

CSE 190

Software System Design and Implementation

Spring 2003

Lecture 2: Tools, Tips and Tricks

or

“Where Do We Start?”

My Related Experience

- Graphics
 - Started out in Scientific Visualization for Chemistry Dept.
 - Took CSE 167
 - 4 years total OpenGL experience
- CSE 190 last spring
 - Shrapnel (Team Vertigo)
 - pisa.ucsd.edu/cse190/2002/cse190g3/
- Win32 / MFC / DirectX / CVS
- 7 years total C++ experience
- Hardcore gamer – I live for this stuff

Who are You?

- Everyone has their own background...
 - Let's find out...

Some Philosophy

- Get to know your group
 - Know your strengths and weaknesses
 - Don't be afraid to try something new
- How much is too much?
 - Set your sights as high as you can
 - Implement from the bottom up with basic features
 - Prioritize all of your features (necessary, nice, wish-list)
- Talk to your group
 - Sounding out a problem often solves it
- Don't lose your work: Use CVS
 - TortoiseCVS (www.tortoisecvs.org) – does *most* of what you need
 - WinCVS (www.wincvs.org) – does the rest, but not as easy to use
- Make something *fun*

DirectX vs. OpenGL

- DirectX is a collection of libraries
 - ♦ Each deals with a particular component of your game
- DirectX 9
 - ♦ DirectX Graphics
 - ♦ DirectSound
 - ♦ DirectInput
 - ♦ DirectPlay
 - ♦ DirectShow
- OpenGL is just a graphics library
 - ♦ Low level library
 - ♦ GLUT adds some window management features
 - ♦ SDL (www.libsdl.org) adds more DirectX-like features

DirectX vs. OpenGL (2)

- Which to choose?
- Both support most functions
 - ♦ OpenGL extensions, DirectX CAPS bits
 - » extgl helper library at www.levp.de/3d/
- OpenGL
 - ♦ Can be faster if you know what you are doing
 - ♦ You are probably more familiar with it
 - ♦ May be harder to do the latest tricks (e.g. shaders)
- DirectX Graphics
 - ♦ Looks a *lot* like OpenGL now
 - ♦ Probably new to most of you
 - ♦ Cryptic structure and function names

DirectX vs. OpenGL (3)

- OpenGL can be used in place of DirectX Graphics
 - Can still use DirectSound, DirectInput, DirectPlay
- GLUT, SDL, DirectX and Win32 all have ways to create windows, change resolutions/color depth and toggle fullscreen
 - Can't usually mix and match, though
- From previous years
 - Groups using OpenGL had a *much* easier time than those using DirectX Graphics
 - Don't underestimate the learning curve

DirectSound

- Used to be DirectSound, DirectSound 3D, DirectMusic
- Load and play sounds and music in your game
 - Supports MIDI, wav, streamed mp3, etc.
 - Multiple emitters
 - 3D sound
 - Channel effects (e.g. reverb)
 - Interactive music (DirectMusic Producer)
- Other libraries provide similar features
 - FMOD (www.fmod.org) – very easy set-up
 - OpenAL (www.openal.org)

DirectInput

- Library for managing input devices
 - ◆ Keyboard
 - ◆ Mouse
 - ◆ Joystick
 - ◆ Steering wheel
 - ◆ Force feedback
- 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
 - ◆ Needed if you want to support joysticks/gamepads

DirectPlay

- DirectX communication library
 - ◆ Interface to host/join games, send/receive messages
 - ◆ Can use TCP, IPX, PPP 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)

DirectShow

- Video/audio playback and capture
- Media players, editors, etc.
- Probably not something you'd use for this project
 - ♦ Intro Movie
 - ♦ Video Texturing
 - » Broken in DX8, might have been fixed in 9

Resources on the Web

- Google (google.com)
 - ♦ *The* search engine
- Games-oriented sites
 - ♦ www.flipcode.com – code, articles
 - ♦ www.gamedev.net – code articles
 - ♦ nehe.gamedev.net – tutorials, sample code
 - ♦ www.gametutorials.com – tutorials, sample code
 - ♦ www.gamasutra.com – game design and industry articles
 - ♦ www.penny-arcade.com – *the* gamer webcomic
- Graphics/Math sites
 - ♦ www.opengl.org – news, message boards
 - ♦ www.magic-software.com/SourceCode.html - various
 - ♦ www.realtimerendering.com/int/ – collision detection routines



