Teaching Building Game Engines: An entry-level course on game programming without pre-built Game Engines

Michael D. Shah
Yale University
New Haven, Connecticut, USA
michael.shah@yale.edu

ABSTRACT

Colleges and universities offering courses and degree programs to prepare students for careers as game programmers often have a fundamental tension when determining which tools and software packages to teach students. The fundamental tension involves: *which prebuilt game engine to teach with* or *should students build most of their projects from scratch*? The reason this is a tension, is that game development technology has historically evolved very fast (i.e. specific game engines, technologies, graphics, etc, could go out of favor), but at the same time entry-level job applications to game development studios may expect a portfolio of projects showcasing that a student would be ready to contribute to a game or multimedia software relatively fast in a fast-paced and deadline driven industry.

In this paper, we present our singular course on teaching fundamental game programming skills by *building game engines* from scratch. The course is designed to provide a pathway for students in game development as an entry level course (similar to the way algorithms is a foundational topic for computer science), while also providing students a pathway to learning industry specific technologies on their own time. In this paper we will provide our curriculum, student feedback, structure of the course, and data from teaching the course at multiple institutions.

ACM Reference Format:
Michael D. Shah. 2026. Teaching Building Game Engines: An entry-level course on game programming without pre-built Game Engines. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Educator's Forum (SIGGRAPH Educator's Forum '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3799829.3812510

1 INTRODUCTION

In 2022, video game industry revenue exceeded $183 billion dollars, and has since been projected to grow to $397.21 billion by the year 2029 [Intelligence 2024; Rao 2024]. From past trends and future projections, the gaming industry will continue to grow alongside other domains exercising similar skillsets such as visualization, augmented reality, and virtual reality. Colleges and Universities have thus prepared game related curriculum and computer science courses to answer student demand to enter this industry, as well as help prepare students for careers in this sector of the entertainment industry. The expansion of game design courses and programs continues to grow including at top ranked schools [News 2024] and as early as 2005 already 300 programs existed [Rajagopalan and Schwartz 2005].

Figure 1: Provided are a variety of game engines, games, and tools built by students. Some game engines also integrate hot reload, scripting languages, physics, and more components depending on the game.

Alongside the growth in industry has grown an industry of *game engines*. A *game engine* is a software package where many common components used in the development of a game (graphics systems, sound systems, asset loading, low-level memory management, input systems, etc.) are provided in a single tool and often accompanied with a graphical user environment (GUI). Games can then be built using a game engine by adding 'scripts' or otherwise modifying the original engine code to change the behavior. Thus a game engine can be described as *data-driven framework* for creating games, where the work of creating a game can be simplified to the game developer only needing to provide art assets and shorter programming scripts to create games [1]. According to the Game Developer State of the Gaming Industry 2024 survey of 3000 professional developers, 14% of professionals use Unreal Engine

[1]The complexity of the game and scripts of course can vary widely, and the authors note that game projects can take several years to complete even with the help of a game engine.

SIGGRAPH Educator's Forum '26, July 19–23, 2026, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2552-4/2026/07
https://doi.org/10.1145/3799829.3812510

(by Epic Games) and 33% use Unity3D (by Unity Software), and a similar survey shows metrics nearing 50% of all developers using engines [Byshonkov 2024; Joshi 2024]. Game engines have existed since the beginning of the industry, but it now appears a majority of surveyed developers use game engines. This means there may be considerable future pressure from the game industry to use the most popular game engines for recruiting and risk mitigation on their behalf. However, the risk to a student entering the game industry is that game engines and tools can change rapidly, and outside business, economic, and even political factors influence the "proper" engine to learn. Due to the emergence of a few dominant game engines, (but with the possibility of rapid change), we believe that it is more important to teach students about the foundational architecture of a game engine so that students can thrive long term in the game and related industries.

