New York University School of Engineering
Department of Computer Science and Engineering

Intro to Game Programming • CS3113 • Spring 2015

Tuesdays and Thursdays, 4:30 p.m. – 5:50 p.m. • MetroTech 2, 8th floor, Room 812

Professor Ivan Safrin
Email: is1296@nyu.edu **Phone:** (646) 492-2593
Office hours: Tuesdays and Thursdays, 9 a.m. - 10 a.m.

Prerequisites

CS2134 Data Structures and Algorithms.

Course Description

"Intro to Game Programming" is a comprehensive course designed to provide an overview of modern game programming practices and their hands-on implementations. We will study the basic building blocks of interactive computer games and learn how to implement them as cross-platform C++ applications using the SDL and OpenGL libraries. Throughout the course, we will be building simple game prototypes around the topics learned in class, building up to a final game project to be completed by the end of the semester.

Course Topics

- Cross-platform game programming using C++ and SDL.
- Rendering 2D and 3D graphics using OpenGL.
- Game physics and collision detection.
- Matrix algebra, linear transformations.
- Animation and special effects.

Course Overview

1. Introductions, overview of tools, setting up a development environment.
2. Basic graphics using OpenGL.
3. Input, time-based movement, basic collision detection.
4. Sprites, text and sprite animation.
5. Game physics, fixed time step.
6. Level editing and procedural generation.
7. Playing sounds.
8. Matrix transformations, advanced collision detection.
9. Advanced OpenGL graphics.
10. Particle systems, complex animation.
11. AI programming.
12. Introduction to 3D graphics.
13. Advanced 3D graphics.
14. Shaders and lighting.
15. Post-processing.
16. Final project demos.

Reading materials

None required. All needed information will be provided in class and online, however the books below are recommended supplementary reading.

Class slides and other helpful materials are available on Github at
https://github.com/ivansafrin/CS3113

Very comprehensive book covering in great depth many of the concepts of graphics programming that we will be studying:
Foundations of 3D Computer Graphics (Steven J. Gortler)
http://www.amazon.com/Foundations-Computer-Graphics-Steven-Gortler/dp/0262017350/

And if you need to brush up on your C++:
Programming: Principles and Practice Using C++ (2nd Edition) (Bjarne Stroustrup)
http://www.amazon.com/dp/0321992784/

Grading / Assignments

Each week, you will have an assignment to complete by the beginning of next week's first class. Most of these assignments will involve implementing the concepts we learned that week as a simple game prototype. As part of the class, you will be expected to create a Github account if you don't already have one and submit your assignments via a git repository (we will go over how to do this during the first class). **Assignments must be submitted on time (before next week's first class, unless otherwise specified) or your assignment grade will be docked half a grade for each day it is late.**

You will also be expected to create a larger game by the end of the semester, which will serve as your final project. You may team up with another person for this project.

Your final course grade will be based on completion of weekly assignments (40%), tests (20%) and your final game project (40%).

Attendance Policy

Attendance will be taken every class. I**f you miss more than three classes, your final grade will be affected.** Please keep in mind that we have a lot of material to cover in a fairly short time and missing even a single class will likely set you back!

Use of external code

For all of the assignments, including the final project, you will be expected to write all of the code yourself. Some example and helper code will be provided for you in class and online. If you wish to use other code, such as an open-source library or snippets of open-source code to implement features **NOT** covered in the class, you may do so, **but you must check in with me** beforehand and all open-source code used in your projects must be clearly marked and properly attributed.

Detailed syllabus

1. **Introductions, overview of tools, setting up a development environment.**
   - Introductions.
   - Overview of game programming.
   - Overview of SDL and cross-platform game programming.

   Assignment: Set up your development environment.

2. **Basic graphics using OpenGL.**
   - Overview of OpenGL and programmable hardware.
   - Drawing polygons.
   - Transformations (moving, rotating and scaling).
   - Loading textures and drawing images using textured polygons.
   - Texture filtering.

   Assignment: Draw a scene using provided assets.

3. **Input, time-based movement, basic collision detection.**
   - Reading elapsed time and time-based movement.
   - Reading and reacting to input.
   - Detecting collisions between squares and circles.

   Assignment: Pong!

4. **Sprites, text and sprite animation.**
   - Texture atlases and texture coordinates.
   - Text rendering using font textures.
   - Creating a basic sprite system.
   - Sprite frames and animation.

   Assignment: Space Invaders

5. **Game physics, fixed time step.**
   - Fixing the time step.
   - Acceleration and gravity.
   - Better collision detection response.

   Assignment: Single screen platformer.

6. **Level editing and procedural generation.**
   - Rendering tiles.
   - Creating and loading level maps.
   - Procedurally generating levels.
   - Using noise for more natural level generation.

   Assignment: Platformer with multiple levels.

7. **Playing sounds.**
    - Loading and playing back sounds.
    - Fading sounds based on distance.

    Assignment: Add sounds to a previous game.

8. **Matrix transformations, advanced collision detection.**
    - Storing transformations as matrices.
    - The matrix stack.
    - Transforming vectors using matrices.
    - Collision detection of transformed squares.

9. **Advanced OpenGL graphics.**
    - Vertex buffer objects.
    - View culling.

10. **Particle systems, complex animation.**
    - Particle system simulation.
    - Tweens and math-based animation.

11. **AI programming.**
    - State machines.
    - A* pathfinding.
    - Node-based pathfinding.

12. **Introduction to 3D graphics.**
    - Perspective, cameras and view frustum.
    - Drawing in 3 dimensions.
    - 3D transforms.

13. **Advanced 3D graphics.**
    - Mip-mapping, anisotropic filtering.
    - Frustum culling.
    - 3D collision detection.
    - Loading OBJ files.

14. **Shaders and lighting.**
    - Loading and using GLSL shaders.
    - Passing vertex data to shaders.
    - Passing uniforms to shaders.
    - Lighting using shaders.

15. **Post-processing.**
    - Rendering to texture and Frame Buffer Objects.
    - Screen-space fragment shader.
    - Common screen-space effects.

16. **Final project demos.**
    - Students present their games in class.