Division II: Retrospective Essay

Isaiah Mann

Intent

I started my Division II to make games and become a computer scientist. At the start, I was relatively inexperienced with both game development and computer science. I'd taken two courses in computer science, one of which was game programming. There was something in coding that inherently spoke to me. Quite possibly, it was the simple, mathematical logic. Back in high school, I was always drawn to the algebraic mode of thinking, a mode that is often applied in coding. All code can be broken down into inputs and outputs. Every component is an algebraic function when abstracted. Back two years ago, I had yet to frame it this way. My main motivator was the success I'd found in my two computer science courses thus far (*Intro to Game Programming* and *Web Development I*). In *Intro to Game Programming*, I initially struggled a to learn basic principles and techniques (e.g. for-loops and conditionals), but I aggressively attended TA and Office hours and gradually became one of the more adept coders in the class. In Web Development, I'd also opted to create a game for my final project. As one of the few students in the course with prior programming experience, I did very well with the material and my project stood out among other more static web pages from the class. These successes drove me to concentrate in computer science and game development.

When I initially conceived the concentration, I envisioned creating large, grand scale games. I expected to become versed in a variety of languages and techniques, and write many different types of programs. Originally, I also intended to pursue a Master's Degree in Computer Science.

I was interested in human-computer interaction: looking at how the users interact with programs. I was also initially more focused on visual media. My Division II began as a broad focus on visual art and media. I was interested in creative technologies and collaborating with artists. I was also considering comic book studies, an area I'd taken two courses on in Division I. This was not a study I ended up taking further courses, but I was able to extrapolate the skills and techniques to a course that looked critically at video games as texts.

My Division II has been a transformative experience. I knew from the beginning I wanted it to involve computer science and game development. However, upon beginning, I focused almost exclusively on these two areas. I've taken some courses with only a tangential relevance, but a bulk of my coursework have been dedicated to learning how to program and make video games.

New Learning

More than anything, my new learning has centered around programming, as a bulk of my coursework has been in computer science. I've also learned about the game development industry through my more recent courses, internships, and projects.

When it comes to computer science, I've learned a staggering amount about object-oriented programming. I've also learned more about Unity and C#, the primary development environment and language I've been using. I've dabbled in other fields of study, but it's these two staples: Computer Science and Game Development that have continually been highlighted and crucial in my work.

In addition to learning Object-Oriented principles, I've also spent some time with functional programming, using Clojure, I've learned about the Model View Controller architecture through a database driven websites course. I've worked with web frameworks, done web development work, and worked at a lower level with assembly code and basic hardware. The most complex hardware project I completed was building an 8-bit processor. I've learned about asymptotic analysis and Big-O notation.

However, object-oriented programming has been my key focuses, and has driven and evolved how I code. Before I embraced a methodology, coding seemed clunky and unwieldy. I initially saw scripting the exact interactions you expected to see as the only possibility. Each programmer needs a paradigm to operate in that separates them from hardcoding and transforms them into engineers. My first proper exposure was in Programming with Data Structures at Smith College. This course introduced me to Java and a variety of new data structures and algorithms. I still remember the most difficult assignment in the course: building a compressed array that recognized consecutive repeats of an element and tracked the number of recurrences instead of storing individual elements. For me, this demonstrated potential I'd never previously imagined in programming. For one, I saw how a simple data structure, e.g. an

array could be transformed and mutated into something more powerful and modular. I also saw how a data structure could be designed to operate in an outwardly simple manner (like the aforementioned assignment), but internally contain a great deal of complexity. I realized how simple concepts could be combined and layered to form highly advanced procedures. This semester was the peak of my proficiency in Java to date. I finished two separate final projects in a matter of days (though partially by necessity of timing). I was beginning to grasp the core concepts behind constructing complex programs: layering simple design patterns to create a powerful system.

Advanced Programming Technique was the course that pushed my learning of object-oriented programming to its current point. At a mechanical level, the course used JavaScript —which I had always assumed strayed dramatically from object-oriented principals. However, this course demonstrated how object-oriented concepts could be applied to even a loosely typed language like JavaScript. This course also demonstrated modularity and compartmentalization in a comprehensible fashion. Through my work in this course, I learned to write small, self-contained classes. I endeavored to write classes to be individually simple, but powerful when combined together. In addition, I learned to only to give class the information they require, ensuring balanced distribution instead of a single omnipotent class.

