How Do I Get a Position in the Games Industry? The FAQ

Michael Zyda, University of Southern California

The question most often asked of me at all levels is, "How do I get a position in the games industry?" I will attempt to answer this in the kindest way possible using my sage years of experience.

After 17 years at the University of Southern California (USC) of founding, running, and building the games program, the question most often asked of me is, "How do I get a position in the games industry?" This is being asked at all levels—by seniors in high school, soon-to-be-college-graduates who have not taken any game-building classes, and many others who started down other career paths that ended before they did. I attempt to answer it in the kindest way possible using my sage years of experience. There are many subquestions that are part of this, and I list and answer them as smartly as I know how.

Digital Object Identifier 10.1109/MC.2022.3151459
Date of current version: 6 May 2022

First of all, you really have to want to get a position in the games industry and be compelled to do this—if that is you, then great! Read on! If you are just thinking about it but studying/working in something else, it's probably not for you. In modern times, getting a position in the games industry means some years of study at the right university and experience with building games in teams. It used to be that all you had to do was take beginning programming and data structures courses and then drop out of school to found your own studio. That is rarely true now and not a recommended pathway to the games industry, says the professor who teaches games courses in a university.... So, let's start out by assuming you really want to do this, and you are ready for some education!

WHAT GAMES PROGRAM SHOULD I ATTEND?

This question gets asked of me all the time, even though I have been at USC for 17 years and founded the games program there. I usually do not answer this with, "USC, of

EDITOR MICHAEL ZYDA
University of Southern California;
zyda@usc.edu

course"; I usually try to provide guidance as to what the potential student should look for in a games program.

The way I usually start out is to let the questioner know where hiring is happening in the games industry. First of all, one of the constants in the games industry is that the demand for engineers/programmers is always strong:

- 60% of the demand in hiring for positions in the games industry is for engineers who know how to build games and work in cross-disciplinary teams
- 30% of the demand is for artists who know how to utilize the tools used by the games industry and produce art for the cross-disciplinary team
- 5% of the demand is for creatives in music and sound production for games
- 5% of the demand is for gameplay designers—and, for the most part, that demand is for seasoned designers, not fresh university graduates.

Cross-disciplinary teams means programmers, artists, gameplay designers, musicians, and sound engineers. Notice that I am not including business, legal, and marketing people, who are also essential. I am just going to focus on the actual game developers, not the corporate-side leeches. No. I really appreciate everyone!

SOME GAME SCHOOL PROGRAM QUALIFICATIONS

Engineers/programmers

When I say engineers/programmers, the school you are looking for ought to be teaching you C++ as a first language, followed by a data structures in C++ course, followed by a team-based project of some sort where your C++ skills are utilized. If the school you are considering is still teaching these courses in Java, you ought to run away. If the hiring people in the games industry hear that you learned Java as a programming language before you learned C++, they will terminate the interview and not consider you further. That's just the way it is. Also, C# is not C++, and, while skills in C# are great for building side-scrollers for the mobile market, C# will not help you get a position in a AAA-title-building game development studio. I pointer you to C++! programmer humor.

That program also needs to include a solid course in networking, preferably at the applications layer, and a solid course on operating systems. All AAA titles are networked and need responsive software built with an understanding of operating systems. These two courses are essential, and any program on game development that does not have this solid base for the engineers ought to be avoided, as the games industry hiring people will expect that you have had these. They will test you—you cannot just skip these courses and get found out later in the interview.

Another course engineers typically take is game engine development—this is not a course on how to use a game engine but how to build one. This course has been stuck inside of most games programs since their start circa 2005—when the expectation was that students would build their own game engine because there were not any good open source ones then. Now, a real game engine is a big piece of software. For example, Unreal Engine has more than 16 million lines of code, and your custom code on top of that engine will be 200,000–1,000,000 additional lines by the time your game ships.[1] Epic has probably spent more than US$1 billion developing this engine.

Therefore, the expectation that you will build a fully functioning game engine inside of your university course is a bit off. The closest I have ever seen a student team get to developing its own game engine was during our year-long advanced game projects course, where one half of the development team built its game, *Tales From the Minus Lab* (Figure 1), on top of the Unity game engine, while the other half developed a game engine that supported the functions Unity provided to the game. This could have been a train wreck, but the team was awesome and delivered both a game and a game engine. Most game engine development courses just task you with building the graphics rendering pipeline part of the game engine in C++. That is probably an excellent experience, and I highly suggest that you find a program that requires it.

