*Syllabus*
# CSC – CPE 471
# Introduction to Computer Graphics

| | |
|---|---|
| **Professor:** | Christian Eckhardt |
| **Office:** | Building 14, room 221 |
| **Phone:** | 756-2416 |
| **office hours:** | see PolyLearn top section |
| **email**: | ceckhard@calpoly.edu |
| **Schedule:** | see PolyLearn top section |

## General

Welcome to computer graphics. This course will teach you the fundamentals for writing your own interactive computer graphics applications. This course requires substantial math and programming skills. Experience with C or C++ will be essential and experience with linear algebra will be helpful.

## Development Environment

We will be using

- C++ as the main language
- OpenGL/GLSL as the graphics API
- GLFW for windowing, mouse/keyboard input
- glm for matrix operations
- CMake for cross-platform compilation and development

All your programs must compile in the CSL (but you may develop under other operating systems).  I expect you to be curious about creating interactive visual computing artifacts with algorithms and math, along with being willing to experiment and learn on your own to create interesting computer graphics programs.

## Course Objectives, by the end of the quarter students will

- Understand the graphics pipeline and the basic implementation of the pipeline in modern hardware (and graphics libraries)
- Understand and be able to apply coordinate transforms and affine transformations and understand the application of such transforms using a vertex shader
- Understand and be able to apply basic geometric computations involving points, lines, planes and triangles related to spatial queries
- Be able to program basic data structures to represent a mesh (including applying transforms and necessary data for shading)
- Understand rasterization (algorithm, stage in the graphics pipeline, and relationship to fragment shaders)
- Understand basic hierarchical animation
- Understand and be able to write local shading models (Phong, etc.)
- Understand and be able to apply basic texture mapping
- Be introduced to the application of more advanced computer graphics (including rendering, animation, real time and visualization)
- Practice translating mathematics into C/C++ programs
- In addition, specific to this quarter, expose students to GPGPU via image processing program joint with Applied Parallel Computing class CPE 416 (details to follow).

## Final grade breakdown

- 25%: 10 Lab exercises
- 30%: 4-6 Programming assignments
- 20%: 2 midterms
- 20%: One larger final programming project
  - project must be approved by the instructor (details to follow)

**Labs:** Each lab exercise is due within two lab sessions. For example, if a lab is assigned on Tuesday, it must be completed by the end of Thursday's lab. Labs must be checked off by me or the TA during the lab unless you have my prior permission.

## Assignments

There will be 4-5 substantial individual programming assignments. If your program is late you will lose 20% within first 24 hours after deadline, 40% within 48 hours, 100% after 48 hours. However, you get 2 *free* days for the entire quarter which can be applied to the four programming assignments only. You do not need to explain why you are using the days—these two late days will be automatically applied to any late assignments. After your two late days have been used up, the late penalties apply.

## Midterms

There will be 2 written midterms. No notes.

## Project

You will get roughly 3 weeks at the end of the quarter to work on the final project. You may use any code base for the project, including mobile and web-based.

## Recommended (optional) texts

Please consider “Fundamentals of Computer Graphics” by P. Shirley or "Foundations of 3D Computer Graphics" by S. Gortler. For a reference for modern graphics programming: “OpenGL ES 3.0 Programming Guide" by D. Ginsburg, et. al. There are also numerous helpful tutorial sites for example: http://learnopengl.com/

## Participation

I expect you to participate in class and engage with the class material (studies suggest that taking longhand notes is one of the better ways to guarantee your engagement with the material in class) )[1] I also expect you to form a community of scholars for the duration of the quarter (and hopefully longer). My teaching style is very interactive – if you want to know more about why see ([2]). **Laptops** have been shown to be distracting in lecture[3] and **are not allowed unless specified (or a specific exception is negotiated) -- same for cell phones.** There also might be random pop quizzes during the class as a part of participation.

