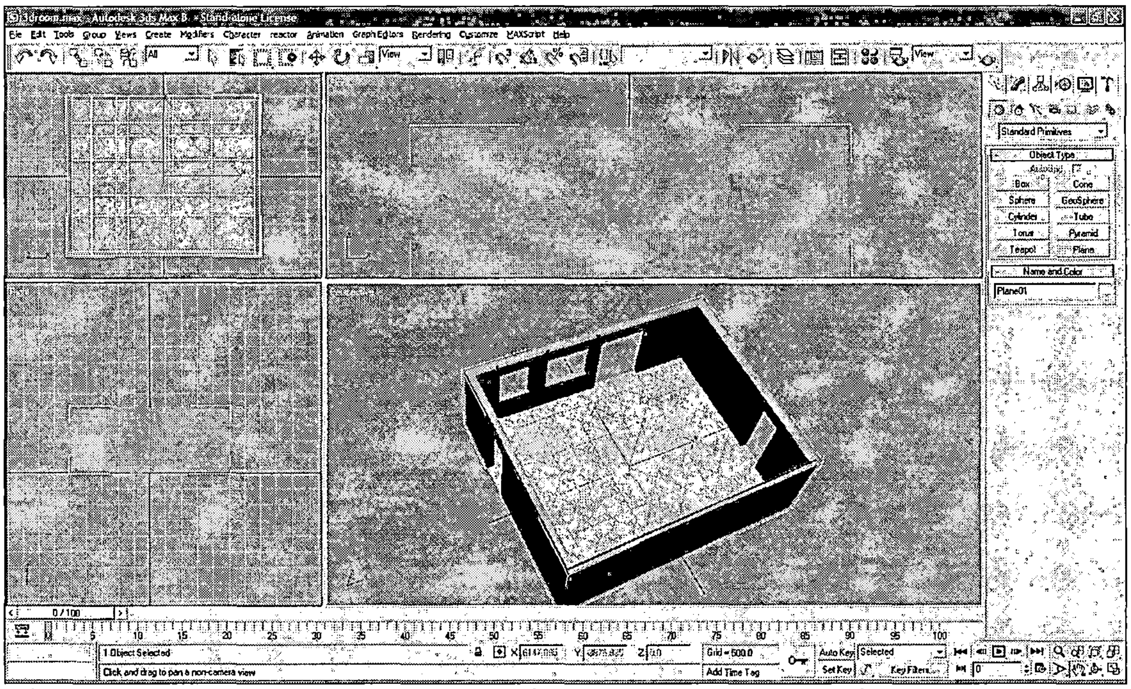


Figure 17. Finished Smiley with Original Drawings

The next important thing in the project is the room. I created many sample rooms and tried to export them into Director and see how they were looking in the final version. Here are some of the sample rooms I have created using 3D Studio Max.


3droom.max - Autodesk 3ds Max 8 - Stand-alone License
File Edit Tools Group Views Create Modifiers Character reactor Animation Graph Editors Rendering Customize MAXScript Help
Standard Primitives
Object Type
AutoGrid
Box
Cone
Sphere
GeoSphere
Cylinder
Tube
Torus
Pyramid
Teapot
Plane
Name and Color
Plane01
0 / 100
1 Object Selected
Click and drag to pan a non-camera view
Grid = 500.0
Auto Key
Selected
Set Key
Key Filters...
Add Time Tag
Figure 18. Sample 3D Room with Polygon Modeling

Next 2 pictures show two kinds of rooms I developed, but as I tried exporting them into the Director, the size of the final movie became so big and I thought I should combine both of them and show the sub model when necessary, otherwise hide them.

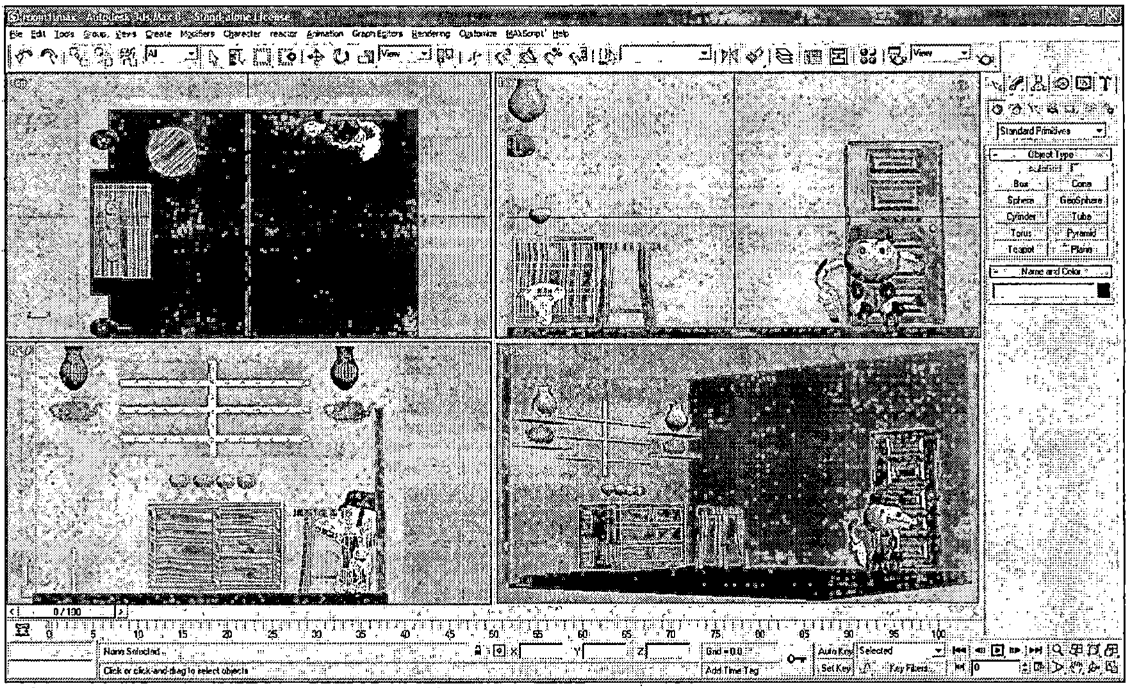


Figure 19. Sample Furnished 3D Room 1

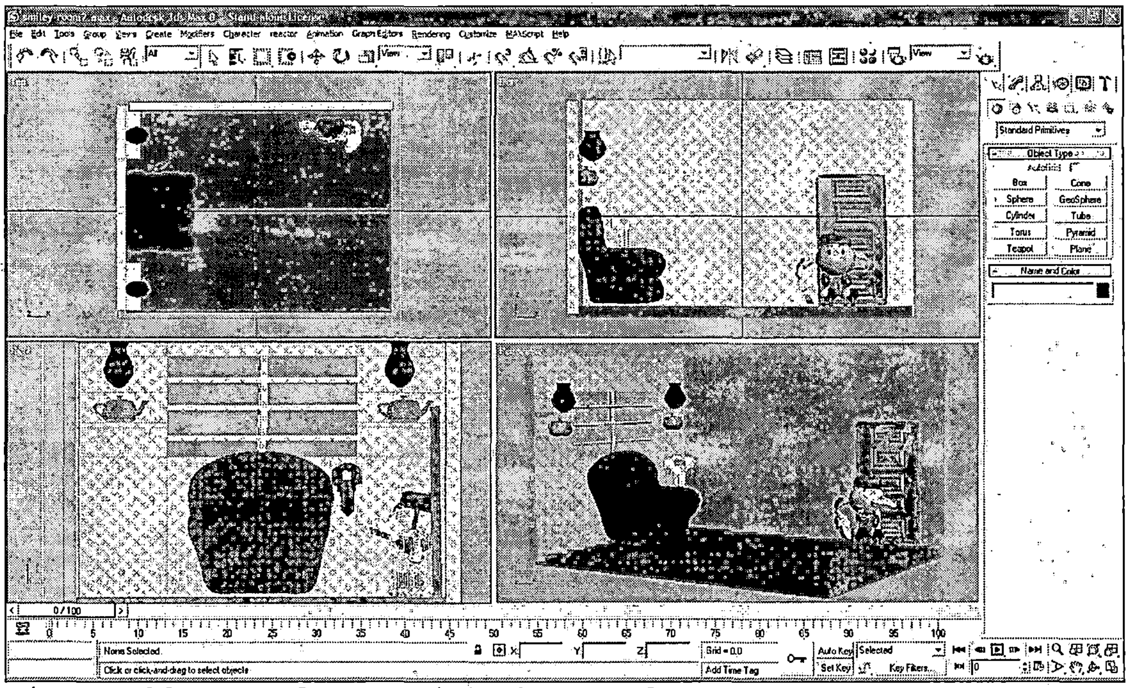


Figure 20. Sample Furnished Room 2

The next thing I have done here is the key, which is also very important in the project. I used the best feature of 3D Studio Max, polygon modeling and generated the key based on a picture.