One of the lessons I learned very early on but am still finding new ways to implement is never to hardcode variables. While this is one of the first and most basic rules of programming, it has still been one of the most challenging. The core reason I find it so difficult is that all programs requires a certain level of specificity to operate. Otherwise, with everything genericized, you have a system that is highly powerful but wholly ambiguous. I've tried to find the best practices to minimize hard coding. I realize that end user input is key. And beyond that, enabling non-technical input from the developers. Something I've sunk a great deal of time and code into is manufacturing techniques for parsing and reading in text so that a non-technical developer can influence and shape a program's behavior.

I find the idea of accessibility in writing systems to be a driving factor. I strive to write code that can be of use to a non-technical user (e.g. building support for text parsing). However, I also believe it's important to write code that is useful to you as the engineer and other developers. One of my biggest pet peeves is engineers that design whole systems without bothering to provide documentation or clear instructions for use/modification. I personally think that the most advanced system is still useless if it's wholly uncommented and inaccessible to other coders. Another technique I learned from my Advanced Programming Technique class

was to name variables and methods expressively. As the texts and professor explained, comments will often become obsolete and are impossible to fully maintain. However, the code is always an up-to-date reflection of the system. Furthermore, I consider the idea of modularity incredibly important. In January 2016, I took on a project that was too large to complete in the timespan. As predicted, I didn't finish the project. However, I wrote many utility and helper classes in an abstract, non-project-specific format. This has allowed me to reuse those scripts across multiple projects. Spending the extra time to write modular code has been exponentially helpful in future projects.

In coding, I try to be aware of more than the current project. I look at how the systems I'm developing can be applied to future projects and serve my overall evolution as a programmer.

For as much time as I've spent worrying about code and scripting long into the night, I've spent an equal amount of time in game development. The first fact I learned about Game Development (this fact came from Ira Fay, now my Advisor) was that what many mistakenly call "Game Design" is actually "Game Development" This is a key distinction as I've been a developer for a great majority of my time at Hampshire, but rarely designer. My time in game dev has been spent on programming and project management. And through these, I've learned a great deal more about all fields involved. This collaborative nature is one of the most inviting features of the field. I've worked alongside talented artists, musicians, writers, designers, business professionals, producers, and quality assurance testers, many of whom I have little in common with in terms of skills. Despite differences in skillsets, there is a unifying passion for the medium of videogames and their creation.

My first team project was Word Snack HD. My first semester of Division II, I signed up for Interdisciplinary Game Studio: my the first collaborative programming and game development project. I was accepted as a member of the programming team. Having already used Unity, I found that I was a strong, contributing member of the team and the course. My prior experience put me ahead of the learning curve on the programming team. At times, I found it difficult to keep my work contained and share responsibilities with the other ten members of the programming team. However, the final result was successful, and is currently available for download on the App Store and Google Play. Through the process, I learned a key lesson: the most valuable breed of programmer is not a lone wolf who writes all the code themselves. The ideal programmer is able to hold up a balanced portion of the work, and support others when they're struggling with tasks and bugs. I slowly evolved from a lone wolf coder to a more

collaborative one as the semester continued. I still have my moments to this day where I'm less a team player and more of a loner: opting to write the code solo instead of collaborate with others.

Collaboration is something I often still struggle with. When to be authoritative. When to take step back. When to be quiet and listen. When to be a supportive member. When to pause and take a break. Game development forces close collaboration under high pressure. I've seen many issues arise due to this pressure. Alternatively, I've also seen great success in collaboration, and found it to be one of my favorite areas to work with others.The aforementioned love of the medium keeps everyone engaged and pushing forward.

My most eye opening experience in game development was the MassDiGI Summer Innovation program. It focused on the industry side of video games and technology more than I'd experienced before. I was introduced to the concepts of networking, professional email, monetization, and publicity.

