

CIS660: Advanced Topics in Computer Graphics and Animation

Dr. Stephen H. Lane
(shlane@cis.upenn.edu)

Spring 2005

Overview

- SigGraph Paper Presentations
- Project Design Doc
- User Interface Design

SigGraph Paper Presentations

- **Speaker**
 - 20 minute presentation (approx. 10-12 slides)
 - Include video/demo/images of work, if available
 - 5 minutes Q & A
- **Audience**
 - Responsible for reading papers in advance
 - Come to class ready to ask (and be asked) questions
 - Class participation will be factored into course grade

Course Grading

- Breakdown
 - 15% Primer
 - 25% SigGraph paper presentations
 - 10% Authoring Tool Design Document
 - 50% Authoring Tool Implementation

SigGraph Paper Presentation Guidelines

- Presentation Overview/Roadmap (*1 slide*)
 - (Tell me what you are going to tell me)
- Body of Presentation (Total of *8 to 10 slides*)
 - The principal thesis of the paper. (*1 slide*)
 - The background work leading to the interest in, or need for this paper (1 or 2 slides)
 - The major topics of the paper. (*5 or 6 slides*)
 - Describe the algorithms, techniques, experiments, etc.
 - Evaluate the paper. (*1 slide*)
 - Is the work significant?
 - Are the results convincing?
 - Did they compare their results to other work? to reality?
 - Are the methods practical?
 - Do the methods appear worthwhile to try to re-implement?
 - What questions would you ask the authors?
- Summary and Conclusion (*1 slide*)
 - (Tell me what you told me)

Authoring Tool Project

- Project Objective:
 - Develop an authoring tool enabling creation of a new type of:
 - user interaction,
 - animation/simulation capability, or
 - 3D graphics special effect
- Requirements
 - Assume you are in a tools group at a game development company or special FX house
 - Need for the tool
 - Target Audience
 - How will the tool be used

Authoring Tool Project - Con't

- Source of Technology
 - SigGraph Conference proceedings over the period 2001-2004
 - Types of functionality/effects to be implemented
 - Character animation
 - Facial animation
 - Physically-based modeling and simulation
 - Shaders (i.e. surface properties such as hair, fur, ice, etc.)
 - Clothing Simulation
 - 3D Modeling
 - Rendering and Lighting
 - Natural phenomena – (particles, smoke, fire, explosions)
 - Natural phenomena – (fluids, water)
 - Artificial life
 - Behavioral Animation Intelligent agents
 - Image-based Modeling and Rendering

Tool Project Groups

- Group1:
 - Joe Kider
 - Liming Zhao
- Group2:
 - Mike Lang
 - Kennedy Behrman
- Group3:
 - Ju Hee Kwon
 - Eric Dziurzynski
- Group4:
 - Warren Longmire
 - Nick Haldar-Sinha
- Group5:
 - Michael Brainerd
 - Gary Katz
- Group6:
 - Michael Lehr
 - Paul Turner
- Group7:
 - Feng Zhang
 - Ghulam Lashari

Authoring Tool Project - Con't

- Phases of Development

- Requirements

- Design

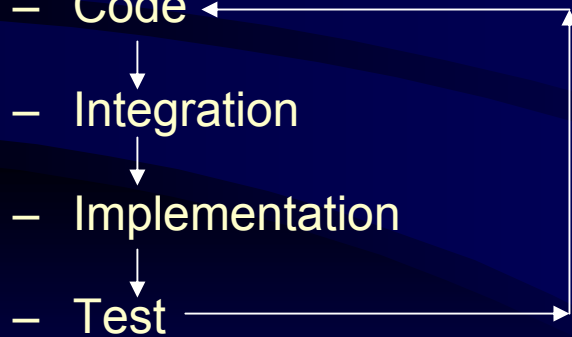
- Detailed Design

- Code

- Integration

- Implementation

- Test



Authoring Tool Project - Con't

- Implementation Details
 - Develop tool to extend the capabilities of the Maya authoring environment using:
 - MEL
 - C++ plug-in API
 - Tool can be used to:
 - export data for use with a third-party run-time engine
 - create a new effect for use in Maya
 - Recommended Reading:
 - [Complete Maya Programming: An Extensive Guide to MEL and the C++ API](#), by David A.D. Gould.