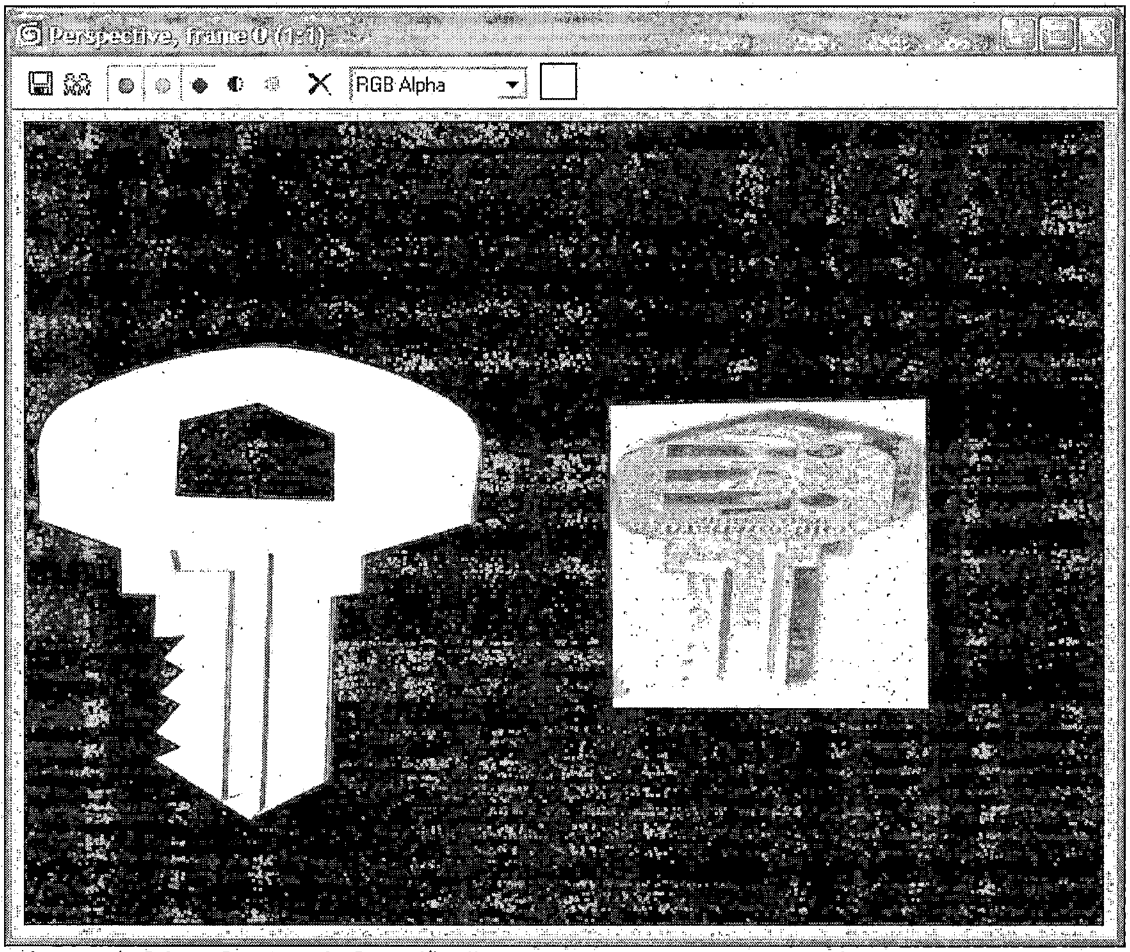


Figure 21. 3D Model of the Key Based on a Picture

I used the polygon modeling for the 3Dmaze too. For this also I used a maze picture and modeled based on it.

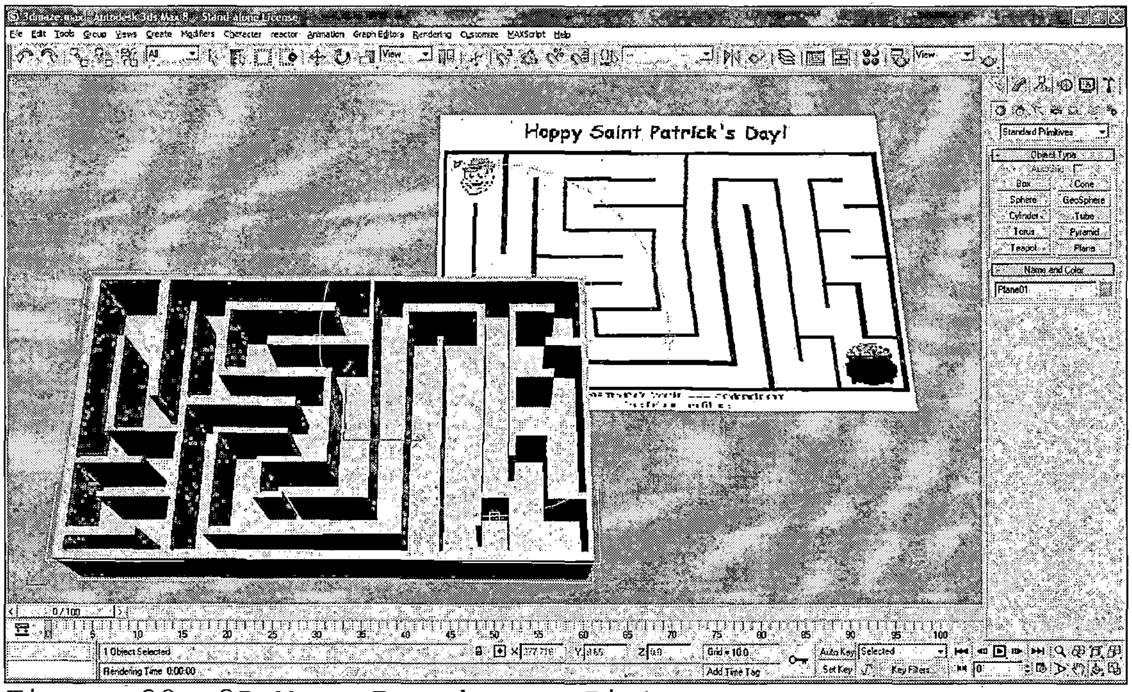


Figure 22. 3D Maze Based on a Picture

I followed wonderful video tutorials by 3Dbuzz.com and got a good picture of the tools inside it, and what is the best to use for which condition etc [2]. I am able to achieve smiley animation like walking but unable to export into Director due to the bone animation limitations. The other limitation I faced here is the problem with textures. I modeled a table with racks and used beautiful texture in there, but Director is not able to accept the texture and gave me a plain 3D model. This limited me from doing so many things in 3D Studio Max as I have to think in terms of Director.