I was assigned as the producer of my team's game, Crafting Life. I worked with a team of two other programmers, three artists, and one audio designer. Everyone on the team was friendly and hardworking. There were some issues in terms of skills: both on the programmers end and the artists. However, any failings in prior experience were compensated for by the team's optimism and determination.

I struggled at times with leadership and being authoritative. However, overall I consider MassDiGI a valuable experience. Being in charge of a project encouraged me to assume greater responsibility, increase my communication, and develop decision making skills. While I struggled throughout the summer, it ultimately encouraged me to take on more leadership challenges in games this semester. It also inspired me to start a concept that would evolve into my Division III. MassDiGI kept me very busy throughout the summer. Whether I was programming the game, coordinating amongst team members, presenting the game at demo events, or attending lectures by game developers, I was working very consistently throughout the summer.

This sense of urgency and constant work has followed me through the remainder of Division II. I can confidently say as hard as I worked my first year, I've doubled the capacity of work and responsibilities. At times, it comes at great cost to my rest, happiness, and stability. Overachieving is something I've always struggled with, but it's reached a new extreme. Partially in the number of responsibilities I've taken on in recent semesters.

This may seem grim and dramatic, and at times it feels that way. But I feel there's important new learning to be found here. One of the key discoveries I've made is that I constantly need a challenge to focus on. Challenges keep me pushing forward. Given what I've seen of my drops in productivity and liveliness over breaks, I much prefer being swamped with work to having none at all. On the other hand, I've also come to the realization that I often intentionally overload myself with work to ensure I'm constantly in a state of stress. This is clearly not something I was able to remedy in Division II. This final semester is possibly the busiest I've ever been. However, I was given an important piece of wisdom by my committee chair, Lee Spector: "Depth is superior to breadth." This piece of wisdom is easy to remember as I tend to associate it with breadth and depth first search (programming algorithms to navigate trees and graphs). The key argument in this aphorism is that it's superior to specialize in a single focus than to spread oneself too thin. I entirely agree, but only through having experienced what it's like to be spread far too thin. It's been extremely challenging keeping up on so many different tasks and responsibilities. In part, I realize I have a knack for multitasking. Despite the colossal amounts of work, I've continued to maintain a steady output and not cave under the responsibilities. However, I'm also keenly aware of the diminishing returns and will strive not to make these same mistakes in planning my future.

Time has been a concern from the start. My courses first semester: Interdisciplinary Game Studio, Discrete Math, Computer Systems 1, AI, and Digital Art explored different areas of programming and computer science. I found all of them engaging. All these topics helped solidify my primary reason for learning computer science: was to develop games. This was also my first semester taking five courses at once. At times, it was a very rewarding experience. However, the sheer amount of work required resulted in a demanding schedule.

In addition to these courses, I was also participating in multiple extracurriculars, many of which I've retained throughout my Division II. Firstly, "The Cutest Little Freaks in the Universe," an Improv troupe I co-founded and help to organize. This was our first semester auditioning new members and performing publicly at Hampshire. I found the experience to be stressful, but very rewarding. We faced challenges in organization, interpersonal dynamics, and time commitment, but I would call our first semester as a performing troupe successful. This was also my first semester as a signer for The Omen: Hampshire's long running free speech publication. The signing duties were relatively light, but I found myself overstretched at times. A year laterI handed off signer duties.

Fall 2014 being a semester of firsts, I was also a Dungeon Master for Deathfest, Hampshire's semesterly roleplaying tournament. I enjoyed this because it took in elements of game design and improvisation to create something theatrical and unique. The organizing leading up to the event proved stressful as many of the student organizers were not readily available or communicative. Despite not being in a leadership role, I found myself doing a large bulk of the work: ordering food, making and hanging up posters, reserving space, etc.

During the Spring Semester, I furthered my work in Multicultural Perspectives taking Marisa Parham's Video Games and Boundaries of Narrative course. This course looked at video games from the perspective of the Humanities: observing their cultural significance, social commentary, and impact on the world. In addition, I also took further courses to build my programming skills: Database Driven Websites, Introduction to Computer Science II, and Programming with Data Structures. Programming with Data Structures, which I took at Smith with Professor Eitan Mendelowitz, proved to be one of the most valuable courses I've taken in programming. It showed me the power of object-oriented programming. Including the usefulness of inheritance, overloading, and abstraction. I also took Calculus in Context at Hampshire in an attempt to bring my Math Skills up to par with my programming skills.