Once you are so equipped with appropriate programming/engineering skills, you should then take a course on game design, which ought to use your programming skills to actually build something, not just hand you scissors, cardboard, and markers for you to paper-prototype something over a whole semester. Don't waste your time—build it!

The next thing you ought to do in this program is spend a semester working in a team, designing and prototyping a game that you can pitch for development over a two-semester period—we call this the advanced game projects (AGP) course. This ought to be a simulation of an industrial team, meaning you build a game over an academic year that is so unique that, when you show it to hiring people, they will immediately want to know your role in its development and, maybe, even consider hiring your entire team. That is the goal—it might be a reach, but that is what you should be shooting for.

Now, today, it is very easy to download game starter kits from Unity or Epic, but, if all you are going to do is build yet another side-scroller, no one is going to care. You need to build something that shows off your technological prowess and game design skills. If your game doesn't have networking, 3D characters, cool shaders, and something that grabs a players' need for gameplay, then you will not get far with the hiring people for the AAA-title games industry.

Artists

The most important lesson is that there is not just one kind of artist in the games industry—remember that 30% of the hiring demand is for artists. Roughly, there are concept artists, 3D modelers (character modelers and environment modelers), texture artists, lighting artists, and animators. Therefore, if you are selecting a games program, you ought to choose one that has a substantial offering in game art and design that joins with the engineers and gameplay designers in a yearlong AGP course.

Concept artists are required at the start of most game development to sketch out or model what the game will maybe look like. This is nice to have at the beginning, as it has a large impact on the selection of the technology to be used in building the game. Concept artists are specialized—they have skills that sell the project before production, when the money people have not yet said yes. They are also great for putting together a storyboard for the illustration of a complex part of the game during development.

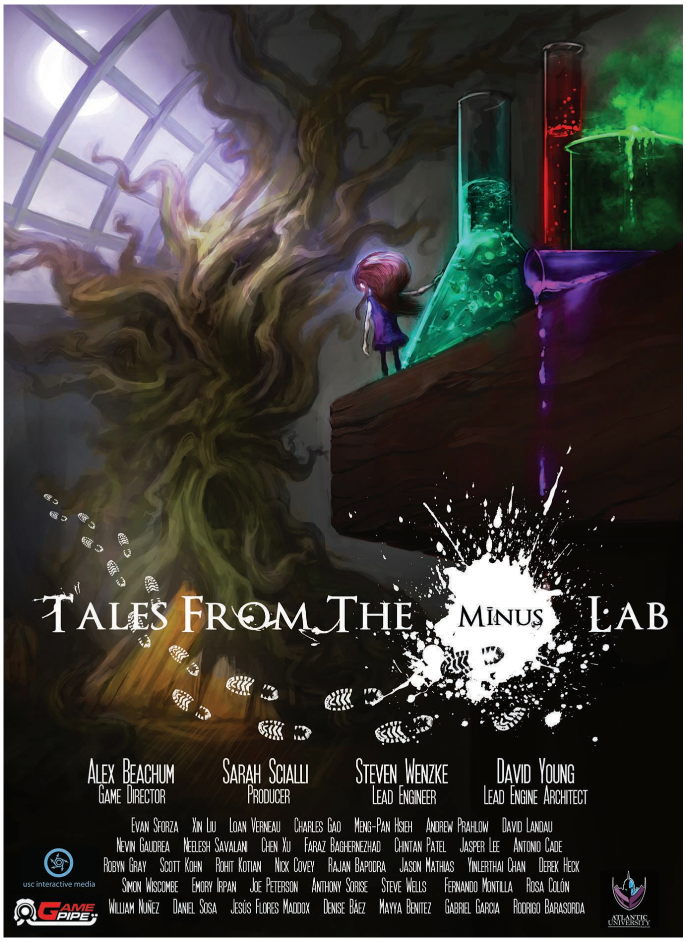
TALES FROM THE MINUS LAB
ALEX BEACHUM GAME DIRECTOR
SARAH SCIALLI PRODUCER
STEVEN WENZKE LEAD ENGINEER
DAVID YOUNG LEAD ENGINE ARCHITECT
EVAN SFORZA XIN LIU LOAN VERNEAU CHARLES GAO MENG-PAN HSIEH ANDREW PRAHLOW DAVID LANDAU
NEVIN GAUDREA NEELESH SAVALANI CHEN XU FARAZ BAGHERNEZHAD CHINTAN PATEL JASPER LEE ANTONIO CADE
ROBYN GRAY SCOTT KOHN ROHIT KOTIAN NICK COVEY RAJAN BAPODRA JASON MATHIAS YINLERTHAI CHAN DEREK HECK
SIMON WISCOMBE EMORY IRPAN JOE PETERSON ANTHONY SORISE STEVE WELLS FERNANDO MONTILLA ROSA COLÓN
WILLIAM NUÑEZ DANIEL SOSA JESÚS FLORES MADDOX DENISE BÁEZ MAYYA BENITEZ GABRIEL GARCIA RODRIGO BARASORDA
usc interactive media
GAME PIPE
ATLANTIC UNIVERSITY COLLEGE