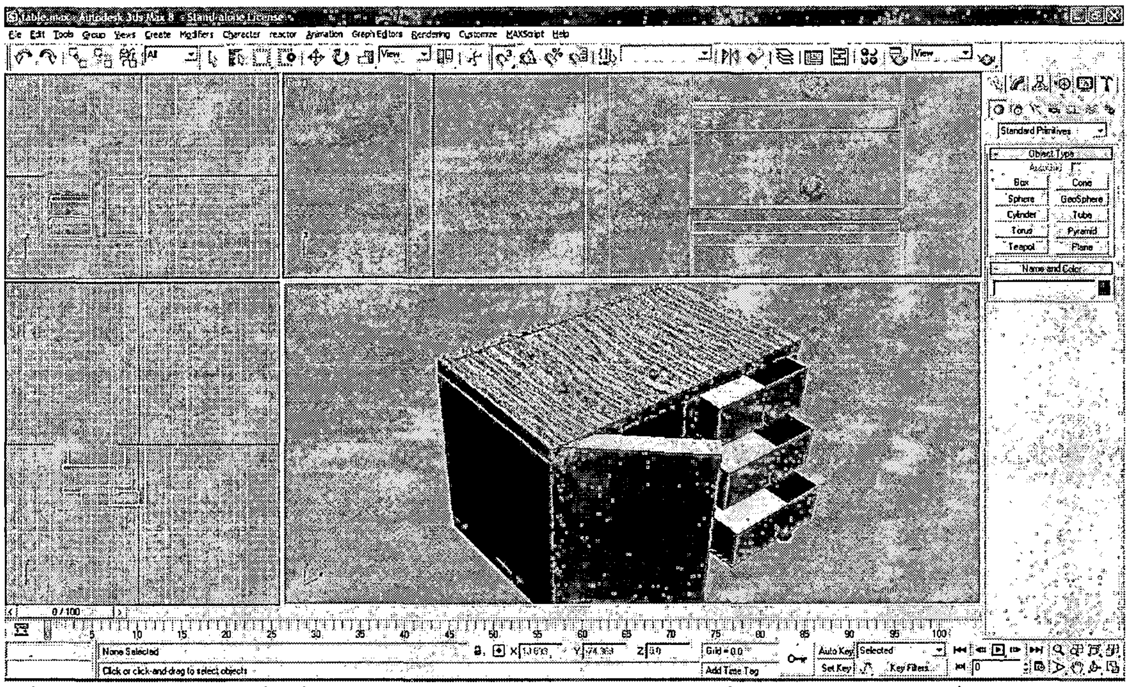
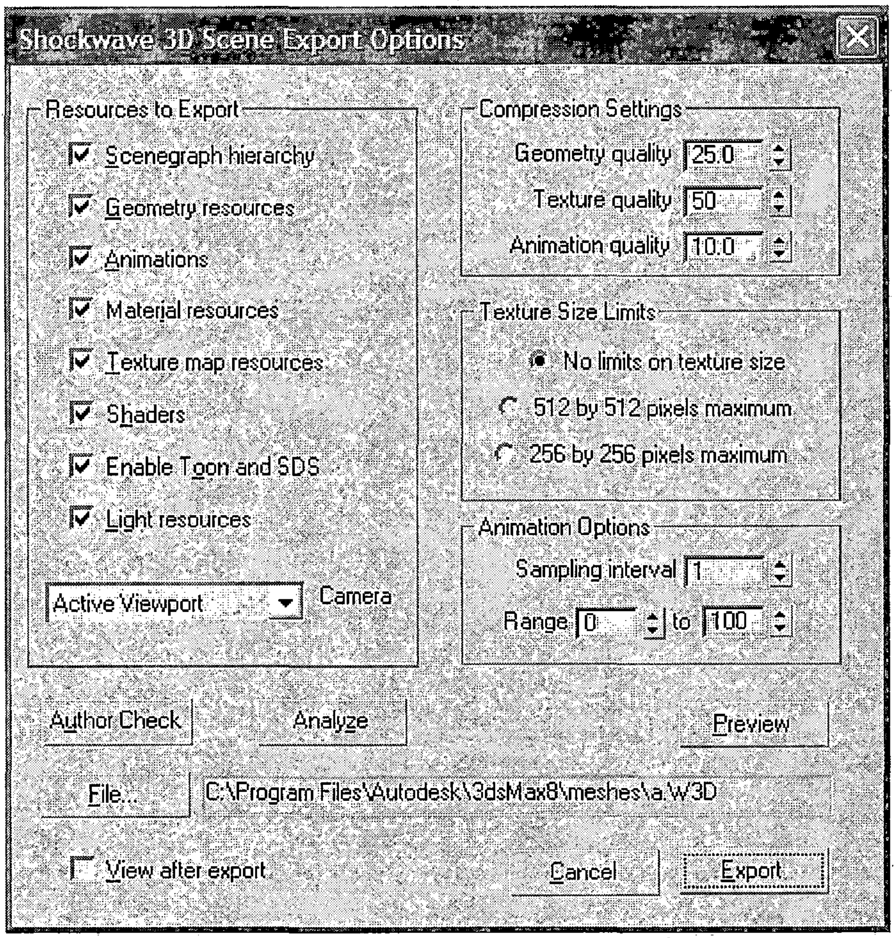


Table.max - Autodesk 3ds Max 8 - Stand-alone License
File Edit Tools Group Views Create Modifiers Character reactor Animation Graph Editors Rendering Customize MAXScript Help
Standard Primitives
Object Type
AutoGrid
Box
Cone
Sphere
GeoSphere
Cylinder
Tube
Torus
Pyramid
Teapot
Plane
Name and Color
0 / 100
None Selected
Click or click-and-drag to select objects
Grid = 0.0
Add Time Tag
Auto Key
Selected
Set Key
Key Filters...
Figure 23. Original Table Screenshot from 3D Studio Max

These are some of the models I have created using 3D Studio Max and the other models can be seen in the screenshots of the Director game in the following sections.

Exporting Models from 3D Studio Max to Director

3D Studio Max has a plug-in called Shockwave 3D Scene Exporter, W3D which is a valid format for Director to import the 3D Model into it. 3D Studio Max also gives some options whether to include all of the original shaders, materials, animation etc. Usually I selected all of them to get the best result, but the draw back is the increase in size.


Shockwave 3D Scene Export Options

Resources to Export
- [✓] Scenegraph hierarchy
- [✓] Geometry resources
- [✓] Animations
- [✓] Material resources
- [✓] Texture map resources
- [✓] Shaders
- [✓] Enable Toon and SDS
- [✓] Light resources

Active Viewport Camera

Compression Settings
- Geometry quality 25.0
- Texture quality 50
- Animation quality 10.0

Texture Size Limits
- ● No limits on texture size
- ○ 512 by 512 pixels maximum
- ○ 256 by 256 pixels maximum

Animation Options
- Sampling interval 1
- Range 0 to 100

File... C:\Program Files\Autodesk\3dsMax8\meshes\a.W3D

- [ ] View after export

Figure 24. Shockwave 3D Scene Export Options

Sound Editing in Audacity

One of the important parts of the project is reading comprehension. Kids should read the whole comprehension and answer the questions based on what they read. But, if they just have to read the whole paragraphs, the game becomes boring for them. So I thought of adding a voice to the comprehension and they will listen and follow sentence by sentence in the comprehension. That should help them to get concentration in listening, accent understanding etc.

to record and edit the sound I choose the best known open source software Audacity [15].

