Course Syllabus
CS441/541: Game Development for Mobile Platforms
Fall 2019

Instructor and Teaching Assistant
Instructor: Patrick H. Madden
Office: Q04, Engineering Building
Lecture hours: MWF 9:40-10:40
Lecture classroom: ENGB N1 (?)
Office hours: MWF 11-12, and by appointment. T/Th morning hours are usually available.
Email: pmadden@acm.org
Webpage: www.cs.binghamton.edu/~pmadden/courses/cs441/index.html

Course Description
This course focuses software development for mobile computing platforms, such as smartphones and tablets, with an emphasis on games. Students will develop interactive applications, and utilize the wide variety of sensors and networking features available on the platform, along with basic elements of graphics programming and animation. The course also covers the mechanics of distributing software for mobile computing platforms. Both iOS and Android operating systems will be covered. The course will feature a mix of individual and team projects. Prerequisite: CS 140 (Programming with Objects), CS 375 (Design and Analysis of Algorithms)
The course is designed to be hands-on, with lots of coding. Lots of coding. Let me stress this point again. Lots of coding. If you do not like coding, you will not have a good time in this class. In fact, you will have a very unpleasant time. There will be a number of programming projects; if you need a language manual open in your lap while you are writing code, this is not the course for you. Students are expected to be comfortable with C or Java – Objective C will be covered in the lectures.

There will be both individual and team projects. All code will be submitted through GitHub. The GitHub commit messages will be checked, to make sure everyone is contributing to the group projects.

Prerequisites
CS140, CS375, and competence in at least one programming language.

Course Objectives
This is a hands-on course, focusing on applying computer science theory to practical programming tasks. Much of the focus will be on games (because they’re fun to make, and fun to play), but classical computer science comes into play too.

Textbook
- All course materials are available on-line. The course web page will have links to resources.

Main Topics
- Basics of setting up Bitbucket and using Git.
- Source code versioning, diff, synchronizing repositories.
• Creating projects in Xcode and Android Studio.

• Model-View-Controller paradigm.

• Threading and event-driven programming.

• Basic computer graphics (drawing lines, shapes, etc.)

• Game-style physics (collisions, faked gravity, etc).

• Basics of frame animation.

• Graphic asset management.

• Team coordination.

• Entrepreneurship.

During the course, programming projects will include developing a number of simple games. Projects may include clones of breakout, asteroids, space invaders, 2048, flappy bird, angry birds, and more. Towards the end of the semester, teams will be able to select projects, and work to complete them. Students are encouraged to be creative (but keep the creativity tasteful).

Lecture Notes
Much of the class will be handled in a “flipped” style, where work will be done during class. Students are encouraged to bring laptops, and to follow along with code being developed as part of the lecture.

Grading
1. Your grade will be based on

• There will be 7 major projects. The first 6 are 10% each; the last project is worth 20% of the grade.

• We will have a number of small coding tasks, roughly 10, worth 2% each. On Fridays, the students will be randomly paired, to check through each others coding tasks – think of this as practice for code reviews that you might do in a commercial software development job.

• For each project, you will need to create a Git repository, and make regular commits to track your progress. To stress this point: much of the grade will be based on documentation that you have in fact worked on the code yourself. The score for any project will be computed as follows:
  10%: complete README.md
  10%: screen shots of the project
  10%: a Git status log, showing when you made commits, and what was done.
  40%: you must make commits over multiple days. For each project, you should have at least one commit for half the number of days of the project. Do not put all the work off until the last minute. If, for example, there are two weeks for a project, you should have commits on at least seven different days. For each day you are short, 10% will be deducted (up to the 40% for this part of the grading).
  10%: review and evaluation of other projects. After an assignment is submitted, a summary of all work will be on the web – take a look at the work done by your peers, and provide an evaluation. 10%: originality; based on the peer review, some assignments will be awarded additional points. 10%: the top 5 evaluated projects will present their work in class, for up to six minutes per project/team. These presentations will be evaluated, gaining from two to ten points.
For letter grades, the A/A- range is typically 90 and above, Bs will be in the 80-90, Cs are 70-80. Ds are unlikely; if you cannot perform at an adequate level, an F will be assigned. An F is generally easier to replace on a transcript than a D, by taking the class again.

