UMBC URCAD Remarks
Joshua Barczak
4/25/2012

I'm in an industry that is notoriously difficult to get into, and a lot of people think this is because game studios are like enchanted cloisters, in which you get paid large sums of money to play games and drink free beer. I am happy to confirm for you … that … I have, on occasion, received free beer. I have also played games on company time, which, when they are unfinished, is not as much fun as you might expect. When you learn the reality of this business, you end up wanting in less for the fun, and more for the challenge. I'm going to be talking today about how my UMBC experience helped prepare me for this. I'm going to start, though, with a brief introduction to my employer.

Firaxis is a small company, around 120 people, located just up the road in Hunt Valley, Maryland. We're an established studio, and we're known best for making thinking games. Our gameplay is focused on resource management, planning, economics, and strategic combat. In our particular pond, we are a very large and well respected fish. Our games are fun, they're commercially and critically successful, and our players find them so addictive that it inspired our marketing folks to invent a fictional support group.

I'm a relative newcomer, I've been there a little over three years, and worked on one shipped title (Sid Meier's Civilization V), which has the distinction of being the only game I know of to have a state holiday declared on its release date. The game has been a great success, I'm proud to have been a part of it. I also want to avoid claiming too much credit. Please don't make the mistake of referring to me as "the guy who made Civ5", I am only one small cog in a grand machine; dozens of people were involved. In particular, you'll be interested to know that UMBC is very well represented, both on this project and in the company at large. In fact, of seven people at Firaxis who do my job, four are UMBC alumni.

I have found it difficult explaining to people what I do. When I first tell them where I work, they're intrigued, and I'm typically asked something like: *"Are you the one who designs the games?"* which isn't right. So, after further explanation, I might be asked *"So… you make pictures… with the computer…are you a graphic designer?"* which also isn't right, I'm not an artist of any sort. I am actually a software engineer, which means that my job is to write a long, tedious sequence of instructions for a computer to follow in order to produce some useful result. My specialty is computer graphics, which means my software is in charge of producing the images you see on the screen. A brief description might be:

- Start with 3D objects which an artist makes, and a designer places in game
- Figure out which of these you can see
- Animate them, and calculate their projections on the screen
- Calculate pixel colors (which touches on physics and signal processing)
- Do some image processing on the results

In order to make the game fluid and responsive, this needs to be done between 30 and 60 times per second. My job is engineering, a bit of applied physics and math, and a certain amount of "creative trickery," because getting it to go fast enough often requires cheating.

While it's a bit of a stretch to call me a scientist, I can think of two ways in which research skills impact my job.

The first point is very practical: **Fixing any software bug of sufficient complexity involves the scientific method.** Software development is extremely iterative, lots of people are making changes all at once, and they will make mistakes, and as a result, things get broken, which must be fixed. When something breaks, your first question should NOT be: "*How do I fix this?"* If you begin with this question, you will end up hiding the bug, and it will lay eggs, and over time, you end up with an infestation. Rather, your first question should be: *"Why did it break?"* This in turn leads to other questions:

- "What conditions are present?" (observation)
- "What other conditions must be present?" (deduction)
- "What could cause all these conditions?" (hypothesis)
- "If it were causing them, how could I know?" (testable prediction)
- You repeat this until you fully understand the problem, then fix it.

My second point is more general, and applies in virtually any field: **Problems of sufficient scope do not have ready-made solutions.** In real work, there are no books or lecture notes that will tell you how to solve your exact problem, because the right answer depends on your particular needs and circumstances. Any written resources you do have will have limits:

- They may leave things out. I've found that CS journal papers tend to be very light on details. They will give you just enough information to understand the core ideas and contributions, and leave it up to the reader to fill in the particulars.
- They may be "fluffed up" and be less effective and less widely applicable than they first appear.
- They may be outright wrong. There may be some failure case which the authors didn't encounter or think up, but you did. When this happens, you will have to fix it all by yourself.

You MUST be able to take what others have done, understand it, do a critical analysis, and synthesize a new solution. To this end, a fundamental understanding of your subject is much more important than the accumulation of rote knowledge (especially now that we have Google.)

One thing that will distinguish your research experience from your coursework is that it is the first time in your education when neither you NOR your professor knows what the answer is. Your professors can give advice, and they may have some vague ideas, but working it all out is your responsibility. This is an inflection point, because you've now transitioned from simply learning the answers to **discovering** the answers, and in the

process of discovering them, you are learning the general art of discovering answers. I went to share three specific lessons I learned when I went through this.