On aug, 2006, the developing team of Audacity has a brief explanation about the tool as below:

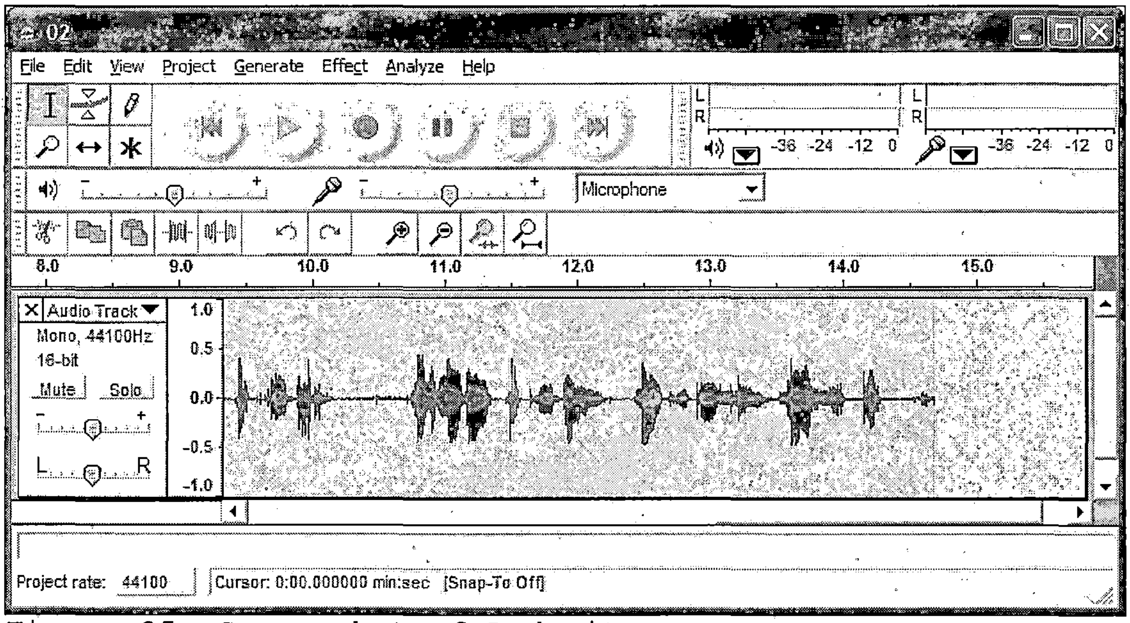
> Audacity is a free, Open Source, cross platform digital audio editor. The source code for Audacity is released under the GNU General Public License. The graphical user interface for the editor has been produced using the wxWidgets library Audacity is extremely popular in the podcasting world due to its wide availability, multiplatform support, and the fact that it is free.

The developing team of Audacity has listed all good features in their webiste[15]. Here are some of the best features:

- Importing and exporting WAV, MP3 (via the LAME MP3 Encoder, downloaded separately), Ogg Vorbis, and other file formats
- Recording and playing sounds
- Editing via Cut, Copy, Paste (with unlimited Undo)
- Multi-track mixing
- Digital effects and effect plug-ins.
- Additional effects can be written with Nyquist

- Amplitude envelope editing
- Noise removal
- Support for multichannel modes with sampling rates up to 96 kHz with 24 bits per sample
- The ability to make precise adjustments to the audio’s speed, while maintaining pitch, in order to synchronize it with video, run for the right length of time, etc. Unlike many other programs, Audacity has very few audible artifacts (doubling, r chorus type effects) when lengthening or shortening a file.
- High ease of use
- Large array of plug-ins available

The screenshot of Audacity while editing a voice record for a reading comprehension on Gandhiji is given below.


02
File Edit View Project Generate Effect Analyze Help
Microphone
Audio Track
Mono, 44100Hz
16-bit
Mute Solo
Project rate: 44100
Cursor: 0:00.000000 min:sec [Snap-To Off]
Figure 25. Screenshot of Audacity

Game Development in Director

I followed great tutorials by Dean Utian for learning Director and 3D lingo [1]. First, we have to import 3D models and any sounds if necessary. Then we have to makeup all 3D models in the Director and use the 3D lingo to find a model. Please note that lines prefixed with -- are comments in Lingo.

```
-- initiate properties and 3D world
pSprite = sprite(me.spriteNum)
p3Dmember = sprite(me.spriteNum).member
p3Dmember = resetWorld()
pCharacter = p3Dmember.model("smiley")
pKey = p3Dmember.model("key")
```

The above code is used to initialize the 3D scene along with 3D models in it. It is usually coded in the behavior of the 3D scene sprite. The character in the project is smiley, which is the name of the smiley 3D model in the 3D scene developed by 3D Studio Max. Now, both the key and the character were referred with variable names pCharacter and pKey.

<u>Character Moment</u>

Then, the moment of the smiley comes into play. Here, I coded the moment based on the arrow keys as most people use that for usual moment of anything in a game.

```
On keyDown
-- check to see which key has been pressed
-- and set the property relating to that key to TRUE
--- 123 = left arrow key
If keypressed(123) then pLeftArrow = TRUE
--- 124 = right arrow key
If keypressed(124) then pRightArrow = TRUE
--- 125 = down arrow key
If keypressed(125) then pDownArrow = TRUE
--- 126 = up arrow key
If keypressed(126) then pUpArrow = TRUE
end
```

The above script recognizes key presses and validates the true/false evaluates based on them. Then the character us moved based on pixel translation. The speed of the character depends in the pixel value.

```
On characterMove
-- if right arrow is pressed,
-- if rotate the character 5 degrees about the z-axis
If pRightArrow then pCharacter.rotate(0,0,-5)
-- if left arrow is pressed,
-- if rotate the character -5 degrees about the
z-axis
If pLeftArrow then pCharacter.rotate(0,0,5)
-- if the up arrow is pressed,
-- move the character 5 pixels along the y-axis
If pUpArrow then pCharacter.translate(0,5,0)
-- if the down arrow is pressed,
-- move the character -5 pixels along the y-axis
If pDownArrow then pCharacter.translate(0,-5,0)
end
```

The major part this whole project is collision detection. As we have different models in the 3D scene of room, we have to find the meshes of the every model and see whether the collision detection occurs or not. In order to find the collision detection, Director provides 2 ways to do: Collision modifier and model under ray.

Collision Detection Using Collision Modifier

```
On createCollisionDetect
-- add collision modifier to the "wall"
p3Dmember.model("wall").addModifier(#collision)
-- set bounding geometry for collision detection to
bounding box of model
p3Dmember.model("wall").collision.mode = #box
-- resolve collision for wall
p3Dmember.model("wall").collision.resolve = TRUE
-- add collision modifier to the character
pCharacter.addmodifier(#collision)
-- set bounding geometry for collision detection to
bounding box of model
pCharacter.collision.mode = #mesh
-- resolve collision for character
pCharacter.collision.resolve = TRUE
end
```

By attaching the collision modifier to objects in a 3D world, we can detect collisions with all the objects to which the modifier has been applied. Once the collision modifier has been attached to a model, the following collision modifier properties are accessible (This is taken from Director's Help):

- enabled (collision) allows you to turn off the collision detection (by setting the value to FALSE) while the modifier is applied. The default value is TRUE.
- resolve indicates whether collisions models stop when they collide with each other. If the resolve is set to FALSE, then that model will continue to move. The default value is TRUE
- immovable indicates whether a model can be moved as a result of collisions. This default value is FALSE.
- mode (collision) indicates the geometry used for collision detection and could be:
- #mesh uses the actual mesh geometry of the model's resource. It is more precise but usually slower than #box or #sphere.
- #box uses the bounding box of the model. It is useful for objects that can fit more tightly in a box than in a sphere, such as our wall.
- #sphere is the fastest mode, because it uses the bounding sphere of the model. It is the default value for this property.

The use of the collision modifier is useful for relatively simple applications only. It can be very processor intensive, especially if there are complex models and a large number of modifiers attached to objects. The recommended technique for collision detection of more complex models is to use the models UnderRay command. Because of these limitations I faced some problems in the movement of smiley. The model behaves weird due to the bounding box of the model mesh. This method is also tiresome as I need to add collision modifiers to each and every model in the scene and will finally became more process intensive. So, I thought of going for models under ray.

