2025-2026 GAWD "3D Animation / Game Design" Advisory Board meeting minutes.

Faculty at meeting:

- **Will Bohmann**
- **Matt Cronin**

3D Animation / Game Design Advisory Board Members:

- **JoAnn Patel** - Champlain College
- **Anthony Daniels** - River Bend Career and Technical Center
- **Curtis Aube** - Union Street Media
- **Joshua Buck** - Champlain College
- **Chris Tall** - WayForward Technologies

Updates:

- Brand has moved from Computer Animation to **Game Animation and Web Development**. It was simply time that we tied into our ever growing Game related coursework.
- After a down year **enrollment has ticked back up**, and we hope to continue this trend.
- We have started to bring back pre-Covid "3rd place" activities such as extra classroom time to socialize, collegial Smash tournaments, setup and left out social prompts such as chess, checkers, and uno games to **build peer-to-peer, and peer-to-faculty real life socialization**. Some of our students have simply been brought up on a computer/phone/video games and can't communicate human to human. Hopefully creating more of these low stress, low stakes 3rd places and activities allow for development of these communication skills.

This cycle we are looking at feedback on having a broader breadth of curriculum, albeit at a shallower depth.

A very interesting phenomena that has happened in the last 5 years have been the **shortcuts that are becoming more prevalent in the industry**. Jumping directly to libraries of premade assets, jumping to AI to get a code snippet, moving faster past the foundational lessons and knowledge acquisition - **we are embracing it**. Students are using these tools and techniques (such as AI to get to step 10, not needing or caring what step 1-6 are) outside of the classroom, and for the last couple years I have been trying to slow them down to “learn the basics”, and I don’t feel like this is the right decision anymore.

An example is our experience at the SkillsUSA National SkillsUSA Game Design competition. We have typically enforced a rule that all our assets must be created by the *artists themselves*. We have held this standard in our classroom and Vermont competitions. Conversely when we arrive at the national championships in Atlanta we are competing with teams that haven’t created any of their own 3D assets. Due to using off the shelf assets, their game ideas, mechanics, sound design, replay value, UI and pitch are more developed, and we are seemingly at a disadvantage at the minimum viable product stage.

Another example is character animation, specifically for use in our games. Why would I spend weeks perfecting a walk cycle when I could just download MOCAP data (that is free) for the jump, roll, and sprint animations I need and apply to my characters?

Think of a cooking competition where we have forced ourselves to grow our own tomatoes. We are at the point as chefs where we are just going to go buy or “download” the tomatoes going forward. I think we have arrived at the tipping point.

Continuing with that mindset we are going to lean into the shortcuts, often jumping over foundational lessons and tasks. **And this gives us pause.**

Our question to you this cycle is how are you handling this in your own spheres of influence? Will and I do feel conflicted that we aren’t doing as much of a focus on the foundational elements, but it feels like the world is accelerating and that focus on the foundation is less and less important. As a business that is advertising to students as customers, we also don’t want to be seen as holding the students back while outside of class and side projects can follow a faster development pipeline. We are following a bit of the “Move fast and break things” mindset of early Facebook.

With the volatility going on right now in our industries, it feels we have to go as fast as we possibly can to match the rest of the world, even if this means we miss out on some foundational elements. At the same time these foundational elements and workflows may not be foundational any longer - and **we don’t want to have a 2015 mindset, in 2025 and beyond.**

Thoughts?

JoAnn Patel - This is a loaded question and depends greatly on context. The idea that foundational knowledge is no longer important is misguided. AI tools enhance knowledge, they don't replace it.

In regards to course content and use of AI and off the shelf assets in the classroom, I think it depends on the goals of the projects. If the focus of a course is game design, then using free art assets is fine, because the goal is not about creating art, it is about solid game play mechanics. If the goal of a course is learning to 3D model, texture and animate, then learning the foundations of those skills is important. AI art is not totally replacing artists in the industry. It can speed up workflows, especially in the ideation phase of a project, but artists are still needed to create polished game ready assets.

I went to several AI art related talks when I was at Unreal Fest in June 2025. All of them concluded that AI for art, especially 3D, still has a lot of issues and it can be harder to art direct AI art than art made by an actual human. Most said AI created 3D models can be fine for background assets, but artists are still needed to iterate on and refine important assets.

AI can definitely be used to speed up and assist the learning process, and we are starting to explore that in some courses. For instance, next semester in our Tech Art course (python scripting in Maya) the plan is to teach the fundamentals in the first part of the semester and then allow students to use AI to assist them in their final project. They will be able to use AI more effectively after first learning the fundamentals of python.

So the decision on when and how to allow students to use AI is context dependent and shouldn't be a substitute for foundational learning, it should be used to enhance learning. A student who uses AI to write code snippets for use in a game project should be able to clearly articulate exactly what that code snippet is doing. They should not just tell AI what they want the code to do and not analyze the results that AI gives them.

Using off the shelf art and animation assets is fine as long as art creation isn't a learning outcome for the project.

I think this can be a both and situation. For some projects, like the ones for Skills USA, students should use off the shelf assets if that is what most schools across the country are doing. But there should be other courses and projects in which the fundamentals are taught so students gain a solid foundation in whatever area they are pursuing.

We are not currently allowing students to use AI created art in projects within the Game Art program. In advanced courses we do allow students to use assets from Fab/Quixel for some assets in their scenes (large game environments) as long as the primary assets are created by the student and as long as they cite the use of those assets. I don't know of any game studio who would currently find it acceptable for an entry level applicant to have a portfolio full of AI generated art.

And to be clear - when someone is applying to the Game Art program at Champlain College, we do not want them to include AI art. We only want to see art they have actually created.