In the three math courses I've taken during Division 2: Discrete Math, Calculus in Context, and Linear Algebra, I see potential correlations to programming; however, the concepts are introduced too abstractly for me to fully invest in or immerse myself into. The most interesting point in Calculus was working with RStudio, but even my impressions of it were limited. I've been concerned that a lack of math knowledge will negatively impact my performance as a programmer, but so far, I have only found that to be the case in the most theoretical of computer science work.

In terms of math and programming, I believe they share many core principles: abstraction, variables, formulas. However, to excel in one, does not necessitate the other. As a programmer, I feel very accomplished for someone who's only been coding for three years. I also feel fully equipped mathematically as a software engineering using only arithmetic, basic algebra, and trigonometry.

My coursework has focused more on core computer science than mathematics, something which can be seen again in my Spring semester. It was comparably challenging to my Fall Semester, but while I was still taking five courses, the demands of the courses were lower than the first semester. I also continued with all the aforementioned student groups. By

the end of the semester, I was finding it somewhat difficult to complete multiple final programming projects, but I managed to hand everything in in a timely manner.

I returned to school early and was an orientation leader for my second semester in a row. I participated in leading optional activities in improv and roleplaying in the stylings of Deathfest. This proved an excellent way to engage with the incoming class and get students interested in on-campus activities. Apart from being an Orientation Leader, I also came away from the summer working on multiple projects. I was as the web developer for a small group of game developers known as Mustachio Games. I was also working with three other students from the MassDiGI internship on a small mobile game. Unfortunately, I would have to abandon both of these projects as the semester wore on, due to other time commitments.

Fall 2015 was easily the busiest I'd been thus far. I got very little sleep and was constantly booked the entire day for meetings, classes, and schoolwork. My course load was considerable: I was taking five classes myself and TA-ing for a sixth course. In addition to this, I was developing a concept for a student game development studio that would eventually become GlowLime Games. I continued to participate in improv and Deathfest, despite being very stretched for time. I also began attending meetings for Hampshire's standup club and performed in a couple shows throughout the year. On top of these many commitments, I was also interning at Petricore Games, a startup I'd encountered through MassDiGI. They're both based out of Worcester, MA. My internship with Petricore involved me commuting once a week to work at their headquarters in downtown Worcester. It was a very exhausting experience, but also rewarding to work with professional game developers who were fresh out of college.

I attended three different game development events during my fall semester: Boston Festival of Indie Games, Unite Boston, and Gameacon. All three were excellent opportunities for networking and seeing more of the industry. Networking is a skill I'm still developing and something I struggle with. I'm seeing that it's a crucial part of the industry so I'm trying to improve as I go.

My coursework in Fall 2015 was mostly programming related. The exceptions were Computer Animation 1 and Linear Algebra. Computer Animation I found very interesting but also very difficult. It's something I've been intrigued to learn more about for a long time, but in a semester where I was so pinched for time, I found myself loathe to devote too much to it. I also took the aforementioned Linear Algebra, and though I later saw direct correlations to programming in the use of transformation matrices, much of the coursework proved too abstract to keep me engaged. It became another casualty of a too busy schedule.

Easily the most demanding and largest scope project I've ever taken on started in this same Fall, and was inspired by MassDiGI the summer before. GlowLime Games is an educational service company serving the Five College area. I designed it to bring more game development opportunities to the Pioneer Valley and the students in the consortium.

The courses I enjoyed a great deal more were the programming related ones. Software Engineering was an interesting chance to learn more about the non-coding aspects of computer programming, namely careful documentation and planning. While I found taking the time to do proper documentation in personal projects overly time consuming, applying the principles of careful software design has proven very helpful in my work. I was also happy at the chance to open a connection between Hampshire and MassDiGI, in that two of the projects the students took on were projects developed in the MassDiGI Summer Innovation Program. My hope is that collaborations like these can pave the road for grander ones in the future. I greatly enjoyed a course at Smith of the same nature: Advanced Programming Technique. It stressed building small, single purpose classes in your program to allow for the greatest levels of abstraction and modularity. This is something I've taken very much to heart, and began incorporating into my systems as best I can. While it's also sometimes time consuming to construct the foundations of such programs, I've found that correcting errors and making modifications becomes incredibly streamlined when the program is broken down into small manageable pieces that come together into a cohesive whole.

