Exploring Skill Sets for Computer Game Design Programs

Bungie, Inc. Employees Share Their Insights

by Markus Geissler, Professor and Deputy Sector Navigator, ICT/Digital Media, Greater Sacramento Region

Abstract:

Based upon several responses from employees at Bungie, Inc., a developer of computer games based in Seattle, to both general and specific questions about the skills sets required for students who wish to work in the computer game industry, technical knowledge, especially in computer programming, but, more importantly, solid interpersonal skills, the ability to think as part of a team consisting of individuals with varying skill sets, and the strength to solve complex problems should be essential program student learning outcomes (PSLOs) for Computer Game Design or similar academic programs. To that end academic program designers should strongly consider an interdisciplinary preparation which includes coursework that addresses both technical and professional PSLOs and which may be offered by content experts across multiple academic departments.

Author's note:

In May 2019 the members of Cosumnes River College's (CRC) Computer Information Science faculty discussed whether or not to add a Game Design component to the department's course offerings, possibly to complement existing programs at nearby colleges. To better inform the discussion I volunteered to get some industry input via a family friend who works at Bungie, Inc., an American video game developer based in Bellevue, WA which is the developer of computer games such as Destiny, Halo, Myth, Oni, and Marathon. Several Bungie employees were kind enough to respond first to my initial request for a conversation and then specific questions by CRC's CIS faculty members which I forwarded to the folks at Bungie. I hope that their clear and informative responses will help guide how ICT faculty at community colleges, as well as other colleges and universities, will approach Computer Game Development or similarly named programs so that graduates of such programs can be adequately prepared for some of the many different roles that game development team members may need to play on the job.

A. Background

Many young people who grow up playing computer games are interested in creating their own games, and faculty in Information & Communication Technology programs regularly face questions about what courses students should take to prepare themselves for a career in Computer Game Design or similarly named fields. Many faculty are not sure how to respond to such requests for three reasons: The first reason is that, based on the number of students who inquire, only a small percentage would actually be able to find employment in what appears to be a very competitive field, and faculty don't want to give false hope to students who will most likely not end up working in the computer game design industry. The second reason is that, similarly to other ICT disciplines, a successful computer came design company will hire employees with a variety of skill sets, which include computer or creative arts skills, and it would be most difficult, if not impossible, for an academic program to offer courses that would students to acquire all of the skill sets required for developing computer games. And finally, even if a faculty member has spent a substantial amount of time getting to know a student, the inquiring student would likely know their interests and related skills much better than the faculty member.

If structured curriculum guidance or appropriate student learning outcomes were easily available for Computer Game Design or similar academic programs, they could serve to guide faculty responses as well as independent student pursuits. And in light of ever-changing technologies for both development and content

delivery, it is difficult for academic institutions to develop and maintain program student learning outcomes (PSLOs) in ICT-related fields. This challenge, however, might serve as an incentive for PSLO designers to rethink their approach and focus on the bigger picture by defining PSLOs that emphasize professional skills—which are also known as "soft skills", "success skills", or "21st century skills"—over technical skills. And while this article primarily draws on responses from a small group of employees at one specific computer game developer, albeit an industry-leading one, their statements should encourage those who would consider offering a Computer Game Design or similar academic program to define PSLOs and corresponding curriculum that address both technical and professional skills.

B. Defining the Computer Game Designer Skill Set

To get started please read the thoughts of Lisa Brown, Senior Game Designer, which she sent in response to a general solicitation to spend ten minutes in a recorded ZOOM session to provide some input about the skills that Bungie, Inc. employees regularly use in their capacity as a game designer, developer, or other team member.