FIGURE 1. The *Tales From the Minus Lab* poster from the USC Advanced Games Project course.

There are two kinds of 3D modelers: character and environment modelers. Character modelers build the 3D person/animal/robot you want for your game. Animators rig those 3D characters so they can move. Sometimes, your character modeler can rig, but model rigging is highly specialized, and it may be another person with different talents.

Environment modelers build the 3D models of the gamescape where the game takes place. They build buildings/mountains/forests where our 3D characters can frolic/run/shoot—or whatever they do in such scapes. The tool used by environment modelers is the same one character modelers and animators use, Autodesk's Maya.

Texture artists take the 3D models and apply a texture to the polygons that make up the 3D models. Texture artists tend to act strangely if you give them a digital camera to use—they will take pictures of paint peeling, dirt, sand, mud on your car, and just about anything that they might be tasked to apply onto the 3D models. I appreciate them greatly, as they make the 3D models/worlds look dirty and used, like the real world. Texture artists often work with lighting artists to add shaders to the polygons of the model so those textures are applied. Lighting artists light the scene, the entire 3D world and all its parts, with the hope that that lighting runs in real time, with the 3D modelers getting the blame if they use too many polygons for that world.

It is a rare games school that has such a comprehensive game art and design program—usually, game schools have to reach out to art schools and beg them to please do art for the games being built in the AGP course. Such relationships are valuable and delicate, and they should be treated with the utmost care.

Check if the games school you are considering has 1) a game art and design program or 2) strong relationships with outside art schools. If neither of these is true, then consider looking elsewhere. The alternative is that all your games use "Asset Store downloaded" models, which look nice individually but never look well together, as they rarely match the concept art you have, in either digital form or your head. Access to real artists makes the games school great—be there.

Gameplay designers

Gameplay design is how you create the story and interactions the player of the game performs. There is the overall story, sometimes called the *backstory*, and then the story for the player, meaning the player's role in that story and how he or she executes that role through some kind of interaction—button presses, keystrokes, and so on.

Game schools that teach you how to do gameplay design are going to teach you the history and vocabulary of gameplay, the various genres of games that have been created, and how you produce a game design document (GDD) that becomes the guide for your development team, a guide that needs to be continuously updated as game production proceeds. GDDs have sections for the introduction, audience, platform, system requirements, key features, reference games, gameplay overview, gameplay elements, gameplay verbs, gameplay philosophy, pillars of gameplay, what-this-game-is-not disclaimer, gameplay overview flowchart, win/lose conditions, replay value considerations, user interface (UI), gameplay scope chart, player character discussion, game world structure, levels, story overview, full narrative, and one-page production plan.

The introduction is a brief paragraph on what the game is about—the succinct game name and what do you do in that game. The audience is whom the game for and their ages—you want to build your game for a particular audience, and you need to know that before you start building it or drafting the rest of the GDD. The platform is what kind of device the game being built for, and system requirements are any special requirements for that platform—if you can't answer this in the GDD, stop.

The key features are three or four bullet points on what is interesting about the game you are proposing—if you cannot think of these, then why are you suggesting building this game? Do we really need yet another 2D side-scroller? I believe we do not and, in fact, suggest that Unity and Epic remove that starter kit from their websites! Please.

Reference games are the games that have influenced you to draft your GDD—they might be reference games in that "your game will play like them," "your art style will be similar to them," or "your story will be reminiscent of them." Hopefully, it will not be "the same game as xxx with different character skins," a 2D side-scroller, or a tower defense—I don't like tower defenses, either.