## Honesty

Although I encourage you to have lively discussions with one another, all work you hand in must be your own work. If your program or parts of your program are plagiarized from another student or unapproved sources including tutorials, you will fail the course and a letter will be put in your file with Cal Poly Judicial Affairs. Note some old tutorials do not use modern graphics – if you use them, this can result in problems. You can talk to one another about your solutions and you may look at another student’s code that has a bug (I encourage you to help each other with de-bugging), but you cannot look at someone else’s working code.

Note that I expect your OpenGL code to conform to at least OpenGL 3.0 standards (sometimes referred to as “modern graphics”) some specifics include no use of immediate mode for rendering and no OpenGL matrix stack calls (instead we will be using glm for a matrix library) and all shading will be computed using GLSL shaders. Your code must compile and run on the machines in the CSL.

## Disturbing Behavior

Students with continuous disturbing behavior will fail the class in order to restore a reasonable learning environment for all the other students

Only emails beginning with a proper greeting and a corresponding ending will be answered.

[1] http://www.theatlantic.com/technology/archive/2014/05/to-remember-a-lecture-better-take-notes-by-hand/361478/

[2] Applying the Seven Principles for Good Practice in Undergraduate Education" (1991) Chickering and Gamson

[3] http://www.yorku.ca/ncepeda/laptopFAQ.html

## Schedule

The following schedule for the lectures and assignments may change – it is meant to serve as a basic outline of topics and schedule

| | | | |
|---|---|---|---|
| Week 0 | 9/14/17 | Introduction – the graphics pipeline | Labs1: rasterizer start |
| Week 1 | 9/19 | Rasterizer (lines and barycentric) & meshes | Lab2: rasterizer triangle |
| | 9/21 | 2D coordinates and coordinate transforms | Program 1 due (Sat) |
| Week 2 | 9/26 | openGL pipeline & GLSL shaders | Lab3: hello OGL |
| | 9/28/17 | More GLSL & Points and vectors & planes | |
| Week 3 | 10/3/17 | Parametric functions and generating geometry | Program 2A due – shader |
| | 10/5/17 | openGL pipeline & Geometric transforms I | |
| Week 4 | 10/10/17 | Geometric transforms II | |
| | 10/12/17 | hierarchical modeling & review | Program 2B due – xforms |
| Week 5 | 10/17/17 | **Midterm 1** | |
| | 10/19/17 | Lighting and Shading I | |
| Week 6 | 10/24/17 | Lighting and Shading II - Texture mapping | |
| | 10/26 | Textures, frame buffers | |
| Week 7 | 10/31 | Textures, frame buffers | Program 3 |
| | 11/2 | Viewing transforms and Camera I | Final proj. proposals due |
| Week 8 | 11/7 | Viewing transforms and Camera II | |
| | 11/9 | Final project grab bag (animation, collisions, special effects, etc.) | |
| Week 9 | 11/14 | Geometry and animation | Program 4 due |
| | 11/16 | Clipping, Culling, hidden surface removal | Final project check-in 1 |
| Week ? | 11/20-24 | ***Thanksgiving break week*** | |
| Week 10 | 11/28 | Review | Final project check-in 2 |
| | 11/30 | **Midterm 2** | |
| Final | 12/7 | Thursday 1:10-4pm | Final Projects demo |

## Some random things

“What is Creative Computing? (from https://creative-computing.appspot.com/faq)

Creative computing is about **creativity**. Computer science and computing-related fields have long been perceived as being disconnected from young people's interests and values. Creative computing supports the development of personal connections to computing, by drawing upon creativity, imagination, and interests.

Creative computing is about **computing**. Many young people with access to computers participate as consumers, rather than designers or creators. Creative computing emphasizes the knowledge and practices that young people need to create the types of dynamic and interactive computational media that they enjoy in their daily lives.

Engaging in the creation of computational artifacts prepares young people for more than careers as computer scientists or as programmers. It supports young people's development as computational thinkers – individuals who can draw on computational concepts, practices, and perspectives in all aspects of their lives, across disciplines and contexts.”