Collision Detection Using Models Under Ray

In order to use this feature, we need to have a bounding box of some of the models in the 3D scene, So I created bounding box for smiley and key as I need to know when the collision happened.

```
-- create model resource
charModRes = 3Dmember.newModelresource("charModRes",
#box)
charModRes.height = 60
charModRes.width = 70
charModRes.length = 90
```

```
-- create model and reference to it
pCharBoundingBox= p3Dmember.newModel("CharBoundingBox",
charModRes)
pCharBoundingBox.worldPosition =
pCharacter.worldPosition
pCharacter.addChild(pCharBoundingBox, #preserveworld)
boxShader=p3Dmember.newShader("transparentShad",#standard)
boxShader.transparent = TRUE
-- set opacity of shader 0 is invisible - only works if
transparent prop = TRUE
boxShader.blend = 0
pCharBoundingBox.shaderList = p3Dmember.shader
("transparentShad")
```

In the above script we created a bounding box for the character with its position corresponding to the character model. The box was made a child of the character so it would always move with it. A shader was created with the *transparent* property on (TRUE). This allows the *blend* property to be set to less than 100%. Setting the *blend* to 0 will make the model invisible.

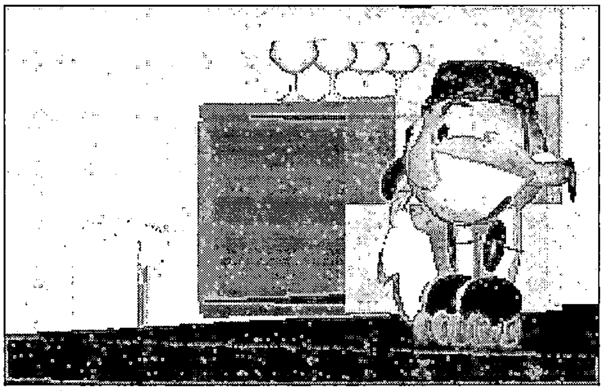


Figure 26. Bounding Box of Models when the Blend=80

```
on collisionDetect me
-- create a list to store collision data created by
modelsUnderRay
-- cast ray to left
collisionList = p3Dmember.modelsUnderRay
(pCharBoundingBox.worldPosition,\
```

In the above statement, we create a list (*collisionList*), to hold the information generated by the *modelsUnderRay* command. *#detailed* is used for the *levelOfDetail* parameter and will return a list of property lists, each representing an intersected model. *#distance* is one of the properties that will appear on the property list, which, in our case, represents the distance from the character to the point of intersection with the model.

```
-- if ray picks up models, then activate checkForCollion
-- handler and send the collisionList as a parameter
if (collisionList.count) then
```

```
me.CheckForCollision(collisionList[1])
if (collisionList[1].model = pkey) and gmatch=1 or 2
then me.checkObjectFoundDistance(collisionList[1])
end if
-- cast ray to right
collisionList = p3Dmember.modelsUnderRay
(pCharBoundingBox.worldPosition, \
pCharacter.transform.yAxis,#detailed)
-- if ray picks up models, then check for collision
if (collisionList.count) then me.checkForCollision
(collisionList[1])
-- cast ray forward
collisionList = p3Dmember.modelsUnderRay
(pCharBoundingBox.worldPosition, \
pCharacter.transform.xAxis,#detailed)
-- if ray picks up models, then check for collision
if (collisionList.count) then
me.checkForCollision(collisionList[1])
-- cast ray backward
collisionList = p3Dmember.modelsUnderRay
(pCharBoundingBox.worldPosition, \
-- if ray picks up models, then check for collision
if (collisionList.count) then me.checkForCollision
(collisionList[1])
end
```

This handler is activated when a model is picked up by *modelsUnderRay*. So, *collisionList[1]* is assigned as a value to the parameter *thisData*

```
on checkForCollision me, collisData
-- grab the #distance value from the CollisionList
dist = collisData.distance
-- check if distance is less than width of bounding box
if (dist < pCharBoundingBox.resource.width/2) then
-- get distance of penetration
diff = pCharBoundingBox.resource.width/2 - dist
-- calculate vector perpendicular to the wall's surface
to move the character
-- using the #isectNormal property
tVector = collisData.isectNormal * diff
-- move the character in order to resolve the collision
pCharacter.translate(tVector,#world)
end if
end
```

So, when there is a collision between the key and character it calls the below function and based on the proximity we can specific things here.

```
on checkObjectFoundDistance me, thisData
dist = (thisData.distance)-15 -- 15 is to increase the
proxmity of the contact point with the key
```

```
if (dist <5) then
-- the contact happened and the code related to that
goes here
end if
end
```

Linking All Movies

One if the best feature of Director is that it supports passing values between different Director movies. The biggest problem I faced in the project is size of the movies. As the 3D scene is with lot of polygons, though I manages to make most of the models with low polygon count it affected the size of the movie and the time to download on web. This also limits me from doing so many things like exploring the whole room by the kid etc.

Importing a cast member makes a copy of the file being imported and brings the entire set of data for that castmember into the movie. As a result, the movie becomes correspondingly larger, and any changes made to the original file have no effect on the castmember in the movie. Linking a cast member does not make a copy of the file being linked. All that is brought into the movie is a reference to the file's location on your file system. The movie does not become significantly larger, and any changes made to the original file are reflected in the castmember in the movie.

Each method has its advantages and disadvantages. Many Director movies benefit from a hybrid approach - some of the cast may be imported, some may be linked. It is not only possible, but probably a very good idea to make the decision on a cast-by-cast basis. This is the method I followed throughout the project.

```
global gmatch
on mouseUp me
gotoNetMovie
"http://vanisworks.com/Documents/matching.dcr"
end
```

I used a global called gmatch and once it is declared in the script of all the movies which are linked, the values remains the same even If you switch between movies. This helped me to know whether the use finished the game in the matching or not by returning a value of 1 to the original movie.

<u>Director Interface</u>

The Director interface is easy to get used if you have some prior experience with Flash or Photoshop. This helped me to learn the tool quickly. The following figure shows the interface of Director movie start room.