The gameplay overview has two parts—one is a single-sentence log line that tells you succinctly what the game is about. If you need two sentences, delete the second one. The other thing the gameplay overview section has is about three or four paragraphs that provide a high-level overview of your game's story and the character's role in it. I know that writing the short story is much harder than writing a novel, but you must resist making this overview overly overt.

The gameplay elements section is a short bullet point list of things like exploration, puzzle-solving, and swordplay. These are high-level things your player will be doing. Gameplay verbs are the things the player can do to experience the gameplay elements with the press of a button on the UI—walk, examine, pick up, use, and swing axe.

The gameplay philosophy is the overall philosophy of the development team with respect to the proposed game: "Our game seeks to innovate in the first-person shooter realm by drawing indelible chalk marks around each passed virtual character until the original sidewalk is no longer visible," or something like that. Usually, an additional five to six paragraphs are provided to explain this innovation to convince the green-light committee or investors that this is not just another side-scroller.

The pillars of gameplay section is a short bulleted list of how we are going to deliver on our promise of innovation in the game. "What this game is not" is a disclaimer to the reader (the development team, investors, and so on) to assure them that we are not creating yet another 2D side-scroller; we are building a significant 3D slice of the global metaverse that will surely make us all entertained and wealthy beyond our wildest dreams, along with a signature in blood.

The gameplay overview flowchart is a block diagram that shows how the various levels of the game are connected. Levels are complete, standalone parts of the game that are entered via success at another level or from the start of the game. Each level, typically, has a win/lose condition that lets you go on to either the next level/game end or the respawn location when you have lost the level. Each game must have an overall win/lose report when you get to the end of all levels or the end of all experience when it has been determined that you have completely lost.

The replay considerations section is the part of the GDD where you indicate either that you can only play the game once through, as things are exposed to you by the end of the game, or that you can play again and receive a new experience. The UI is where the buttons you press are and how they activate the gameplay verbs. In the UI are also informational displays that tell you your score or health so that you know if you are about to experience a miserable and humiliating virtual pause in your online entertainment.

The gameplay scope chart has way more detail than the gameplay overview chart. Its purpose is to provide an understanding of the complexity of what needs to be built. The player character discussion tells the development team the point of view of the character—first or third person—and how the player relates to the character in the overall narrative of the game.

The game world structure is a detailed narrative on what the hopefully 3D world of the game entails and how the characters move through it. Each level is broken out as to what it looks like and how it integrates into the overall game world.

The story provides the context for the game and a detailed understanding of how the player character moves through and operates inside of that story. There is an overview as well as a full narrative, with as many details as are useful for the development of the game.

There is a one-page production plan that indicates how many expected days each major task should take with respect to each specialized person involved in the game's development. The game producer who puts this together and does a good job has a position for life. If the game ends up taking eight years to produce after all of the development money has run out, then that producer might consider a different career in life.

That is what a GDD contains, and your mileage may vary. Gameplay designers are taught how to draft these through multiple projects in their educational program, sometimes testing their ideas with paper prototypes and, more frequently nowadays, via quick prototypes using starter kits downloaded from the Unity or Epic websites. The game school that you choose should provide you with some notion of how they provide you this education and as to how many of these gameplay designs have gone on to be deployed inside of hit commercial games.

Gameplay designers who do well in their educational programs rarely move directly into an industry role as lead designer. They typically enter industry as a level designer; technical designer; or, sometimes, assistant producer. Level and technical designer roles require skills beyond just gameplay design, and it is recommended that that additional education be pursued.

THE AGP COURSE

Now, at all of the schools you are looking at, one thing you want to make sure is that there actually is a 12- to 14-month long projects course, maybe with the title "AGP." The structure of this course is that it is a two-semester-long development project—say, the fall and spring semesters—with the game designs selected for production in the previous academic year's spring semester. At USC, we typically receive about 20 GDD submissions, and a faculty and industry committee gets this down to six or seven finalists for production. Winning teams in the AGP course receive lab space; a team of student developers—maybe 20–65 students; and oversight from faculty with experiences in gameplay design, engineering, art, music, and sound. This course has run from 2005 to the present and has placed some 3,000–4,000 students into positions in the games industry. Look for a school like that and have that kind of experience, and it should be rather straightforward for you to get a position in the games industry.[2]