> *Lisa Brown, Senior Game Designer:*
>
> Hi Markus! I'm really excited that you're reaching out to get this info. I don't have a lot of time right now for a real-time chat, even a short, one, would you be okay with email correspondence? To start, here's a brain dump of skills I often see lacking in game program graduates and suggestions on how to help. Caveat: I went to a liberal arts undergrad, so I realize I'm probably biased a lot there, and then did a graduate program at Carnegie Mellon's Entertainment technology, so this is where some of my thoughts are coming from. I also graduated from grad school 10 years ago so I realize times have changed and not all these suggestions may apply.
>
> 1. Critical thinking and the ability to dig into something difficult and heavy and analyze it. Part of this I think is because of the nature of school itself, where students are conditioned to "find the right answers," and so critical thinking digs can often be framed as finding "the thing I'm supposed to find" or "the thing the professor wants to hear" rather than learning the skill of analyzing and deriving conclusions for the sake of doing so. I realize this is tough, and I do not envy the work educators have to do to instill this skill! Fortunately I think there are many means to this end that could fit into a curriculum. For me I think it was stuff like taking humanities in college and having to decipher a difficult text that was outside my comfort zone at the time, like the Odyssey, and write critical analysis on it.
>
> 2. The course I took that I feel had the highest value for preparing me for work in game development was improv acting. This was a required course at CMU's ETC, and may seem unusual at first for inclusion in a tech-heavy program. However, improv delivers on a number of key skills that are absolutely critical for working in game development:
> - The ability to work in teams, react to spontaneous situations in a human way, and work with people really different from you to serve a collective cause
> - Thinking about storytelling in an interactive fashion
> - "Humanizing." Improv sharpens very specific, unique human skills that are often overlooked in programs heavily focused on tech, coding, and learning tools. You can be the most skilled coder

in the world and it's meaningless if you can't interact with an interdisciplinary team. Some places are making this a required course for CS majors! https://news.slashdot.org/story/19/05/19/0249211/college-requires-all-cs-majors-to-take-an-improv-class

3. Speaking of interdisciplinary team work - any sort of team-based project work that requires collaboration with other majors or disciplines. I realize cross-department collaboration in academia can sometimes be tricky, but the experience is so so value. Make your CS people work with artists, for example. I did a residency at Harrisburg University a couple of years ago, and one collaboration that impressed me was the game design class (which was making board games) worked with a technical writing class. The technical writing students took a pass on the rules that the game design students had for the games they made. Sometimes this cross-disciplinary thing can be accomplished by things like having programmers learn how to use Maya and having the artists learn how to script. Not to become proficient, but to gain tools in communicating with people outside their discipline and understand some of the constraints those folks may be under.

4. Written communication. A huge part of game industry work is communicating with other people, conveying ideas, solving problems, pitching. Ideas are meaningless if you can't communicate them.

5. Statistics is really useful. I also found stuff I learned in discrete math to have application in my work.

6. Iteration. This is another one of those skills that is often lacking just due to academic conditioning. Students spend their whole lives learning that failure is bad and that the final form of something is what matters. An example of this I observed with students during my residency is they would have a playtesting session for their game, and I would ask "did you get good feedback?" and they would say "yes, everyone liked the game." Of course, good feedback is helpful feedback where you learn what you can fix about your game, so this was actually pretty useless feedback. It's that failure thing, students get afraid when they get feedback on a game that it means they've done something wrong or poorly, but it's the opposite. Game development is founded in iterative process and development, so some way to get students to lean into the process, the number of iterations, would be helpful. I have no idea how to do this!

Those were my big initial thoughts, let me know if it's helpful, or if you have specific questions and I can answer them!

Please note that Lisa's thoughts are less so technical but more so about the skills to achieve better interaction and understanding between team members who are coming to the table from different academic backgrounds and experiences. And while some may wonder how someone with a liberal arts background would end up in a computer game designer role, it should make ICT (Computer Science, Computer Information Science/Systems, Information Technology, Information Systems, Software Engineering, Cybersecurity, etc.) faculty think about which skills we would need to teach our technically interested students so that they might succeed as game designers.

To learn a bit more about the technical skills that would best benefit the graduates of Game Design programs, here are the responses by other Bungie employees to specific questions submitted by CRC faculty:

A. What is the C++ equivalent (in philosophy and difficulty) to the game engine and graphics libraries that Bungie uses in their games? Classes in the Standard Template Library like stacks, queues, linked lists, etc.? Or something else?