In most of the game art courses students have to post their work to weekly milestones in Canvas and we look at their work in progress each week. And they usually have to turn in the project files as well as the final render, so we haven't had a problem with students trying to turn in AI work as their own. And honestly, most of the game art students are pretty anti AI art because they actually like making the art themselves.

As far as the admissions portfolio goes, we have the three required drawings that have to be created using traditional media, and with those I think it would be relatively easy to tell if they are AI. Other work applicants submit may be more difficult to tell, but we haven't had a problem with that so far. I suppose we'll see how that changes as time goes on.

A lot of our students are concerned with the ethics of AI art, since it is basically incorporating whatever it finds online, and that indeed is one of the biggest problems. Most big studios that are incorporating AI into their workflows are using internal systems that are built off a library of the work from that studio in order to avoid possible IP infringement and to keep an external LLM from training off their content.

Anthony Daniels - Every single year... EVERY. SINGLE. YEAR... that I ask the members of my advisory board if we should be "an inch deep and a mile wide" or vice versa, the response that I always get is that colleges and employers want students that have a broad range of experiences in many different pieces of software/content areas, as opposed to a narrow specialization in just a few things. The line I always hear is "get them the general knowledge in everything, and we'll show them the rest." The other thing that I always hear in a similar vein is "teach them how to learn, so that they can learn by themselves." If your school does/encourages PBL-style projects, there's no reason why a student can't take a deeper dive into the subject areas they have a special interest in.

As far as assets/AI coding/etc, we're also finding our way as new subjects of our new robot overlords. What I've been telling my students as far as AI goes is, "make the code yourself, but if there's a particular area that you're stuck on, have AI take a look and give you suggestions." So far it seems to be a good balance between the students learning code/how it works, and not having to bang their heads against the wall when they hit a problem. Similarly, when we do HTML/CSS pages, we do about a week of hand coding to begin, but then start to copy and paste code from other sources. For game assets, we again do something similar; each student must create at least ONE asset themselves to put into their games. This can be something simple like a crate or box, but they have to have created the object, the image file, and the UVs map themselves. After they understand how this works, they can go and use other free assets if they like (I don't have them make 1st person/3rd person controllers as we're just too pressed for time), but they need to have demonstrated the skill of creating their own beforehand. I think it's important for them to know how to make something to use in a game engine since asset creators are still a thing for game studios (and you can still make a few bucks selling your designs online too).

I think it's great that you're doing more social events! Before COVID, I was very much a "learning comes first, then socialize." Now I feel like we need to make days for the students to get to know each other outside of their class responsibilities. I see similar things with my daughter and her friends; the skills of talking to someone on the phone, asking an adult you don't know for something, etc. is being lost. Great that you're addressing it!

Recruiting hasn't started for us yet (thankfully). I don't mind recruiting but balancing the students' work and our guests is always a challenge, especially for students that already find the work challenging. It's a mixed bag for sure.

Other than that, let me know if you need my .02 and thanks for your input on my board as well. Talk to you soon.

Curtis Aube - The tech industry is changing at a rapid rate. Coding assistance and review has taken off and can change the way we work. It's a bit like having a pair programmer with you all that time that can write code very quickly. I still find that I need to check and correct it. It makes a lot of mistakes (usually because it doesn't fully understand the context), but can still accelerate the process quite a bit.

I think you have the right idea in focusing more on breadth and less on depth of knowledge. Things like the walk cycle may be important in certain circumstances, but it will be rare and they can still learn it later if they need to. Focusing on the bigger picture will let students experience more aspects of a project and get to a nice prototype quicker. I'd expect many fundamentals are still important. Probably in that last 10% of the project that takes 90% of the work. That part where you need to start polishing the project and diving into the details. Allowing students to get there faster might start to reveal a pattern of which fundamentals are still important or what the new fundamentals are.

Something that I think has been true for a long time and is still true today in any company is that development time is best spent on the core of the business (The part that is unique and provides value). For a business, this means using third party libraries and connecting to external APIs for anything that's not directly impacting the core (Things that are generic and already solved). This allows you to move faster and get to the stuff that really matters for your business. I think the same principle applies to the students' projects. If they use premade assets and AI tools, then they can focus on the core aspects that make their project unique. They still need to know how to develop those core aspects of the game and that's where I think some of the fundamentals come back in.

With tech jobs, I think it's really important to continue to develop your soft skills. Technology will continue to change and some of the things you learned in the past will eventually become worthless, but the soft skills you've developed will transfer to any job and can really help you to stand out and be successful.

Chris Tall - I'm curious as to how you've observed the use of AI affect your student's learning? I remember back when I was in CAWD, we hand-coded all our HTML/CSS in Notepad++, as was the norm. Times sure have changed.

You may be surprised to hear this, but I have personally not yet used LLMs even once. That said, I've had plenty of exposure to it in the form of AI code reviews, and reading code written by Claude that was prompted by others.

To be honest, in the limited exposure I've had with this technology, I've found it to be disappointing. The vast majority of comments it generates for code reviews are either pointless, or redundant (example: suggesting a null check for a class member variable that has already been instantiated in its declaration). When it comes to code generation, it does a bad job of taking the context of the larger code base into consideration when developing a solution. It tends to "think" in isolated bubbles - which just doesn't work for large scale software development.

At this point I'd probably put myself staunchly in the "AI pessimist" crowd. I'm very much a believer in cultivating the ability to think through problems, and developing solutions by oneself (even if it takes time).

To answer your question on how I'm handling this technology, I'd simply say I'm "cautiously avoiding it" for now. I fully expect at some point I'll need to (be mandated to) use it. But until then, I continue to code the old-fashioned way.