Individual project assignments will be due at midnight (checked into Github). The code must compile and run correctly for any points to be awarded. The Github status log will be used to confirm that you did the work yourself. On team projects, all team members must have commits (and follow the at-least-every-other-day policy).

Some teams will self-organize, others will be randomly selected. Being able to talk to people you don’t know (and might not even like!) is a useful skill!

**Submitting Code**

You should use Github for your code repository. Name your projects p##_lastname – in other words, Prof. Madden would call his first project p01_madden. Naming in this manner will make it simple to download and organize the projects for grading and sanity checking. Each project should have a markdown Readme file, and add the user “profmadden” as a contact – so that Prof. Madden can see the code.

**Using Code**

There’s a vast array of software on-line. Go ahead and look at tutorial or example code, but MAKE SURE YOU CLEARLY STATE WHERE IT COMES FROM. INCLUDE THIS IN THE README FILE FOR YOUR PROJECT. There are lots of great skeletons of apps from both Apple and Android. Being a productive mobile app developer doesn’t mean you have to write absolutely everything from scratch, in a complete vacuum, with no outside information. Go ahead and look at things. BUT MAKE SURE YOU STATE CLEARLY WHERE YOU GET THE THINGS YOU FIND. Credit where credit is due! For each of the projects, make sure there’s a substantial contribution from YOU – not just cut-and-paste replacement of the names in the comments, or changing a few variable names here and there. And read through the code you find carefully, so that you understand how it works. Reading code is an excellent way to learn how to do things.

You are also encouraged to help your classmates, answer questions, and point them in the right direction if they’re lost. The goal of the course is to have everyone build the skill they’ll need to be able to create apps. I’m sure you know the difference between helping someone, and doing their work for them. Go ahead and help. But don’t do their work.

If one of your projects uses a lot of “borrowed” code, and you don’t clearly disclose where the code comes from, you’ll lose all the points for the assignment. Depending on the severity, you may fail the course, and also have a little chat with the dean of students.

**Intellectual Property (and selling stuff on the app store)**

It’s easy to set up an account with the app stores, and to sell the apps that you create. It assumed that everything that you write for this course will not be sold (or will be free if placed on the app store). If you desire to sell an app that you wrote – you may use YOUR OWN code developed during class, but not that of other students. If you have a team that you like, and you wish to
collaborate on apps that you will sell – this is outside of the class (and discuss with your team mates what your expectations are).

While not a course requirement – students are encouraged to think about starting their own software companies. App stores are easy ways to get your “product” to market, and it takes a very small investment to get set up to create apps.

During the semester, there will be guest lectures from professors in the business school to talk about entrepreneurship, marketing, intellectual property, and so on.

**International students:** check carefully on your visa status if you wish to sell any apps in the app store. Making money may invalidate a student visa.

**Reading Assignments**

As we go along, we’ll focus on different APIs in the various SDKs. Read through them, and look at example applications, so that you’re ready to knock out some code.

**Academic Honesty Expectations**

Please review the academic honesty document and make sure that you understand it! The link is at: [http://www.binghamton.edu/watson/about/honesty-policy.pdf](http://www.binghamton.edu/watson/about/honesty-policy.pdf). Cheating and copying will NOT be tolerated.

**Collaboration**

Students are encouraged to help one another and to form study groups. In Computer Science, you can learn more from your peers than from your instructors and teaching assistants. As long as the help is appropriate, please be generous with your time and expertise when helping fellow students. Doing so is good for you and good for them.

But keep in mind – the ultimate goal of the coursework over your time at Binghamton is to prepare you for a career in computer science. After graduation, there will be interviews – where seasoned experts will pepper you with questions, and test the limits of your knowledge. There will be no way to succeed, other than having a firm grasp of all the material. There will be no opportunities to get help from a friend, and interviewers will not be willing to cut you any slack, or give you a second chance. There will be heavy competition for the most interesting opportunities; your best bet is to work hard to develop your skills to their fullest potential. While letter grades on a transcript might seem like the most obvious goal, it will in fact be your skills that ultimately determine your career trajectory.

**Computers and Other Electronic Devices**

Bring your laptop! Write code! Write lots of code! Play the games you write (but don’t spend so much time playing the games that you don’t have time to write code). Writing code is a lot of fun. Often, it’s more fun that playing a game.