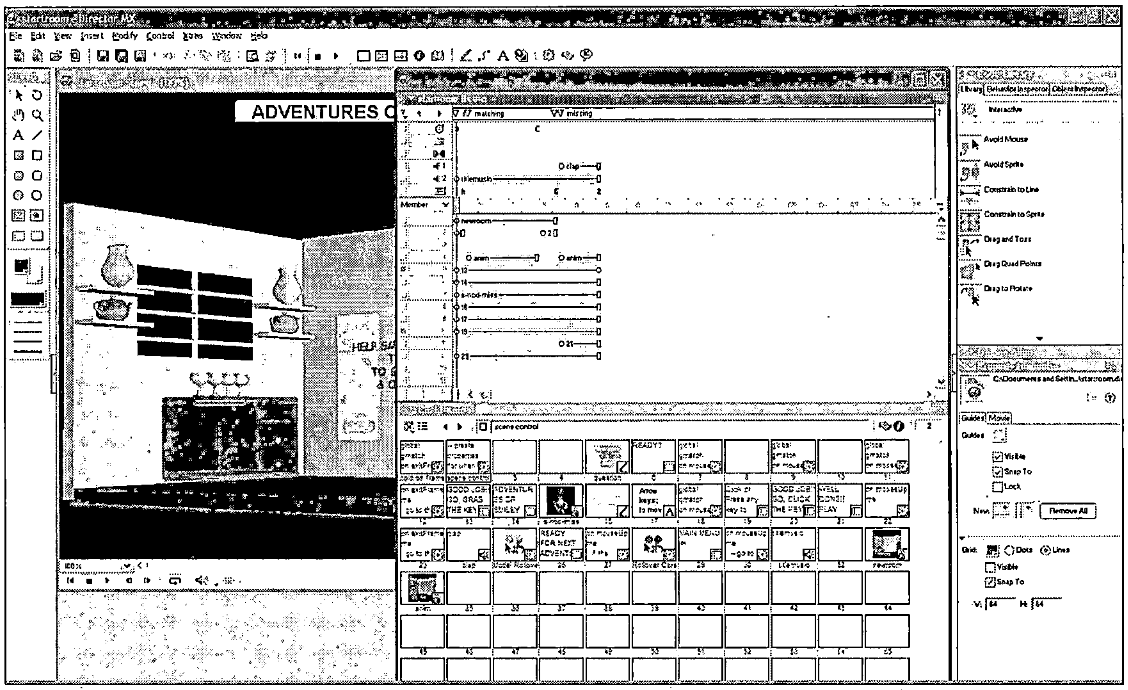


Figure 27. User Interface of Flash Player in Director

Flash the Player in Director

I thought of having the reading comprehensions displayed on a big screen TV, and let the user see in the 3D room itself. I was able to achieve this but it complicate things like user navigation and selection of contents from the screen. The main drawback of this effect is enormous increase in size of the resulting movie. The following screenshot shows a testing movie I have done to play a Flash the player on the walls of the 3D model.

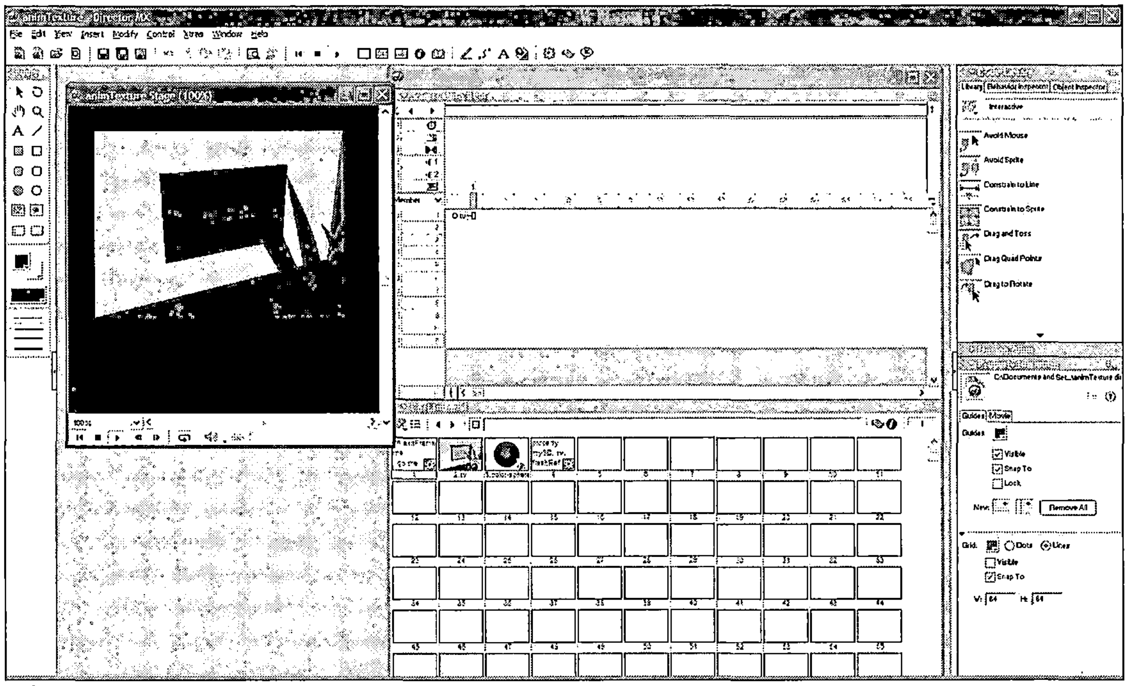


Figure 28. Screenshot of Flash the Player on the Walls of the 3D Model

```
my3D = sprite(me.spriteNum).member
tv = my3D.model("screen")
FLASHRef = member("color-sphere")
FLASHImg = FLASHRef.image
frameCnt = FLASHRef.frameCount
currFrame = 1
texRef = my3D.newTexture("tv", #fromImageObject,
FLASHImg)
tv.shader.texture = texRef
```

this is done by defining the Flash animation as a texture and applied to the 3D model screen.

Digital Video in the 3D Model

In the same path as Flash in the 3D model, I was also able to play a video in real media format on the walls of the model.

```
--Create a box model and assign it to the model resource
and shader created above
pRealCube = p3DMember.newModel("real_cube",tCubeRez)
pRealCube.shaderList = tShader
pRealCube.translate(15,125,15)
--Create a texture object using the RealMedia cast
member.
--You can use this technique with any texture created in
Lingo. You could replace
--#fromCastmember with #fromImageObject and replace
--member("RealMedia") with
--member("RealMedia").image.
pRealTexture = p3DMember.newTexture("real_texture",
#fromCastMember,member("sLions"))
pRealTexture.quality = #high
---Assign the texture to the model's shader
pRealCube.shader.texture = pRealTexture
---Initiate playback of the RealMedia cast member. This
step is necessary only if
--you are using a RealMedia cast member.
member("sLions").play()
```

The above defines the real media file as a texture and applied on the walls of the model. Following figure shows the result of that.

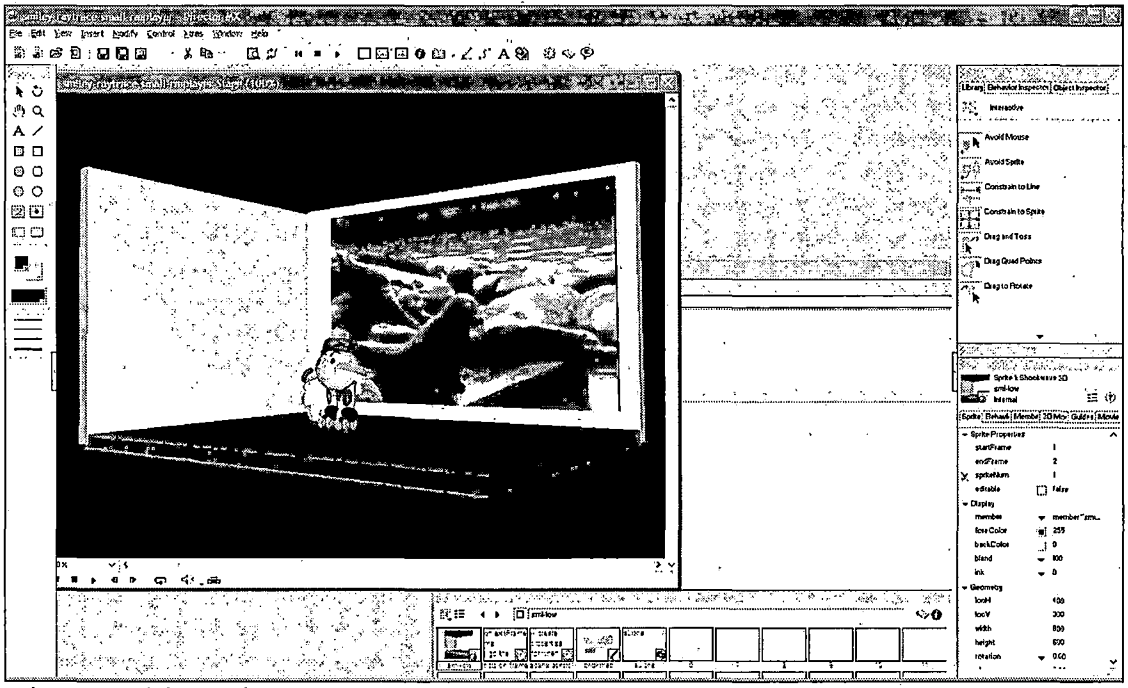
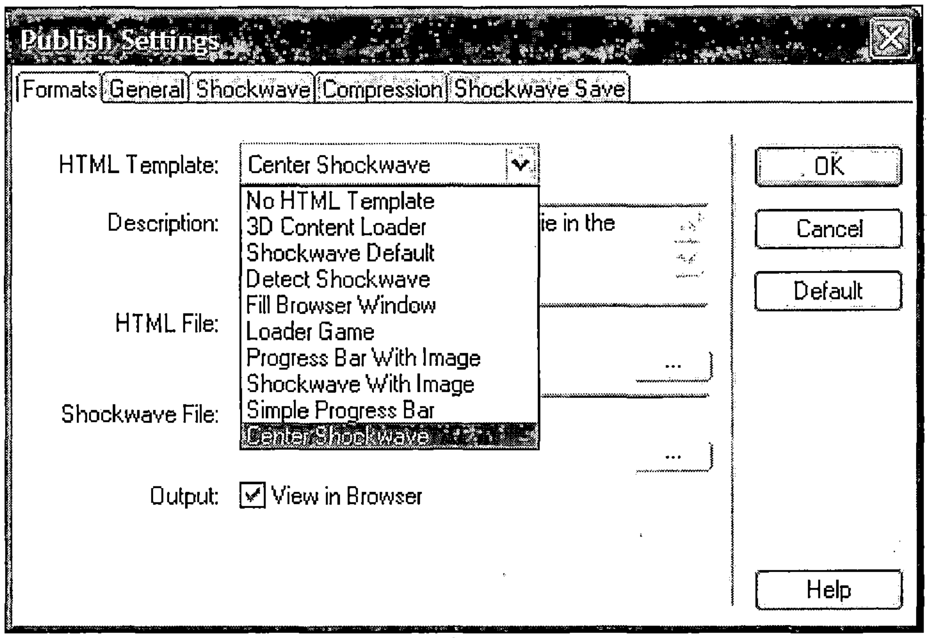


