Programming Game Engines  
ITP 485 (4 Units)

Objective  This course provides students with an in-depth exploration of 3D game engine architecture. Students will learn state-of-the-art software architecture principles in the context of game engine design, investigate the subsystems typically found in a real production game engine, survey some engine architectures from actual shipping games, and explore how the differences between game genres can affect engine design. Students will participate in individual hands-on lab exercises, and also work together like a real game development team to design and build their own functional game engine by designing and implementing engine subsystems and integrating 3rd party components.

Concepts  Engine subsystems including rendering, audio, collision, physics and game world models. Large-scale C++ software architecture in a games context. Tools pipelines for modern games.

Prerequisite  CSCI-102 and ITP-380

Lecture  2 hrs/week

Lab  2 hrs/week

Course Structure  Lectures given on Tuesday. Project assignments and some supplemental lectures given on Thursday; instructor and/or TA available for questions and help during office hours, Thursday lab sessions, via email, and by appointment.

Textbooks  Required:

Recommended:
Grading  Final grade is based upon the student’s score on individual labs and assigned problems, the midterm and final exams, and the grades earned on a multi-lab team-based project. For the team project, it is especially important that students display their names on all documentation and source code you personally produce, so that the instructor will know which student contributed which material.

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual Labs and Assigned Problems</td>
<td>20%</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>20%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>20%</td>
</tr>
<tr>
<td>Team Project: Individual Contribution</td>
<td>20%</td>
</tr>
<tr>
<td>Team Project: Overall Project Grade</td>
<td>20%</td>
</tr>
<tr>
<td>TOTAL POSSIBLE</td>
<td>100%</td>
</tr>
</tbody>
</table>

Policies  It is the student’s responsibility to complete both individual and team-based assignments on or before deadlines as set by the instructor. Before logging off a computer, students must ensure that they have submitted to Subversion any work created during the class or lab session. Any work saved to the local computer’s disk drive may be erased after restarting the computer. ITP is not responsible for any work lost. ITP offers Open Lab use for all students enrolled in ITP classes. These open labs are held beginning the second week of classes through the last week of classes. Please contact your instructor for times and days for the current semester.

Academic Integrity  - The use of unauthorized material, communication with fellow students during an examination, attempting to benefit from the work of another student, and similar behavior that defeats the intent of an examination or other class work is unacceptable to the University. It is often difficult to distinguish between a culpable act and inadvertent behavior resulting from the nervous tension accompanying examinations. When the professor determines that a violation has occurred, appropriate action, as determined by the instructor, will be taken.

- Although working together is encouraged, all work claimed as yours must in fact be your own effort. Students who plagiarize the work of other students will receive zero points and possibly be referred to Student Judicial Affairs and Community Standards (SJACS).

- All students should read, understand, and abide by the University Student Conduct Code listed in SCampus, and available at: http://www.usc.edu/student-affairs/SJACS/nonacademicreview.html

Students with Disabilities  - Any student requesting academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to me (or to your TA) as early in the semester as possible. DSP is located in STU 301 and is open 8:30 a.m. - 5:00 p.m., Monday through Friday. The phone number for DSP is (213) 740-0776.
Programming Game Engines
ITP 485 (4 Units)

Course Outline

Week 1 – Introduction
- Course overview
- What is a game?
- What is a game engine?
- Engine differences between game genres
- Survey of run time engine subsystems
- Survey of tools and the asset pipeline

Reading:
- Course text: 1.2 – 1.7

Lab:

Pongre 1: One-Dimensional Bouncing Ball
- Create project by copying template
- Create simple bouncing ball (Ogre head) in 1D

Week 2 – Tools of the Trade
- Version control and Subversion
- Microsoft Visual Studio tips and tricks
- Profiling tools
- Memory leak / corruption detection
- Other tools

Reading:
- Course text: 2.1 – 2.5

Lab:

Pongre 2: Bouncing, Paddles and Basic Gameplay
- Create simple box meshes for paddles
- Use Ogre’s OIS human input device API to move paddles
- Detect collisions with paddles and bounce ball
- End/reset game if ball misses one of the paddles

Week 3 – 3D Math for Games (Part 1)
- Points, vectors and Cartesian coordinates
- Vector operations, dot and cross product
- 2D and 3D matrices, homogeneous coordinates
- Hierarchical coordinate frames, change of basis
- Comparison of rotational representations
Reading:
- Course text: 4.1 – 4.5

Lab:

3D Math Problems
- Work some problems, see some applications, get some practice

Pongre 3: Congratulations, It’s a Game!
- Add HUD, scoring, and personalization of your choice

Week 4 – 3D Math for Games (Part 2)
- Lines, line segments and rays
- Spheres
- Planes
- Hardware-accelerated vector math with SIMD
- Random number generation

