**Video Game Programming Syllabus**

**CMP 428/717: Video Game Programming.** 4 hours, 4 credits.

**Prerequisites:** CMP 338 and MAT 226 (or MAT 313)

**Professor:** Brian Murphy **Phone:** (718) 960-5117

**Office:** GI-211C **Email:** brian.murphy@lehman.cuny.edu

**Grading Policy**

**Expectations:** Students are expected to learn the material covered in class, from the textbooks, and any other assigned readings. Completing programming assignments is a vital part of the learning experience. Students must write readable and complete programs that execute correctly. Students must exhibit expert knowledge of their code.

**Homework:** Programming projects will be assigned embodying the concepts covered in class and in the text books. **Solo Term Project:** Each student will develop their own fully functional video game.

**Team Term Project:** Each student will contribute to a team effort in the development of a fully functional video game.

**Presentations:** Graduate Students will master a topic from the syllabus and then teach that material to the class. All students will present a video game they have created on their own. All students, in teams, will present a video game that their team has created. Game presentations will involve demonstrating the game play, explaining coding issues, and answering questions about their programs.

**Exams:** There will be several quizzes and a final exam.

**Grades:** The grading policy is posted on the course website.

**Materials, Resources and Accommodating Disabilities**

**Textbooks:** Developing Games in Java by David Brakeen, Core Techniques and Algorithms in Game Programming by Daniel Sanchez-Crespo Dalmau, and AI Techniques for Game Programming by Matt Buckland

**Technology:** Students will need to have access to personal computers with a Java IDE. Such computers are available for student use on campus. For students with computers at home, Java IDE Software is available free of charge on the internet. Speak to your instructor for details.

**Accommodating Disabilities:** Lehman College is committed to providing access to all programs and curricula to all students. Students with disabilities who may need classroom accommodations are encouraged to register with the Office of Student Disability Services. For more information, please contact the Office of Student Disabilities, Shuster Hall, Room 238, phone number (718) 960-8441.

**Course Objectives:**

At the end of the course students should know:

1. how to implement the drawing of polygons from the ability to draw a pixel
2. what a real-time system is and be able to implement one in Java
3. the features of a game loop and how to implement one
4. how to read and respond to asynchronous input within a game loop
5. how to move objects around on the screen
6. how to detect collisions between objects represented on the screen
7. how and why to use double buffering and page flipping
8. how to implement an object to encapsulate frame based animation
9. what translation and rotation are and know the underlying mathematics
10. what a polygon model is and how to implement them
11. how to implement scrolling and parallax scrolling and understand the underlying principle
12. how to implement choreographed AI
13. how to implement chasing/evade and targeting algorithms and understand the mechanisms behind them
14. how to utilize finite state machines to implement game object behavior
15. how to use path finding algorithms such as A\*
16. how to perform perspective transformations in order to display 3D scenes
17. how to implement billboarding and understand the mechanisms behind it
18. how to implement backface removal and understand the mathematics behind it
19. how to use the painter’s algorithms to ensure correct occlusion
20. how texture mapping is performed and the mathematics behind it.
21. how to partition the game world for efficient processing of events

**Review Topics:**

Threads

**Topics:**

The Foundation of Computer Graphics

 Plotting a point (pixel) in a chosen color via Java’s AWT API

 The underlying mechanisms of the point plotting routine

 Devising a line drawing routine given the ability to draw a point

 Devising a polygon drawing routine given the ability to draw a line

 Devising an Image drawing routine given the ability to draw a pixel in a chosen color

 Use of AWT API to do all of the above

Building a Game System Framework

 Real-Time Systems

 The basic Game Loop

 Using a KeyListener on a Frame and requesting the focus

 Pseudo asynchronous input via the keyboard

 Player Controlled Actions

 Static Image based Sprites

 Circle versus Circle Collision Detection in 2D space

 Axis Aligned Rectangle versus Axis Aligned Rectangle Collision Detection in 2D space

 Circle versus Line Collision Detection in 2D space

 Responding to Collision

 Game Object Movement

 Playing Sound

 Avoiding Screen Flicker and Tearing via Double Buffering, Page Flipping, and Synching to the Vertical Blank

2D Scene Building

 Building an Animation Object for Frame based Animation

 Building Animated Sprites

 Translation in 2D space for Sprites

 User and Computer control of Sprites

 2D Polygon Models

 Building an Object to encapsulate 2D Polygon Models

 Translation in 2D space revisited for 2D Polygon Models

 Rotation in 2D space for 2D Polygon Models

 User and Computer control of 2D Polygon Models

 Backgrounds

 Scrolling and Parallax Scrolling

Game A.I.