The final course I took was Interdisciplinary Game Project. At the beginning of the course, I changed my mind from merely pursuing programming roles to also striving to be a producer. This did nothing to lighten my workload, but I found additional chances at a producing role to be challenging and rewarding. I felt a lot of pressure on the games I worked on, being the producer, or assistant producer in the case of the final game, as well as the only programmer on all three. While there were times during the summer that I balked at having two other coders to coordinate with, having to implement entire systems sp;p proved incredibly difficult and time consuming.

I also TA-ed for the first time in Fall 2015. It was for Women in Game Programming, a course comparable to one I'd taken my first year (the same course that first interested me in programming and game development). The assignments were familiar to me, but with the added component that the course stressed the role of women in the video game and software industries. The students taking the course were a diverse mix, and I greatly enjoyed the chance to work closely with many of them: assisting with assignments. I struggled at times to strike a

balance between instructing and allowing them to discover solutions for themselves, but felt that I was at least a responsive and available teaching assistant.

There were a few other projects I became involved in that proved semi-rewarding, but somewhat lacking in payoff. The first of these was attending meetings for a concept known as Tranche.me. I was brought on as a web developer, but the project never got past the planning stage. Another project was Hampshire's Hackathon or HampHack. I was initially interested in helping to plan, but quickly found the project too overloaded with students and became disengaged. I also attempted to assist in building the website, but was of little help due to a lack of communication within the team. Additionally, I assisted the same Division III student who devised Tranch.Me, Ajmal Jackson-Brown with another endeavor: MyRise: a web development workshop meant to help underprivileged youth in Holyoke, MA. I served as a teaching assistant to Ajmal in his pilot program. There was a final collaboration which was by far the most rewarding as it is directly intersected with my area of study and initiative through GlowLime: I assisted Pat King, founder of the Pioneer Valley Game Developers and an Adjunct Professor at Hampshire for Spring 2016 with planning the Five College Global Game Jam.

I initially intended for this to be a GlowLime collaboration, but the other members of the management team quickly fell by the wayside, so I ended up coordinating with Pat solo. I found him to be very easy to work with and we coordinated across the tools I'd become familiar with throughout the semester: Trello and Slack. Slack was introduced to me by MassDiGI as a closed network for instant messaging. Trello I've encountered a few times throughout Hampshire, and used it to create a digital Kanban board, a type of Agile Development we were taught, also at MassDiGI. In addition to using these tools to plan the Global Game Jam, I've also employed them in multiple game development projects and in setting up GlowLime. I've found them both invaluable for project management and productivity.

I brought a capable team of managers together in the Fall of 2015. Over time, we developed a structure for the organization and planned for the coming semester. For the Winter Term, I continued to work heavily on GlowLime Games. I made sure our website was up and running, I helped our hiring manager: Grace Barrett-Snyder build the teams and select the projects we developed, I created and maintained company social networking accounts, and I communicated with members of the management team to coordinate their roles going into the Spring Semester. Through all of this, the core concept I came away with is that acting swiftly and with confidence has surprisingly successful results.

One of the most unexpected events in GlowLime was that I became the Executive Director. Equally surprising was that the position generally suited me. I found that my passion for organization and efficiency extended beyond my own work and could be stretched to cover an entire organization. Here too, I do at times fall into the trap of lone wolfing projects: doing all the work and leaving no room for others. I still believe that speed and efficiency is a recipe for success; I've seen the proof repeatedly in my work. I believe that GlowLime is only at the place it is now because I held those core values so high. However, I've also strived to take a step back and trust my other managers to take on more responsibility. Allowing others to step up has showed me how to better balance and distribute work in collaborative structures. It seems like an obvious piece of learning, but I've struggled with it through my Hampshire career: in the my courses first year, my improv troupe, at MassDiGI, in the upper level game development classes, and now still in GlowLime. But I continue to make small steps in improving, while determining where to draw the line in still pulling my weight.