DEMO DAY, EXPO, WHATEVER THE NAME—SHOWING OFF YOUR GAMES TO INDUSTRY

Whatever games school program you are considering ought to have something called *Demo Day* or *Games Expo*, a biannual event where student-built games are shown off to the games industry. I say *biannual* meaning that there is a demo day at the end of the fall semester and one at the end of the spring semester. The fall demo day is the best one for hiring and internships—most such decisions are made during the time period from December through March. Therefore, showing off great projects at a demo day in December is essential. The spring demo day is also nice, as it is a great way to celebrate the game school's experience for its graduates—it is important to note that, by the end of the spring semester, all of the internship and hiring positions are already filled, and little in the way of hiring will occur at that end-of-spring-semester event.

Now that you know these demo day experiences exist, attend them for the game schools you are considering. They will get you excited about the game development career path and let you know about limitations in the game school's educational program. If most of the games are side-scrollers, maybe find another school that is pushing the boundaries of technology and game development. Figure 2 is provided as a reference for what you should be looking for.

ADDITIONAL EVALUATIONS FOR GAME SCHOOL SELECTION

Once you have studied the educational programs at the game schools you are interested in, you next need to consider the people who run them—you want to learn from those who have

COMMENTS?

If you have comments about this article, or topics or references I should have cited or you want to rant back to me on why what I say is nonsense, I want to hear. Every time we finish one of these columns, and it goes to print, what I'm going to do is get it up online and maybe point to it at my Facebook (mikezyda) and my LinkedIn (mikezyda) pages so that I can receive comments from you. Maybe we'll react to some of those comments in future columns or online to enlighten you in real time! This is the "Games" column. You have a wonderful day!

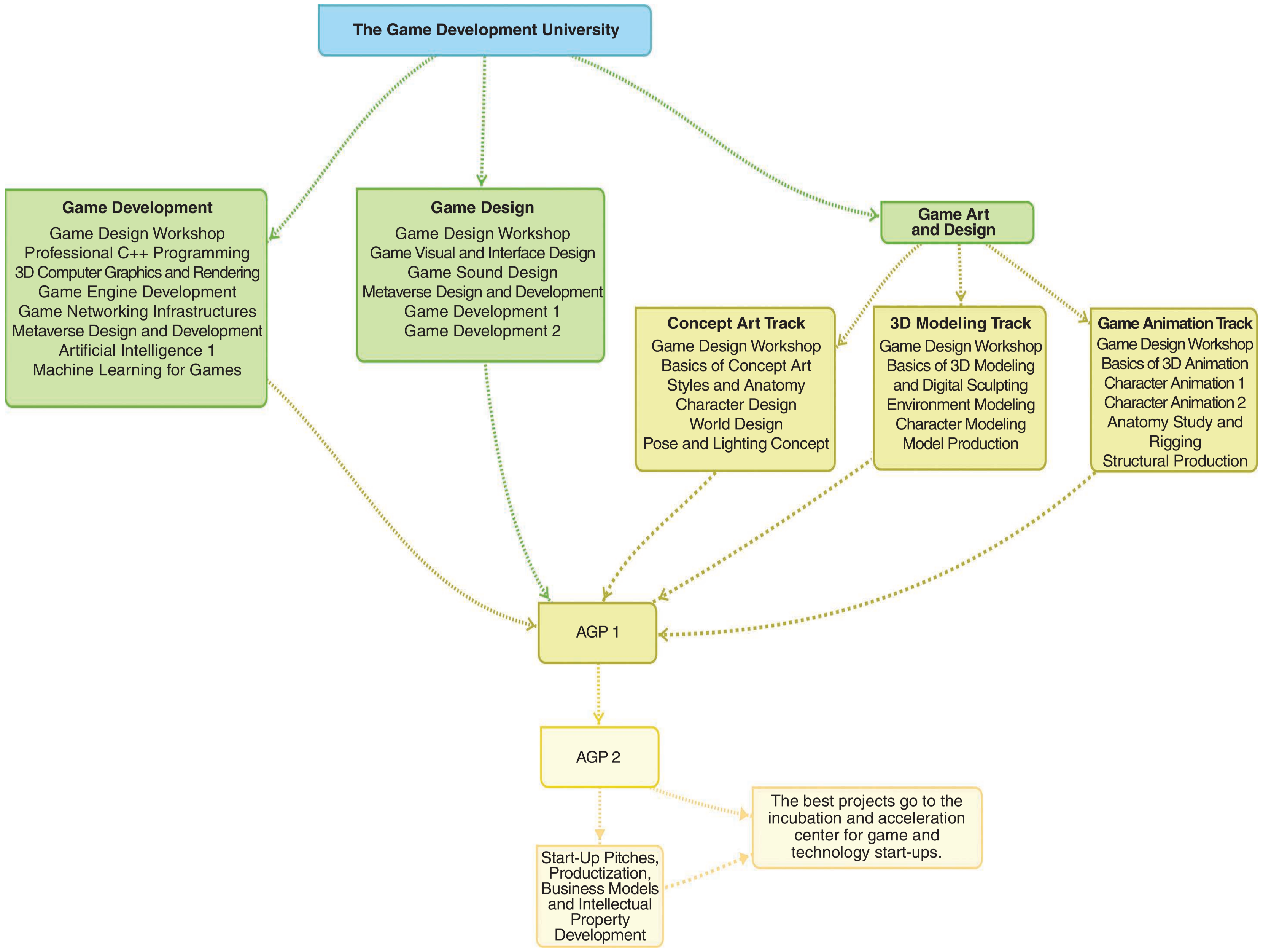
The Game Development University
Game Development
Game Design Workshop
Professional C++ Programming
3D Computer Graphics and Rendering
Game Engine Development
Game Networking Infrastructures
Metaverse Design and Development
Artificial Intelligence 1
Machine Learning for Games
Game Design
Game Design Workshop
Game Visual and Interface Design
Game Sound Design
Metaverse Design and Development
Game Development 1
Game Development 2
Game Art and Design
Concept Art Track
Game Design Workshop
Basics of Concept Art
Styles and Anatomy
Character Design
World Design
Pose and Lighting Concept
3D Modeling Track
Game Design Workshop
Basics of 3D Modeling and Digital Sculpting
Environment Modeling
Character Modeling
Model Production
Game Animation Track
Game Design Workshop
Basics of 3D Animation
Character Animation 1
Character Animation 2
Anatomy Study and Rigging
Structural Production
AGP 1
AGP 2
Start-Up Pitches, Productization, Business Models and Intellectual Property Development
The best projects go to the incubation and acceleration center for game and technology start-ups.