First: **Learning how to think outside the proverbial box.** To explain this part, I need to give some technical background. My *UMBC Review* article was about trying to use a graphics accelerator as a general purpose processor, which is something that it wasn't designed to be. This is becoming very important in the HPC space, and it now has its own acronym (GPGPU), and some reasonable development platforms and tools. In 2003, this was not the case, and the methods we had to work with, by today's standards, can only be described as *clever, convoluted hacks*. In fact, at this point, we hadn't even discovered all the hacks yet. Some really goofy ones remained to be published. This was early, exploratory work that was being done to try and see what these machines might be capable of, and what their limitations were.

Working in this very unusual way taught me a lot about generalizing. I learned how to map what I needed done onto what I had available, and to draw connections between things that didn't have any obvious relationship to one another. The ability to do this leads you to a wealth of new ideas, and occasionally, some of these ideas will even be good ones.

Second: **Learning how to learn the details without being taught.** I was learning how to use programmable graphics hardware from scratch, which was something I'd never done. I had to learn on my own. Learning to use the hardware was not the goal of my work, it was just a necessary condition for reaching that goal. This was excellent practice for my job, and I got a lot of key technical experience from it. I was using programmable hardware at a time when it was brand new, and when very few people (even the vendors) really knew what to do with it. I think that this, in part, led me to my first job, working for the vendor of the hardware I used, in a group which was engaged in very similar work (finding new ways to use the hardware, in order to sell more of it.) Gaming is the dominant market for graphics hardware, which meant that I got a lot of exposure to how graphics is done in the gaming community. When I decided to move over into the game industry, I was able to come in with a skill set which that industry needs and values.

Third (and, perhaps, the most cliché): **Learning that I love this stuff.** When I was a kid, I liked drawing pictures. But I was terrible at it, and I knew it, and was OK with it. After all, there were other things I was better at (like computers). Then, at the end of college, I took a graphics class, and I had this homework assignment that was spitting out pictures:… and the lines were straight,… and the lighting was right, …and the perspective was right, …and the shading was smooth and vivid…and I discovered that I could draw perfectly. I had the power to make a perfect image of anything I wanted, using only math and electricity, and the more I learned, the prettier my pictures would get. As I was thinking these things, I realized that something had shifted. I was now doing HOMEWORK for FUN, and this changed my priorities. I stopped thinking about graduating and trying to sneak into game development, and started thinking about going to graduate school so I could keep doing what I was doing. Paradoxically (and

unusually), I only got the gaming job because I put it aside for a while to pursue something I liked even better.

I'd like to end with two "take-home messages:" one for the faculty, and one for the students.

To the faculty, I would say that if you have students who are like I was, who both like and excel at your subject, then a small investment in attention, encouragement, and gentle prodding can make an enormous difference in their careers and lives. When I was applying to graduate school, I was applying at the last minute, and I had doubts about whether I belonged there, or was good enough to get in, but Drs. Olano and DesJardins both seemed to know better, and both took the time to make sure I knew it. Without their involvement, I probably would not have continued my studies, and my life probably would have been a lot less exciting.

To the students, I would remind you that UMBC, through this program, has given you an opportunity that students elsewhere often don't get. You are all here because you've chosen a subject, presumably your favorite, and are getting the chance to test drive that subject in a controlled environment, with a safety net.

If, at the end, you are happy that it's over, you should find a new subject, even if that means a change in plans.

If, at the end, you regret that it's over, then you should do whatever you can to make a career of it, in whatever form that takes for your particular field: (graduate studies, auditions, grants, whatever it may be.) You should make this your objective, even if that means a change in plans.

Changes in plans may entail sacrifices and personal risk, and you do need a means of support. In particular, if you are aiming at an industry like mine, where there is fierce competition for a small number of spots, then prudence demands that you have a secondary goal in mind. But don't lose sight of the fact that it is ONLY your secondary goal, and is only a means to the end of doing what you love.

You are going to be working at your day job for the vast majority of your waking life, and it is best for all concerned if your work doesn't feel like work. When your work is not like work, you will be happier in your work, and you will be better in your work.

That's it. Thank you very much, Good luck, and congratulations.