Reading:
- Course text: 4.6 – 4.8

Lab:

3D Math Problems
- Work more problems, see more applications, get more practice

Team Project
- Teams generate simple design doc and rough project schedule

Week 5 – Fundamentals of Software Engineering for Games
- C++ best practices
- Data, code and memory
- Memory allocation and management
- Errors, exceptions and assertions
- Data structures for games
- Strings and hashed string ids

Reading:
- Course text: 3.1 – 3.3, 5.2 – 5.4
- RECOMMENDED: Lakos: Ch. 0; Ch. 5 sections 5.1 – 5.3, 5.7

Lab:

Team Project
- Teams work on engine and game play systems in parallel

Week 6 – Rendering and the Graphics Pipeline
- Triangle meshes and tessellation
- Coordinate spaces and rendering transformations
- Lighting, color and texturing
- The virtual camera and projection
- The rendering pipeline
  - Tools stage
  - Asset conditioning stage
  - Application stage: Visibility determination and primitive submission
Reading:
- Course text: 10.1 (review), 10.2.1 – 10.2.3, 10.2.7.4 – 10.2.7.5

Lab:

*Team Project*
- Teams work on engine and game play systems in parallel

**Week 7 – Advanced Shading and Visual Effects**
- The rendering pipeline (continued)
  - Geometry processing and rasterization stages: GPU architecture
  - Rendering engine architecture
  - Introduction to global illumination and programmable shaders
  - Particle effects, overlays and decals
  - Tools for debugging and development

Reading:
- Course text: 10.2 (except 10.2.1 – 10.2.3, 10.2.7.4 – 10.2.7.5), 10.3, 10.4, 9.1 – 9.8

Lab:

*Team Project*
- Teams work on engine and game play systems in parallel

**Week 8 – MIDTERM EXAM**
- Midterm review and preparation

Reading:
- Review prior weeks’ readings

Lab:

MIDTERM EXAM during **lab hours**

**Week 9 – Character Animation**
- Fundamentals of character animation
  - Skinning
- Blending, post-processing and compression
- Animation system architecture and pipeline
- Ogre’s animation system
- Introduction to player mechanics
- Interfaces between game characters and animation

Reading:
- Course text: 11.1 – 11.4 (review), 11.5 – 11.12
  - Ogre Manual (online): Section 8
  - RECOMMENDED: Junker: Chapter 9

Lab:

*Team Project*
- Teams work on engine and game play systems in parallel

**Week 10 – Collision and Rigid Body Dynamics**
- Collision
  - Collision detection basics
  - The Gilbert-Johnson-Keerthi (GJK) algorithm
  - Typical collision system architectures
  - Optimization: Broad phase, narrow phase, spatial subdivision
  - The prune and sweep algorithm
- Rigid Body Dynamics
- Review of point mass linear dynamics
- Angular dynamics, moment of inertia
- Collision response
- Numerical integration
- Ray and sphere casting
- Typical Collision/Physics API: ODE, PhysX, Havok

Reading:
- Course text: 12.3 – 12.5

Lab:

Team Project
- Teams work on engine and game play systems in parallel

Week 11 – Gameplay Foundation Systems
- Introduction to gameplay systems
- World editors
- Components of the gameplay foundation layer
- Runtime object model architectures

Reading:
- Course text: 13.1 – 13.4, 14.1 – 14.2

Lab:

Team Project
- Teams work on engine and game play systems in parallel

Week 12 – Engine Subsystem Integration
- Review: The game loop
- Review: Time in games
- Updating a multi-object simulation in real time
- Integrating rendering, physics and animation into the game loop
- Multiprocessor game loops

Reading:
- Course text: 7.1 – 7.5 (review), 7.6, 14.6, 12.5

Lab:

Team Project
- Teams work on engine and game play systems in parallel

Week 13 – The File System, Resources and Streaming
- File system APIs for game engines
- The resource manager
- World chunk data formats
- Streaming
- Engine start-up, shut-down and configuration

Reading:
- Course text: 6.1 – 6.2, 14.3 – 14.4, 5.1, 5.5

Lab:

Team Project
- Teams work on engine and game play systems in parallel
Week 14 – Rounding Out the Game Object Model
- Object references and queries
- Events and message passing
- High-Level Game Flow
- Other gameplay systems (time permitting)

Reading:
- Course text: 6.1 – 6.2, 14.3 – 14.4, 5.1, 5.5

Lab:

Team Project
- Teams work on engine and game play systems in parallel

Week 15 – Wrap-Up
- Overflow topics as necessary
- Getting into the game industry – resumes, interviews, demos
- Life in the game industry

Lab:

Team Labs
- Teams perform final integration and add finishing touches
- Code freeze one day prior to Gold Master

Exam Week – Final Exam and Team Game Project Due