

# CSE 125 Spring 2005

## Lecture 2 Tips, Tools, and Tricks Where do we start?

## Our Related Experiences

### Karen

- Graphics
  - OpenGL
- Math-Computer Science Major
  - 4<sup>th</sup> year
- Games
  - Addicted to WoW and video games
  - Took CSE125 last Spring
  - <http://pisa.ucsd.edu/cse125/2004/cse125g2/>



### Allen

- Graphics
  - OpenGL
- Programming
  - "I like programming"
- Computer Science Major
  - Graduated Winter 05
- Games
  - Addicted to WoW and video games
  - Took CSE125 last Spring

# Some Philosophy

- Get to know your group
  - Know your strengths and weaknesses
  - Be adventurous
  - EAT!!! (a lot)
- Settings your goals
  - Make a list of everything you want in the game
  - Priority this list and work from there
  - Work from the bottom up, starting with most basic
  - Start small to get a jump start
  - Make sure basics are working before building on top of them
- CVS!!!
  - MUST USE IT, else you will REGRET IT.
  - TortoiseCVS / WinCVS – easy gui for CVS
- Don't worry about AI unless you have time
- Make a FUN GAME – keep it simple for the demo
- Game should not revolve around special features

# I. General Info

- Use the language you're most comfortable with
  - C/C++/C#/Java/Visual Basic...
- Programming help
  - C/C++/Java
- Object-oriented code
- Organization, well-documented
- Decide on interfaces early, keep it simple
- Libraries
  - DirectX
    - Collection of libraries that deal with components of the game
    - DirectInput, DirectSound, etc.
  - STL
    - May want to use if using C++
  - Java
    - Many libraries available

## II. Graphics

- Direct 3D
  - Microsoft 3D graphics library
  - We *might* be able to help with this
- OpenGL
  - Another popular graphics library
  - We can help with this
  - If you use Java, lots of 3D bindings available (LWJGL – Access OpenGL, OpenAL, very familiar style)
- Use vertex buffers (vertex arrays) for speed
- Create/find a particle system

## Direct3D vs OpenGL

- Which to choose?
- Both support most functions
  - OpenGL through extensions, DirectX more natively
- OpenGL
  - Can be faster if you know what you are doing
  - You are probably more familiar with it
  - Somewhat harder to do the latest tricks (e.g. shaders)
  - GLEW extension loading library – handles extensions
- DirectX Graphics
  - Looks a *lot* like OpenGL now
  - Probably new to most of you
  - Cryptic structure and function names

## Direct3D vs OpenGL (2)

- OpenGL can be used in place of DirectX Graphics
  - Can still use DirectSound, DirectInput, DirectPlay
- Use what you're comfortable with
  - Learning a new API has a huge learning curve
- From previous years
  - Historically, groups using OpenGL had a *much* easier time than those using DirectX Graphics
  - Both can make pretty graphics

## III. Sound

- DirectSound
- OpenAL – similar architecture to OpenGL
  - Used in KK
  - Easy to use
- SDL – Simple Directmedia Layer
  - Easy to use
- FMOD
  - Supposedly very easy to use

Pick whichever you want – do a little research  
They all have lots of example code

## IV. Input

- DirectInput
  - Library for managing input devices
  - Keyboard/mouse/joystick/etc.
- SDL
  - Can also manage input
- If all you want is keyboard and mouse...
  - You can just use Win32 events instead of DirectInput
    - WM\_MOUSEMOVE, WM\_KEYDOWN
  - DirectInput can have strange problems on a crash, but needed if you want to support joysticks/gamepads
- Lab will have Logitech gamepads (similar to PS2's)