Figure 29. Video on Walls of the Screen

Publishing Project on Web

Director has a very nice feature of publishing movie on the Internet. It allows the user to decide what option they want, and how the movie should look like in the browser etc.



HTML Template: Center Shockwave

No HTML Template
3D Content Loader
Shockwave Default
Detect Shockwave
Fill Browser Window
Loader Game
Progress Bar With Image
Shockwave With Image
Simple Progress Bar
Center Shockwave

Description: ie in the

HTML File:

Shockwave File:

Output: [✓] View in Browser

Figure 30. Publish Settings in Director

For this project I choose the movie to be located in center and compressed the whole movie to save space. Director should be really appreciated in regards with compression as it almost able to compress the movie up to 60% without reducing the quality.

```
<HTML>
<HEAD>
<TITLE>main</TITLE>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</HEAD>
<BODY bgColor="#000000">
<center>
<TABLE border=0 width=100% height=100%>
```

```
<TR><TD valign=middle align=center>
<OBJECT classid="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/D
IRECTOR/sw.cab#version=8,5,1,0"
ID=main WIDTH=1000 HEIGHT=600>
<param name=src value="main.dcr">
<PARAM NAME=swStretchStyle VALUE=fill>
<param name=swRemote value="swSaveEnabled='true'
swVolume='true' swRestart='true' swPausePlay='true'
swFastForward='true' swContextMenu='true' ">
<PARAM NAME=bgColor VALUE=#000000>
<EMBED SRC="main.dcr" bgColor=#000000 WIDTH=1000 HEIGHT=600
swRemote="swSaveEnabled='true' swVolume='true'
swRestart='true' swPausePlay='true' swFastForward='true'
swContextMenu='true' " swStretchStyle=fill
TYPE="application/x-DIRECTOR"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/"><
/EMBED>
</OBJECT>
</TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

the above code is the basic html code to embed a shockwave movi in the browser. It downloads the shockwave plug-in along with any Xtras from macromedia website.

Problems Faced and Suggestions to Improve Director

The project "game design and development" is kind of whole experience to me in matter of research and learning new tools. I enjoyed every moment in the part of research. When I find some kind of tools I always used to think like, why not use this one for my project? But after some research about that I dropped from that due to unavailability of some features which I wanted.

Most of the problems I faced are due to the limitation of Director in importing 3D Studio Max models.

Texture Mapping

3D Studio Max includes many texture mapping features that are not supported in Shockwave3D. A basic summary of the restrictions is represented in the following graphic:

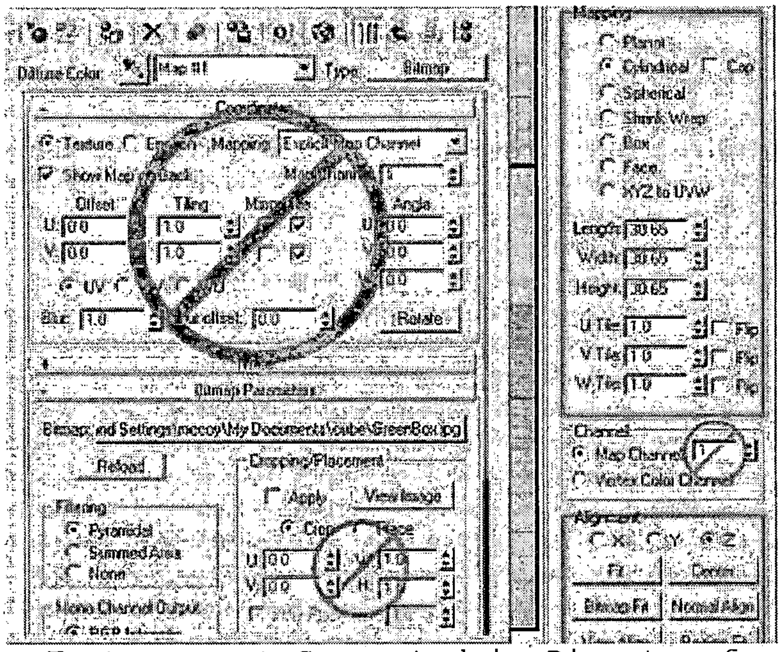


Figure 31. Features not Supported in Director from 3D Studio Max

The following items (marked with special symbols) are not supported:

- Texture coordinate adjustments within the Bitmap texture's Coordinates rollout.
- Texture coordinate adjustments within the Cropping/Placement box of the Bitmap texture's Parameters rollout.
- Texture coordinates in mapping channels other than channel one.

<u>Animation of the Character</u>

This is something though Director supports I could not do it, because by the time I came to know about it I already finished all 3D models of my project. I am able to walk my smiley in 3D Studio Max and after I export it to Shockwave nothing came out due to missing bone hierarchy. So, redoing it means start the project from scratch. Instead I made the character moving on wheels. So, not much physical moment.

Collision Detection

Collision detection is the big part of the project and took me long time to find the right way of collision detection in my project. I tried to use Collision modifiers, but lack of support of that modifier to every model in the 3D scene and complex mesh of smiley made the collision detection weird. So, I decided to go with model under ray after wasting considerable amount of time. Collision modifier is one of the feature that macromedia should change in Director as it is smooch processor intensive and even with a dual core machine and 1GB of RAM is not enough to get the right effect. So, there is not thought of single core and less RAM. Better they should get rid off the feature in order to not confuse users from using it and facing troubles.

Video Size

The export to video file like quick time, windows movie audio/video format is the funny thing in Director.

Since, once you start rendering the video you are not supposed to go to other browser and stay in the Director as much time it takes. Otherwise, you will get the screenshots of the other activities you go into. Even if we follow strict rules the size of the resulting video became too big. This feature need to be updated and this is not just mine but a lot of Director users online.

3D Model Importer

As macromedia Director didn't see much updates after 2003, it is really old if they consider updating the importer based on the current technology. No doubt, that Director is the best tool available to play 3D games online, still it does need some fixes in it.

Sudden Crashes of Director

For this, I am not sure whether to blame windows operating system or Director. It usually happened to me when I am trying to link different movies and updating models in my 3D scene. If this happened, it asks you whether to save the file or not, if you save, you are going to corrupt the original file. After facing this problem for few times, I started having backups always before I start adding a new feature to the movie. Though it ate my disk space, a whopping amount of 500 MB (just for Director movies), I continued to do it.

CHAPTER FOUR

CONCLUSION AND FUTURE DIRECTIONS

Conclusion

This project on game design and development was completed to get hands-on-experience with present day technology used in games. Most of the project involved research about which tools suit particular game genres, and how to use those tools to develop 3D games.

As part of learning these tools, a 3D adventure game was developed using Macromedia Director MX and 3D Studio Max. The purpose of the game "Adventures of Smiley" is to involve children with a fun way of learning using a friendly interface. This game can be extended to any age group based on the difficulty level of lessons, puzzles, etc. Educational games were chosen, because 90% of the age group spend considerable amount of time playing online and video games. Attractive educational games with good sound effects can encourage children to learn while playing. The purpose of these games is to teach skills that employers want: analytical thinking, team building, multitasking and problem-solving under duress. The significance of the project is to involve children with a fun way of learning by using a friendly interface and to improve their concentration level by presenting lessons and articles in an attractive way. The research carried out throughout this project gathered a lot of information about the current game industry, technologies, and game genres and can be used as a reference to the beginner level game programmer.