Jami Lukins, User Experience Designer:

Hmm, I'm not actually a programmer, I'm a UX designer. I'll answer your questions as best I can however.
Bungie uses C++ primarily, but also have proprietary "Bungie Script" which works with our custom game engine. Beyond that I'm not sure.

Luke Timmins, Programmer on the client engineering side:

Right, the Destiny engine is proprietary (as was the Halo engine). After Halo Reach we basically did a from-scratch re-architecture of the Halo engine, pulling over some technology chunks but rewriting most others.
Broadly speaking, the client engine is C++, our tools are C#, and our backend services are C#.
I'm more on the client engineering side, so I can speak to that better. ☺ Many "traditional" C++ engineers find our codebase very C-like. At a high level:

- No runtime allocations – we know how much memory each platform has, and pre-allocate everything we can up front
    - This is one of the main reasons we haven't used STL (or other similar containers) – we can't control their dynamic allocations
    - Also – malloc/new are slow J
- We think carefully about our data – layout, loading, multi-threaded access, etc.
    - Some of the worst ways to tank performance is to ping-pong around memory, crushing your cache
    - It's easy to just follow the "traditional C++ advice" and end up with a system that's hard to optimize and parallelize
    - Mike Acton (Now at Unity) is a big advocate for the Data-Oriented Design movement – he's a bit more extreme than we are, but here's a good summary presentation -> https://www.youtube.com/watch?v=rX0ItVEVjHc
- We don't use a broad swath of C++ functionality
    - Minimal template use
    - Minimal polymorphism

On the graphics side (not my specialization, but I'll take a swing) we have an extensive content pipeline that tries to make it easy for artists to author shaders and have it "just work" on each of our supported platforms (each platform has their own graphics APIs).

B. How much of the Bungie code is creating classes from scratch, and how much is calls to the game libraries?

Jami Lukins, User Experience Designer:
No idea.

Luke Timmins, Programmer on the client engineering side:
The Destiny Engine does use a collection of 3rd party technologies. On the client side we use Havok for our physics simulation, Umbra for visibility/occlusion, and probably a few others I'm forgetting.
On the backend server side we use a collection of off-the-shelf technologies like RabbitMQ and Reddis.
On the tools side we use mostly off-the-self software for content authoring (Max, Maya, etc.) but proprietary tooling for hooking up imported content to the game.

C. What is the minimum education (individual courses, certificates, degrees) that current paid Bungie game developers have? The same for unpaid interns?

Jami Lukins, User Experience Designer:
We don't have unpaid interns. Our old school folk don't have degrees at all and are self-taught, most of our new folk have 4 year bachelor's degrees in computer science.

Luke Timmins, Programmer on the client engineering side:
Agreed – the majority have CS degrees, a few have related-ish degrees like physics or math, which introduced them to programming and they ended up gaining experience along the way.

Lisa Brown, Senior Game Designer:
I will just chime in and say encourage your students to NEVER take an unpaid internship. The vast majority of internships in the games industry are paid. If a studio is offering unpaid internships that's a huge red flag, unless they're like a tiny scrappy indie, and even then I'd give them the side eye.

D. What is the minimum education, with no professional programming experience, required to be hired for a paid developer position at Bungie?

Jami Lukins, User Experience Designer:
There is no minimum required education level, however the minimum skill level generally requires someone to have a 4 year degree in their chosen field. Even that typically isn't enough as hires straight out of college are extremely rare (though not unheard of). A 4 year degree and 3 years' experience is generally a good basis, but ultimately it's the applicant's skill level and personality that really matter. We always give a programming "test" in order to assess an applicant skill as well as a "white boarding" session in the interview to see how an applicant thinks on their feet.

Luke Timmins, Programmer on the client engineering side:

Agree our Associate Engineer slots open pretty rarely – maybe one every year and a half? And right, the majority who apply have CS degrees, but we do see people with 4 year degrees in related fields.
One big thing we look for is drive/initiative around learning game programming. Working on a personal project (even if it's "I built tic-tac-toe in my own engine from the ground up!) is amazing, as is building something with a middleware engine (Unity, Unreal).

E. If our community college were to offer a course in video game development, what topics, languages, applications/environments, should be covered?