Another core concept I've come away from my work these past to years is properly scoping: being able to estimate time and work requirements and scale projects. This was most explicitly voiced in MassDiGI. Setting realistic goals for the kind of game that could be accomplished in 11 weeks. I ignored many of the warnings from the mentors and pushed to make something that was larger than possible. I remain proud of what was accomplished, but I saw other titles that were smaller receive more polish and get closer to completion. Applying this same results based methodology, I've been striving to select and push for smaller scope in the game projects I've been a part of since, and in some cases I believe it's paid off to help the game farther along than they would have gotten otherwise. In other cases, such as GlowLime, I've failed to scope properly and things have gotten somewhat out of hand, but in that case too I try to be aware of our limitations and what we're capable as students working part-time on a project. I think it's key to treat it not as a business and not as a student group, but as something that falls somewhere between the two in terms of level of commitment. Issues of scope aside, I am incredibly excited to be able to take another full year to approach this for Division III —and in a more full time capacity.

Moving Forward

I am nearing the end of Division II and about to step into Division III. The answer to what my Division III project will be is clearly in my mind: GlowLime Games. I've put so much time and

effort into it this past year (likely more than I should), and I feel as though it has so much more potential in the coming year. My main goal is to develop a polished game and make sure it ships to the App Store. Given the huge increase in time I'll have available, I really want to see this through. It feels like a great success as a programmer, project manager, and game developer to publish a game. Plus, I feel that it serves GlowLime overall. Obviously a Division III cannot be an organization. However, it can focus work that serves the organization. I feel that much of the foundational work has been done over the past year, but there is still a great of deal of work left to do in shaping the structure of the organization. We still need to explicitly lay out the protocols and rules guiding its operations and mission. We've worked hard to reduce this down to a marketable statement, and continue to polish our organizational mission as we strive to find funding. Obviously, this is a collaborative project and I am but a piece of it, but I feel proud of the work I've done individually and feel that even the work I did these past two semesters are a solid baseline for a Division III project. I hope that what I'll be able to accomplish with an increased amount of time will be considerably grander. Though I also hope to scale down on the amount of hours I work and practice a greater amount of self care. I want to exercise, sleep, and relax more. In addition, I hope to spend more time not staring at a computer screen. It's entirely possible that I will not hit all these targets, but I remain optimistic.

From what I've seen of software and game development, they are brutal relentless industries that encourage long hours (just like I'm putting in currently) and boundless dedication. Given what I've seen of my own work, I feel confident that I possess these qualities. With all this in mind, I'm happy to know that I'm already developing the skills needed to move into the professional realm. Previously, back at the start of Division II, I was convinced I had to have a Master's Degree or a Doctorate to even attempt to find a good job in engineering, but now I see that I could be just a single year from employment.

However, seeing the opportunities already offered to me as a programmer who is relatively inexperienced and has yet to even attain a bachelor's degree, I realize I now possess a great deal of the skills I need to enter a career in software engineering. This has convinced me not to pursue further work in academia but instead to begin searching for a job during my Division III and step directly into industry after I graduate. I'm incredibly excited to see the differences in industry vs. academia, but more importantly to finally begin to earn a real income.

I already feel like an incredibly accomplished engineer, given the fact that I've only been programming for three years. I'm also shocked by the amount of leadership and authoritative roles I've taken on (though at times they were forced upon me) over these past couple years. I

highly doubt I will be offered any such leadership role again soon after my graduation, and this is not a fact I mind particularly. The stress and responsibility of leadership is often more of a burden on my mind than the hardest coding problems I've tackled. Regardless, years in the future, I feel all the more prepared for rising to a senior level position in my engineering career. I'm confident that I'll have not only the hard skills, but also the soft ones. It's often said that engineers are antisocial people, but I feel as though the work I've done in computer science and game development, has forced me to interact more with people than I ever did previously. Building these communication skills and communications has been equally as valuable as the technical skills I've gained.