2 COURSE PHILOSOPHY

Provided in this section is our guiding principles in how we teach our entry-level course on game development. We have specifically titled this course *Building Game Engines*. This course has been taught over 7 years, at two institutions, and course materials has been shared among several other instructors at the university level. The following are our guiding philosophies:

(1) **Philosophy 1:** *Build games not engines.* Students should start by building simple, classic games from scratch using only a minimal library such as Simple Directmedia Layer to handle windowing and basic drawing of images[Sim 2025]. This allows students to see the common engineering among games, ultimately informing how to build a game engine.
(2) **Philosophy 2:** *Students should use their game engine to build a game.* Building a game engine is the next step after building several small games. It is important that students then actually use their game engine to then build a game.
(3) **Philosophy 3:** *Use a systems programming language.* Students need to understand how to build data structures, access threads, manage memory, and otherwise have exposure to systems concepts as applied to games. We have found this makes the transition to game engine technology much simpler.
(4) **Philosophy 4:** *Read source code of real game engines.* Every time we teach the course, students spend time reading code of open source and commercial engines. This helps them see the same abstractions and that a game engine is not a "magic" tool with a shiny graphical user interface.
(5) **Philosophy 5:** *Avoid 3D.* Adding in OpenGL, Vulkan, and Direct3D belongs in another course. Given only one semester, we have found sticking to 2D to be more than enough material to fill a semester. Keeping focused on 2D is another "teachable moment" to students that even general purpose game engines must eventually be specialized for a subset of game types or technology.
(6) **Philosophy 6:** *This is a hidden software engineering course.* We teach revision control (for source and art assets), team management, and other skills that will serve students even if they do not pursue work in games.

3 COURSE FORMAT

Provided in Table 1 is an example of the modules that we teach. We suggest Jason Gregory's text and Robert Nystrom's texts (free on the web as well) as supplements to the course [Gregory 2018; Nystrom 2014]. We occasionally use other curated web articles for mathematics such as immersive math that are freely available to make our course more accessible [Ström 2025]. The course otherwise is targeted at junior level through masters students who otherwise have taken their core courses including data structures and systems programming.

3.1 Course Programming Assignments

Provided is a listing and brief objective of individual game assignments in our course. The course has several individual assignments for students to complete, and then a final course project completed in teams to serve as a portfolio piece.

(1) Text Adventure - Expose students to turn-based game loop
(2) Battle Mage - Expose students to real-time game loop
(3) Catching Stars - A simple game for understanding collision detection.
(4) Space Invaders - A larger game for managing resources, components, and scenes over two assignments.
(5) Asteroids - The goal of this PSET is to build on space invaders, but add a camera that spans a larger world, and practice mathematics.
(6) Tile City - Students build a small 2D platform based game with tiles similar to the classic Mario game.
(7) The final project of students choice of teams between 1-4 students (See Figure 1 for examples). Students are required to build an engine, a tool that generates data, and a game with multiple scenes with their engine. Accompanying the project is a website with a trailer of the game, screenshots, and an engine diagram illustrating their architecture as shown in 2.

4 RESULTS

Provided below are some anecdotes from students taken at the end of the course. Generally speaking, students after our course have been able to land positions at large AAA game studios and independent game studios. We believe the course content, and final project as a portfolio piece have been helpful in helping students land jobs at studios using game engines and proprietary technologies.

4.1 Student Feedback

- *'I learned many concepts in the class before by myself through fragmented youtube videos. I wish I could have taken this class during my undergrad. This is the best game class I have ever taken." - Fall '25 (1st year MS graduate student)*
- *"Loved getting to be more creative in a CS class." - Fall '24 (CS+Math Student)*

5 RELATED WORK

Our 'build from scratch' and build 'data-driven' software architectures approach is not the first of this type of course. In the graphics field, a 'no-API' approach is taken by Geigel and Shah for teaching introductory graphics courses [Geigel 2025; Shah 2024]. Mark