## V. Networking

- DirectPlay (Deprecated now)
  - DirectX communication library
  - Interface to host/join games, send/receive messages
  - Can use UDP, IPX, etc for actual transport
    - Your game doesn't care (except for performance)
- Lobbies / Server Finders
  - Meeting area for player to chat/form games
  - Might just be automatic (EnumHosts + GUID)
- Client/Server Model
  - Server holds global game state, receives input from clients
  - Clients animate graphical objects as dictated by the server
  - P2P is possible, but more error prone
- Some teams swore that WinSock was easier
  - Less setup code, but no freebies (guaranteed delivery, multicast)
- Libraries
  - <http://www.opentnl.org> (some advanced features)

## VI. Game Engines & Tools

- Engines
  - Better end results
  - May not learn as much as implementing yourself
- Graphics
  - OGRE, Irrlicht, CrystalSpace, many others
  - Abstracts many low level details
  - Many advanced features included (effects, scene graph handling, etc)
  - Learning curve, waste if you spend time learning and end up not using
- Math
  - Probably want to use pre written Vector, Matrix, Solver classes. You want this code to be solid.

## VII. Physics

- Can make a game more interesting
- May (half-life) or may not (mario) need it
- May be easy to fake
- Rigid body physics (if you want this working well, should probably use a pre-built engine, harder to write on your own)
- Libraries
  - ODE, Newton, Tokomak, etc
  - Again, learning curve
  - Can get nice behavior for free
  - Very hard to get similar results if you write your own
  - But you may want 'faker' physics anyway

## VIII. Art & Models

- Do not have to make your own models
  - Use the web, lots of free models to choose from
- Or if you're up for the challenge...
  - 3D Studio MAX
    - high learning curve, if you already know it, then great
    - otherwise, may not have time
  - Milkshape3D
    - low learning curve, freeware, easy to use, but buggy, limited features, frustrating at times
    - Used in KK
  - Blender3D
    - not so good interface, also free
- Textures
  - Photoshop
  - Photos
  - Web
- Animation & Model Loading
  - Find loaders on the web, or write your own...?
  - Know which formats you're working with and stick with them (.3ds, .md, etc.)
  - Converters (3ds -> md, etc.; Deep Exploration)

## Resources

- The web is your friend
- Look at previous class projects
- Games-oriented sites
  - [www.flipcode.com](http://www.flipcode.com) – code, articles
  - [www.gamedev.net](http://www.gamedev.net) – code articles
  - [nehe.gamedev.net](http://nehe.gamedev.net) – tutorials, sample code
  - [www.gametutorials.com](http://www.gametutorials.com) – tutorials, sample code
  - [www.gamasutra.com](http://www.gamasutra.com) – game design and industry articles
  - [www.penny-arcade.com](http://www.penny-arcade.com) – *the* gamer webcomic
- Graphics/Math sites
  - [www.opengl.org](http://www.opengl.org) – news, message boards
  - [www.realtimerendering.com/int/](http://www.realtimerendering.com/int/) – collision detection routines
  - <http://www.geometrictools.com> – lots of classes for math and 3D done well

## More Resources

- <http://www.libsdl.org/index.php> - sdl
- <http://www.google.com> - friend
- Books are a good help sometimes, ask and we may have some

## Implementation Tips

- Programming
  - OO important – keep code modular and isolated from each other
  - Use the debugger
  - Handy console classes available if need
  - Logs
  - Test suites – very useful (look into unit testing)
  - Use asserts properly – when testing for situations that must not happen for things to work
  - Nice to have game settings read in during execution

## Where to start?

- What kind of game are you making?
  - Discuss with group, make sure everyone is happy and motivated by this idea
  - Sketch out design as much as possible
  - What is each person going to concentrate on?
  - Research on all the available tools/libraries/etc.
- Start with sample code
  - Try to run some tutorials, samples, previous class projects
- Get to know each other
  - EAT
- Work with each other, meet up often, ask TA questions
- Set milestones (look at previous group websites for example), schedules, specifications, etc.

## General Tips

- Ask us any questions you have
- Worry about optimization later, get game to work first
- Stuck? Look on the web, ask your group, ask other groups, ask TAs
- Share your progress and work with your group
- Play the game, make it fun

# Who are you?

- Introductions...