Future Directions

This project is a good start for a novice programmer who can read the research carried throughout the project to get to know what is what in the current game industry. The resulting documentation can also help developers to choose the right tool for the game idea he/she has. The online documents can be updated with further information regarding the gaming industry as time goes. So far this project has only limited games for 2 age groups. This project can be extended to any age group of students by making minimal changes in the coding. For example, the project can be extended to preschoolers to have some games on finding colors, pronunciation, memorizing, etc. There is no limit to the potential games that can be produced in this way. The code is provided with good comments and is intended to be understood by novice programmer. For example, a 3D maze can be extended to become any kind of maze, and can be used to form a series of levels that the user passes through while playing the game. The only thing that needs to change is the 3D maze in the 3D Studio Max given with condition that the names of the model should be kept the same. The developer can make these improvements with new 3D rooms for every level of the game and exploring the 3D room by the user. This can be done when you distribute the game on CD using a shockwave projector where size is not a limit. For now, the 3D content and exploration kept simple for online use.

APPENDIX A

PROJECT RESOURCES

PROJECT RESOURCES

The permanent URL to find the project and get the game online is http://vanisworks.com/project.aspx. You can play the game “Adventures of Smiley” by clicking on the link “to play”. The documentation is also available in PDF format, which can be either downloaded or read online. The Shockwave plug-in is needed to play the game online, and can be downloaded from the Macromedia Website. To play the game, follow the live instructions given in the game.

The game can be further modified or extended based on the logic. You will need Macromedia Director to code the game play and 3D Studio Max to do the 3D modeling. The platform should be Windows, because this is the only operating system that Director can be installed. The finished game can be played on Windows, MAC and some LINUX operating systems.

APPENDIX B

TERMS AND ABBREVIATIONS

Terms and Abbreviations

OpenGL

OpenGL (Open Graphics Library) is widely used API for programming the 2D and 3D graphics components of software programs.

SDL

Simple DirectMedia Layer is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video frame buffer.

API

An application programming interface (API) is the interface that a computer system, library or application provides in order to allow requests for services to be made of it by other computer programs, and/or to allow data to be exchanged between them.

Game Engine

A game engine is the core software component of a computer or video game or other interactive application with real-time graphics.

3D Studio Max

3ds Max is a full-featured 3D graphics application developed by Autodesk Media & Entertainment

Shockwave

Software developed by Macromedia to create and distribute multimedia animations on the Web.

Model resource

a model resource is the 3D geometry or polygon mesh that describes the form of a model. A model resource is only visible when in use by a model. Several models can use the same model resource.

Model

a model is the visible 3D object in a shockwave3D world. It is based on a model resource(s) and has materials (shaders) as one of its properties

Material

a material is an entity that allows a model to be rendered with simulated surface attributes, such as color, reflectivity, transparency, shininess, smoothness. A material defines the totality of the look within a shader.

shader

Shaders are rendering algorithms that define the overall look of a mesh

Texture (map or texture map)

A texture is a bitmap image that can be one of a material's attributes.

camera

A camera is a viewport into the 3D world. When a 3D member is placed on the Stage, the sprite we see is a particular camera's view into the 3D world.

light

Lights illuminate objects in a 3D world and enable objects to be seen.

Projector

projector is a small, start-up file that launches the main application stored in one or more separate Director movies.

REFERENCES

[1] Dean Utian. (2006,Oct.). DIRECTOR tutorials and technotes. The University of Southwales, Sydney, Australia. [Online]. Avaiable: http://www.fbe.unsw.edu.au/Learning/DIRECTOR/

[2] 3D Buzz. (2006, Oct.). 3D STUDIO MAX Video training Materials.3D Buzz, Tennesse. [Online]. Available: http://www.3dbuzz.com/vbforum/sv_home.php

[3] Devmaster.net. (2006, Oct.). DevMaster's Game and Graphic Engines Database. Devmaster.net, USA. [Online]. Available: http://www.devmaster.net/engines/

[4] Game programming Wiki. (2006,Oct.).Game programming Wiki. Gpwiki.com, USA. [Online]. Available: http://gpwiki.org/

[5] Game Pipe Labs. (2006,Apr.).Game Engines. University of Southern California, Losangeles, California. [Online]: http://gamepipe.usc.edu/docs/ GameEngines.pdf

[6] Martin Persson. (2006,Jan.). Development of three AI techniques for 2D platform games. Karlstad University, Minnesota. [Online]. Available: http://www.diva-portal.org/kau/abstract .xsql?lang=en&dbid=1

[7] Steven M. Woodcock. (2006,Apr.).Classic games and AI, Colorado Springs, USA. [Online]. Available: http://www.gameai.com/clagames.html

[8] OGRE Developers Team. (2005,Feb.). opensource graphic engines,UK. [Online].Available: http://www.ogre3d.org

[9] Kirupa Chinathambi. (2006, Sep.).3D in FLASH and isometric view,MIT,Masachusets. [Online]. Available: http://www.kirupa.com/

[10] Jobe Maker. (2003, Jan.). First steps of FLASH game design. Peachpit press, Berkely, California. [Online]. Available: http://www.peachpit.com/ articles/article.asp?p=30431&seqNum=1&rl=1

[11] Durbnpoison.(2001, Nov.).Difference between DIRECTOR and FLASH. Merceville,NJ. [Online].Avialable: http://www.mediamacros.com/item/item-1005255569/

[12] Wikipedia.(2006, Oct.).About 3D STUDIO MAX and game engines. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/3D_Studio_Max

[13] Associated Press. (2006, Oct.).Not playing around: Scientists say video games can reshape education,. CNN, USA. [Online]. Available: http://www.cnn.com/2006/EDUCATION/10/17/video.games.ap/index.html

[14] International Hobo. (2004, Mar.). A guide to Computer Game genres. International Hobo, USA. [Online]. Available: http://www.ihobo.com/gaming/genres.shtml

[15] Audacity Developing team. (2006, Aug.). Audacity:Free audio editor and Recorder. Audacity, USA. [Online]. Available: http://audacity.sourceforge.net/

[16] Electric Rain.(2004, Aug.). Swift3D. Electric Rain Inc., Colorado. [Online]. Available: http://www.erain.com