FIGURE 2. A prototype educational program for the Game Development University.

designed, developed, and shipped hit games. You don't want to learn game building from faculty who have not had commercial game-building experience, nor do you want to learn from someone whose primary experience in the games industry was to bankrupt the studio and put 800 developers out of work. The best way to discover something about the faculty is look them all up on LinkedIn and see who is best connected to the hiring people and leads in the games industry. The last time I looked at LinkedIn, I found that I was connected to 8,900 people in the games and computing industries and am followed by another 8,640. Reach into industry is important if you are expecting to actually get a position in the games industry upon graduation. You might also do a Google search on the key personnel to see how they have appeared in news and press releases.

That is how you get a position in the games industry: education, education, and lots of game building. Have fun in your pursuit of that dream! C

ACKNOWLEDGMENTS

When one establishes a games program, there are many people who contribute and help out. This list is not comprehensive. Gerard Medioni, chair of the USC Department of Computer Science, let me create the syllabi for 16 new courses and plan to establish the program. Chris Swain and Victor Lacour were instrumental in making the initial offerings of the advanced games projects course outstanding. Scott Easley, St John Colon, and Laird Malamed helped cast the AGP into something that will live forever at USC. All of my past students helped make the games program professional; great; and, most importantly, fun!

REFERENCES

1. "What is the size of Unreal Engine's source code currently?" Quora.com. https://www.quora.com/What-is-the-size-of-Unreal-Engines-source-code-currently (Accessed: Mar. 14, 2022).
2. M. Zyda, "Educating the next generation of game developers," *Computer*, vol. 39, no. 6, pp. 30–34, Jun. 2006, doi: 10.1109/MC.2006.197.

MICHAEL ZYDA is the founding director of the Computer Science Games Program and a professor of engineering practice in the Department of Computer Science, University of Southern California, Los Angeles, California, 90089, USA. Contact him at zyda@mikezyda.com.

Digital Object Identifier 10.1109/MC.2022.3166647