Resources for the Hardcore

- NEC Digital Library (www.researchindex.org)
 - Computer science technical papers
 - Can be heavy reading, but might be worth it
- SIGGRAPH (www.siggraph.org)
 - The premiere graphics conference, lots of papers available online
- Open-Source game/rendering engines
 - Crystal Space (crystal.sourceforge.net)
 - OGRE (ogre.sourceforge.net)
 - Both difficult to follow, but you can learn from them
- Textures/Models/Sounds
 - Google might be your best bet
 - www.polycount.com – lots of models for Half-Life, Quake, Unreal
 - » Loading code is easy to find online

Implementation Tips

- It's not a perfect world
 - Code assuming something is going to break
 - Check return values from system/DirectX calls!
 - Initialize variables (0xcdcdcdcd, 0xbaadf00d, 0xdeadbeef)
 - Assert on unexpected situations, don't code around them
- Debugging and testing support
 - Printf's, logs, consoles, data structure dumps (from debugger)
 - Step through code as it executes
- Remember that computers are (usually) deterministic
 - If something goes wrong, it can be tracked down

Where to Start?

- Decide (and agree) on what you want
- Start with sample code
 - ♦ nehe.gamedev.net base code, tutorials
 - ♦ CSE 167 or MAE 293 projects
 - ♦ DirectX samples
- Build major components separately
 - ♦ Sound, input, graphics, networking
 - ♦ Get each one working (and modular)
- Try out Extreme Programming (XP)
 - ♦ Pair programming – faster than two people working alone
 - » Even faster to have the whole team in the lab...
 - ♦ Refactoring – plan for change

Then What?

- Integration
 - ♦ Most crucial step
- Integrate Early?
 - ♦ Gives you a good test bed for later changes
 - ♦ Increases dependencies between programmers
 - » Can be mitigated by programming together as a team
- Integrate Late?
 - ♦ Don't find bugs until late in the quarter
 - ♦ It's the 'glue' code that usually takes the most time
- Play the game!

Tools to Help Along the Way

- Deep Exploration
 - ♦ 'Converts anything to anything' – for 3D formats
 - ♦ Demo available
- Milkshape 3D
 - ♦ Shareware, simple interface (too simple?)
 - ♦ Loading code for models, animations is easy to find
- 3DS Max, Maya, Lightwave
 - ♦ Full-featured commercial tools
 - ♦ Game industry artists seems to love Max
 - » 3DS loading code everywhere you look
 - ♦ Find your own copy

Other Tips

- Adapt other code to your own purposes
 - ♦ Even other languages (VB, Delphi, C#?)
 - ♦ DirectX ↔ OpenGL
 - ♦ Find modules that almost fit your needs and hack them
- Graphics
 - ♦ Use vertex buffers (vertex arrays) for speed
 - ♦ Create/find a particle system
 - » All the teams last year used theirs to good effect
- DirectSound
 - ♦ Use soundbuffers, not tracks, for spatial effects
 - ♦ Look at Play3DSound sample, dsutil.cpp (CSoundManager)
- Physics? AI?
 - ♦ Not enough time: design around them
 - ♦ ODE (opende.sourceforge.net) – If you *insist* on having 'real physics'

More Tips

- DirectPlay
 - ◆ Set your own GUID
 - ◆ Look at SimpleClient and SimpleServer
 - ◆ Send updates every tick, or only when things change?
- Collisions – cheat!
 - ◆ Sphere-Sphere and sphere-tri can get you pretty far
 - ◆ Make a game with 3D graphics, but 2D mechanics (kart racing)
- Visual C++
 - ◆ Problems? Ask me
 - » Compiler and linker can be very fickle
 - ◆ Try www.geocities.com/alibagcity/articles/vcshort.html

Wrap Up (Finally...)

- Decide on what you want – lots of variety possible
 - ◆ Last year had 5 games in 4 genres (only 2 shooters)
- Divide up the work (this will change!)
- Start with sample code
- Hit a snag? Look on the web, or ask me
 - ◆ The web is probably faster, but I'm always happy to help
- Integrate ASAP
- Communicate with your group
- Play the game, make it fun

For Next Time...

- Define game idea and group name
- Start working on specification, schedule and milestones, modules, member responsibilities
 - Will be due next week – more via email