 Choreographed A.I.

 Following Waypoints

 Simple Targeting, Chasing, and Evading Algorithms for 2D Space

 Predictive Targeting, Chasing, and Evading Algorithms for 2D Space

 Finite State Machines

 The A\* algorithm

 Using Genetic Algorithms to build your A.I.

 Using Neural Networks to build you’re A.I.

3D Scene Building

 Perspective Transformation

 Billboarding - Adding the 3rd spacial dimension to Sprites

 Building 3D Animated Sprites

 3D Polygon Models

 Building an Object to encapsulate 3D Polygon Models

 The Painter’s Algorithm, Backface Removal and Occlusion

 Rotation about the x, y, and z axis in 3D space for 3D Polygon Models

 Portal Based Rendering, Binary Space Partitioning and Quad-Trees

 Texture Mapping of Polygon Surfaces in 3D

Stepping back to 2.5 D

 Texture Mapping of Vertical Rectangular Surfaces in 3D

 Limiting movement to 4 degrees of freedom

 Building a map for 3D scenes using a 2D Bird’s Eye View

 Populating the map with Billboard based objects.

Do you wonder how video games work their magic?

Do you want to take a peek behind the curtain?

If you do, this course will interest you.

One of the main goals of this course is to expose to you the general inner workings of video game engines at the foundational level. This will provide you more than just a peek behind the curtain, which will allow you to learn the secrets of the video game programming mathemagicians. There are video game related application programmer interfaces (APIs), frameworks, and engines and there is massively parallel hardware acceleration available from graphics processing units (GPUs) with hard coded graphics operations that can handle the foundational and to the uninitiated quite challenging fundamental computational tasks at the core of every video game for you. Unfortunately, dependence on these facilities hides the algorithmic, mathematical, and implementation details of video game functionality so that video game programming can appear to depend on magic to anyone who looks no deeper. For this reason there will be no reliance on the high level provided by such APIs, frameworks, engines, and GPUs in this course.

You will also be steered clear of the other extreme. You will not be programming in machine nor assembly language nor will you rely on direct access to the hardware to wring out every last cycle and “trick” the hardware into doing things that were not envisioned when it was etched into silicon, as I had to do when I was learning to program video games in the early 1980's. In that era computers were too slow to deliver a satisfactory responsiveness unless coding was done at this lowest of levels. Fortunately, this has not been the case for a couple of decades plus.

While you will not utilize APIs, frameworks, engines, GPUs, machine/assembly language, and direct hardware access in your coding efforts, as concepts they will be addressed. Each does play an important role in video game development once the fundamentals are mastered. Professionals coding video game engine components work mostly at the higher level. However, to do so effectively it is of great value to understand the underlying mechanisms that you will be exploring in this course and on those occasions when the high level interface does not provide the desired functionality a familiarity with the material you will see in this course provides a good starting point to build out new functionality.

In this course everything you build will stem from a general knowledge of Object Oriented Java programming, algorithmic thinking, data structures, mathematics, and the ability to place a pixel on the screen at a desired location and in a desired color. You will learn to leverage and expand your skills and knowledge in these areas. If you work hard putting in the necessary time and effort you should be quite impressed by your accomplishments. This course should leave you wanting to know how to do more when we are done. It will have exposed you to foundational material enabling you to more intelligently decide what to explore further and more easily assimilate new ideas in the future. This class will only scratch the surface of each of the topics it covers and yet you will see that you will have learned a great deal by the end.