Course Syllabus
CS441: Game Development for Mobile Platforms
Spring 2016

Instructor and Teaching Assistant
Instructor: Patrick H. Madden
Office: Q04, Engineering Building
Lecture hours: T/Th 6:00-7:25,
Lecture classroom: Student Services Wing 321
Office hours: T/Th 4:30-6pm, and by appointment
Email: pmadden@acm.org
Webpage: www.cs.binghamton.edu/~pmadden/courses/cs120

Teaching assistant: Kevin Di Benedetto
Office: TBD – check web page

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 Bitbucket. The Bitbucket 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.

**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
   • Individual projects (30%)
   • Team projects (30%)
   • Source code presentations and walk-throughs (20%)
   • Final team project (20%)

Individual project assignments will be due at midnight (checked into Bitbucket). The code must compile and run correctly for any points to be awarded. The Bitbucket submission timestamps will be checked – each project must have AT LEAST two commits, and these commits must be AT LEAST 24 hours apart. If the code magically appears in a 3 minute span, it seems unlikely that you wrote it yourself…. A correctly working project will receive a score of 80%. The remaining 20% will be “style points” – make the project interesting, put a little bit of personality into it – if you’re going to make a successful game, you have to make people want to play it.

Team projects have a similar 80% initial score – but with a few added constraints. Each team member must have at least two (substantial) commits to the project, and these must be at least 48
hours apart. Adding your name is not a substantial commit – but adding a few dozen lines of code is. For each team member that does not take part in the project, the score will be reduced by 10% -- there is an incentive for team members to get everyone on the team to participate. Any team member that fails to have two substantial commits (and 48 hours apart) will receive a zero for the project. The “style points” for team projects will count for 20% -- and the teams will give 5 minute presentations to the class on their projects. Prof. Madden will add up to 10% for style, while class voting will determine the remaining 10%.

During the semester, every student will need to prepare a 10 minute presentation showing an API from either the Xcode or Android SDKs, and give a short demonstration. Being able to describe code to team mates is an important software developer skill.

There will be one larger team project towards the end of the semester, which accounts for 20% of the overall grade. The intention with this project is to submit the app to appropriate app store (as a free app). Make it good.

Teams will be randomly selected. Being able to talk to people you don’t know (and might not even like!) is a useful skill!

Using Code

There’s a vast array of software on-line. Go ahead and use tutorial or example code, but MAKE SURE YOU CLEARLY STATE WHERE IT COMES FROM. 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 everything from scratch. Go ahead and use/reuse what you find. 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 non-trivial 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.

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).

If you think of a clever game idea – this would not be a good topic for a course project. For the class, focus on the well-worn classics. If there’s a spark of genius – create the app, and do it on your own!
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.