Jami Lukins, User Experience Designer:
Every single project uses different tools, so it's more important that students are taught how to ramp up on new tools and how to think, but C++, C# and proficiency in the Unity and Unreal game engines are good starts.

Luke Timmins, Programmer on the client engineering side:
Big +1 to the Unity and Unreal suggestions. It's a game-changer for students that people can now freely use professional-grade engines for free! On the flip side, students need to be careful they appreciate and understand the magic under the covers that Unreal or Unity are doing for them J
Brainstorming classes:

- For complete beginners to programming and game development, I'm a big believer in Python.
- I could then imagine having intermediate courses that are teaching C# and using Unity to make games
- Lastly I could imagine a set of advanced courses that are C++ and using a more low-level library like SDL to make a game

Finally, Matt Breindel, who does simulation and foundation/engine/platform work at Bungie, chimed in at the end of the email conversation with another interesting perspective and summary.

Matt Breindel, simulation and foundation/engine/platform work:

Generally, I'll second everything other folks have told you. That said, as one of those folks who came from a different background, I'll add a few thoughts:

1. My degree is in physics and neurobiology. I learned C++ on my own in high school, but didn't do much with it and didn't really touch programming again until my first job (with the exception of some python from time to time). My first job was at Havok as a developer support engineer where I learned a ton in a short period—I like to describe it as "Navy SEALs training for game programmers". I wasn't hired based on my programming skills, I just had to pass a fairy basic programming test. The real skills that I got from college that helped me (and continue to) are around determination and comfort solving really hard problems. You need to be able to look at a problem, say to yourself "Well, I have no idea how to solve this," get over that, and solve it anyway.

2. I haven't done any hiring at Bungie, but I did a lot at previous jobs. I preferred folks with a background in math or physics, and people who were self-taught rather than CS grads. That's because I felt it showed more passion for the domain and because it meant not having to un-teach a lot of things that get taught in CS courses. Don't get me wrong, they are valuable things to know, but can also create habits that don't apply to the game dev domain (e.g., STL or object oriented design).

3. Timmins beat me to the punch with the Mike Acton video. I'd suggest any game dev programming course cover:

a. Data oriented design
b. Optimization
c. Hardware
d. Rapid iteration
e. Debugging
f. Complex code (e.g., working in/modifying unreal engine)

I'd even cover those if it meant less time on algorithms/data structures. I can count on one hand the number of times I've had to write a sort outside of an interview, but I use those topics daily. There are definitely areas where things like Big O are the most important factor (e.g., our content system deals in millions of items, so O(n) vs O(nlogn) is still important), but most of the time you have so few items (usually <100, almost always <1000) that readability, maintainability, iteration, and data access patterns are far more important.

4. I think students should study something that challenges them and that they enjoy, and learn what they think they need for a job on the side (or, if it lines up, studying it as major is great too!). Like others have said, a cool side project goes farther on a resume than any particular degree.

Of course, all that's very much influenced by the kind of engineering I do (simulation and foundation/engine/platform work), so YMMV [Your Mileage May Vary] and you should take those opinions with a heavy heaping spoonful of salt.

In summary, while technical knowledge, especially in computer programming, is certainly an important outcome for computer gaming-related programs, the feedback from those who are playing various roles in development teams at Bungie, Inc. emphasizes that students need to have solid interpersonal skills, an ability to think as part of a team consisting of individuals with varying skill sets, and, notably, the strength to solve complex problems. While community college instructors regularly hear the same "21st century Skills" requirement from partners in various industry sectors, an ability to solve complex problems may be what sets graduates apart from others who are trying to find jobs with computer game developer companies. And while one could argue that upper division courses would present the best environment for students to develop and hone complex problem solving skills, the feedback from these Bungie employees should also encourage community colleges which are considering the development of Computer Game Design programs to design an interdisciplinary course of study where students can not only acquire programming skills but also, and more importantly so, learn to solve increasing complex problems in a diverse team environment.

Related labor market information by Centers of Excellence (COE) can be found here. Note most openings require a Bachelor's Degree in addition to the stated skills.