Authoring Tool Design Document

- Main Sections of Design Doc
 - Authoring Tool Design
 - Authoring Tool Development
 - Project Work Plan

Authoring Tool Design Doc - Con't

- **AUTHORING TOOL DESIGN**
 - Significance of Problem or Production/Development Need
 - Technology
 - Provide overview of technology in two SigGraph papers
 - Why did you choose these papers?
 - How do they relate to each other?
 - What is unique about the combination?
 - Design Goals
 - Target Audience
 - User goals and objectives
 - Tool features and functionality
 - Tool input and output

Authoring Tool Design Doc - Con't

- User Interface
 - GUI Components and Layout
 - User Tasks
 - Work Flow

Authoring Tool Design Doc - Con't

- **AUTHORING TOOL DEVELOPMENT**
 - Technical Approach
 - Algorithm Details
 - Maya Interface and Integration
 - Software Design and Development
 - Target Platforms
 - Hardware
 - Software
 - Software Versions
 - Alpha Version Features (first prototype)
 - Beta Version Features
 - Description of any demos or tutorials

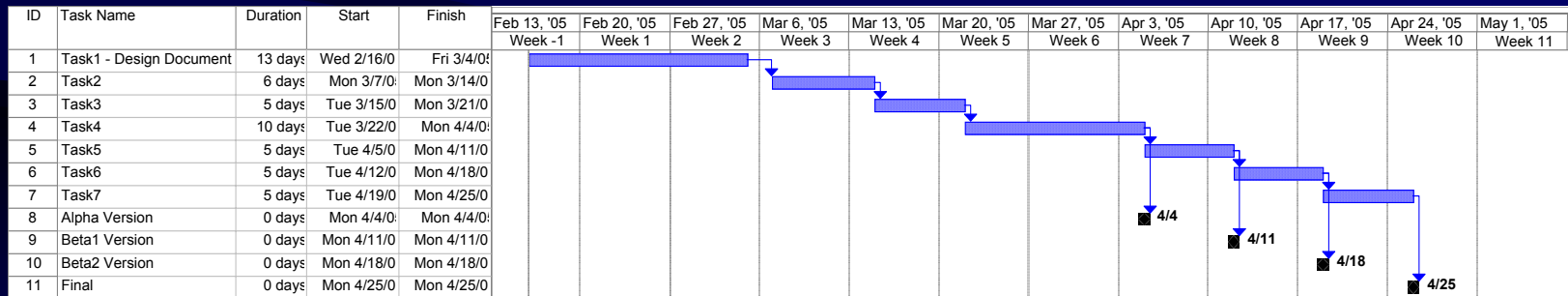
Authoring Tool Design Doc - Con't

- PROJECT WORK PLAN

- Tasks

- Alpha Version
 - Beta Version
 - Final Version

- Schedule (Gantt Chart)



User Interface Design

- Attributes of a good user interface design
 - Invisible
 - Minimal training requirements (easy to learn)
 - High transfer of training (easy to remember)
 - Predictable
 - Few errors
 - Easy to recover from errors
 - People perform real tasks well (efficient to use)
 - It is flexible
 - It is intelligent
 - People like it (subjectively pleasing)

User Interface Design - Con't

- Task Analysis
 - Look at concrete examples of users performing similar tasks
 - Helps define user interface design requirements
 - What does the user need to know to do these tasks?
 - What does the user need to view (on screen) to do these tasks?
 - What are the exceptions and error conditions
 - Allows construction of typical use scenarios
 - Specific examples of how a user might use the system.
 - One scenario for each major class of users (novice, intermediate, expert)
 - Targets what is important to optimize?
 - Will want to make those tasks efficient and easy to use

User Interface Design - Con't

- Results of task analysis:
 - Proto scenarios to be used during design phase
 - List of the things users want to accomplish (goals)
 - Information they will need to accomplish those goals
 - Steps to be performed and interdependencies
 - Criteria to determine quality of results
 - Communication needs of users with other people
- Uses
 - Refine the user interface
 - Demonstrate to management, marketing, customers what your concept is
 - Can replace much textual specification

User Interface Design - Con't

- Guidelines for designing a good user interface
 - Use an Iterative design process
 - Involve the user in the design
 - Less is More ("keep it simple")
 - If complex to explain/document - redesign
 - Concise language
 - Avoid extraneous pictures and information
 - Good Graphic and Color Choices
 - Use shape and color to direct user attention
 - Group related objects
 - Balance white space
 - Few fonts and colors (5 to 7 colors)
 - appropriate contrast
 - some people are color blind (8% of males)

User Interface Design - Con't

- Speak the User's Language
 - Use common words, not "techno-jargon"
 - Metaphors can help or hurt
- Minimize User Memory Load
 - Short-term memory = 7 ± 2 items;
 - Retention = 30 sec to 2 min, unless interrupted
 - Recognize, not recall
 - Menus rather than command line
 - User prompts
 - Take advantage of ubiquitous command sets (e.g. cut/paste)

User Interface Design - Con't

- Consistency
 - Locations of information, names of commands, etc.
 - Same command should always have same effect
 - Size, location, color, wording, function, sequencing, ...
 - Following standards helps
 - Seems easy, but often not followed.

User Interface Design - Con't

- Provide appropriate feedback
 - What system is doing
 - How input is being interpreted
 - Acceptable response times:
 - 0.1 sec for acknowledging a command
 - about 4 sec for an operation
 - Percent-done progress bars help
- Clearly marked Exits
 - Cancel buttons
 - Make all user actions easily reversible (undo)
 - Users (even experts) will make errors

User Interface Design - Con't

- Help the user get started with the system
 - No more than 1 simple overview screen to get started doing real work
- Error prevention
 - Selection rather than recall or entry
 - Remove or grey-out illegal choices
 - Confirm user input
- Good error messages
 - Help users when they are in trouble
 - Easy error recovery.
 - Tell why the error happened and how to fix it
 - Clear language; no codes
 - Be precise. Not "syntax error"
 - Be polite and not accusing; positive wording:
 - Not: "FATAL ERROR", etc.
 - Blame the system, not the user
 - No humor or snide comments

User Interface Design - Con't

- Give the user a mental model of the system
 - Not just a bunch of ad-hoc features
 - Related to consistency
- Provide Shortcuts
 - For experienced users
 - command keys, macros, styles, recent files, etc.
- Accommodate individual differences
 - Novice and expert
 - Handicapped users
 - Customization

Design Adages

- Things that look different should act different.
- If it is not needed, it's not needed.
- The information for the decision needs to be there when the decision is needed.
- The user should control the system. The system shouldn't control the user. The user is the boss, and the system should show it.
- The idea is to empower the user, not speed up the system.
- Don't overload the user's buffers.
- Keep it simple.
- Things that look the same should act the same.
- The user should be able to do what the user wants to do.
- Every action should have a reaction.
- Everything in its place, and a place for everything.
- Let people shape the system to themselves, and paint it with their own personality.
- Error messages should actually mean something to the user, and tell the user how to fix the problem.
- The best journey is the one with the fewest steps. Shorten the distance between the user and their goal.
- Everyone makes mistakes, so every mistake should be fixable.

More Design Adages

- The more you do something, the easier it should be to do.
- Cute is not a good adjective for systems.
- Keep it neat. Keep it organized.
- Consistency, consistency, consistency.
- The user should always know what is happening.
- Minimize the need for a mighty memory.
- Know thy user, and YOU are not thy user.
- If I made an error, at least let me finish my thought before I have to fix it.
- Design for regular people and the real world.
- Eliminate unnecessary decisions, and illuminate the rest.
- You should always know how to find out what to do next.
- If I made an error, let me know about it before I get into REAL trouble.
- Even experts are novices at some point. Provide help.
- Provide a way to bail out and start over.
- Don't let people accidentally shoot themselves.
- Color is information.
- The user should be in a good mood when done.
- The fault is not in thyself, but in thy system.

Even More Design Adages

- To know the system is to love it.
- Deliver a model and stick to it.
- Follow platform conventions.
- Make it hard for people to make errors.
- The system status (i.e., what's going on should always be visible.
- Accommodate individual differences among users through automatic adaptation or user tailoring of the interface.
- Make it easy for a beginner to become an expert.
- No you can't just explain it in the manual.
- Provide user documentation that is easy to search, focused on the user's task, lists concrete steps to be carried out, and is not too large.
- The system should speak the users' language, following real-world conventions.
- Instructions for use of a system should be visible or easily retrievable.
- What does marketing think it wants? Ok, now how do we show them they're wrong?
- What does management think it wants? Ok, now how do we show them they're wrong?
- Allow users to tailor frequent actions.
- Users don't know what they want, and users can't always say what they know.
- Roll the videotape.

- Common sense is an uncommon commodity.
- Make objects, actions, and options visible.
- Data drives good design.
- Help users develop a conceptual representation of the structure of the system.
- Minimize the amount of information a user must maintain in short-term memory.
- It's a jungle. Be careful out there.
- People should not have to remember information across a dialogue.
- Make it impossible to make errors that will get the user into REAL trouble.
- Dialogues should not contain information that is irrelevant or rarely needed.
- Testing, testing, testing.
- Keep the user mental workload within acceptable limits.
- Minimize the amount of information recoding that is necessary.
- Minimize the difference in dialogue both within and across interfaces.
- An ounce of good design is worth a pound of technical support.
- Provide the user with feedback and error-correction capabilities.
- So how is this better than what the competition is doing?
- Provide good error messages that are expressed in plain language, precisely indicate the problem, and constructively suggest a solution.
- Whadya mean, they're not all computer scientists?
- Support undo and redo.
- Different words, situations, or actions should result in different things happening.
- The best user interface is one the user doesn't notice.