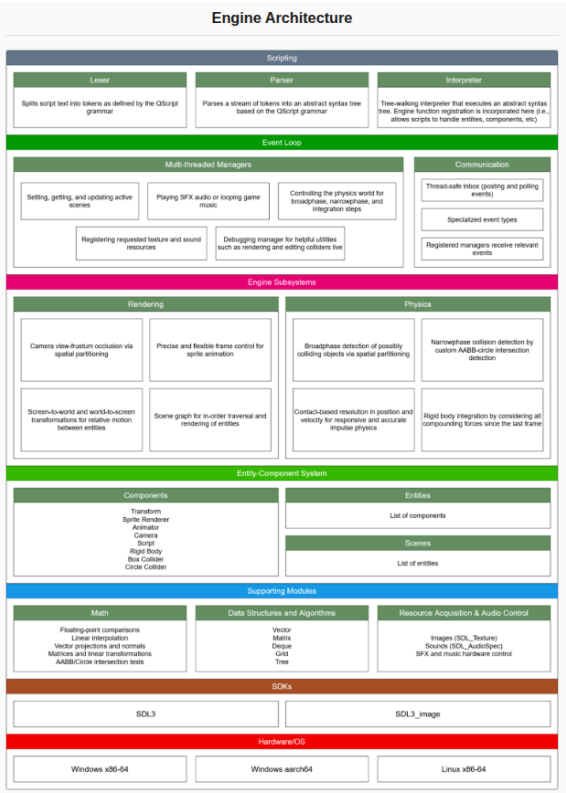

Figure 2: Provided is an example game engine architecture. Students document every component to reinforce how the components fit together, and iterate on their game engines.

Claypool's Dragonfly engine focuses on building text-based games through building a game engine [Claypool 2013]. The Alice project has also used storytelling to teach software engineering concepts through a data-driven engine for decades in the virtual reality space [Cooper et al. 2000; Pausch et al. 1995].

6 CONCLUSIONS AND FUTURE WORK

In this work we have provided our course focusing on building games and ultimately a data-driven game engine from scratch. We teach fundamental software and systems programming skills that students later apply to specific game engines as required for rapidly evolving industry jobs. This is not the first course to do this, but we believe we have provided a comprehensive and realistic view of what students can achieve in a semester that prepares them specifically for using *any game engine* in the future. We also believe our course philosophies will help guide others in building and promoting other similar and new courses in related domains.

Table 1: A sample syllabus of each lecture in a semester long course consisting of two lectures per week for 75 minutes.

Lecture	Title
1	Games and Game Engines
2	Game Genres, Game Applications, and Game Loops
3	Systems Programming Language (C++ or D) Review
4	Introducing SDL3
5	SDL Graphics and Framerate
6	Game Object and Component Pattern
7	Sprite Animation and State Machines
8	Engine Support Systems 1: Resource Managers
9	Engine Support Systems 2: Gameplay and Scripting
10	Game Audio
11	Game Math 1: Spaces
12	Game Math 2: Vectors
13	Game Math 3: Matrices
14	Game Math 4: Hierarhical Transforms and Scenes
15	Tilemaps
16	Serialization Techniques
17	Industry Guest Speaker
18	Physics 1
19	Physics 2
20	Object Pools and Memory Tagging
21	Graphical User Interfaces
22	Systems 1: Parallelism and Concurrency
23	Systems 1: Job Systems
24	Team Check in
25	Work day/Flex day up to instructor's discretion
26	Final Project Presentations

REFERENCES

2025. Simple DirectMedia Layer - Homepage. https://libsdl.org/ [Online; accessed 2025-02-17].

Dmitriy Byshonkov. 2024. GDC & Game Developer: The State of the Gaming Industry in 2024. https://gamedevreports.substack.com/p/gdc-and-game-developer-the-state. (Accessed on 08/06/2024).

Mark Claypool. 2013. Dragonfly: strengthening programming skills by building a game engine from Scratch. *Computer Science Education* 23, 2 (2013), 112–137.

Stephen Cooper, Wanda P Dann, and Randy Pausch. 2000. Alice: A 3-D tool for introductory programming concepts. *J. Comput. Sci. Coll.* 15, 5 (2000), 107–116.

Joe Geigel. 2025. A no-API Approach to an Introductory Computer Graphics Course. In *Eurographics 2025 - Education Papers*, Rafael Kuffner dos Anjos and Karina Rodriguez Echavarria (Eds.). The Eurographics Association. https://doi.org/10.2312/eged.20251009

Jason Gregory. 2018. *Game engine architecture.* AK Peters/CRC Press.

Mordor Intelligence. 2024. Gaming Market Size. https://www.mordorintelligence.com/industry-reports/global-gaming-market/market-size. (Accessed on 08/06/2024).

Sagar Joshi. 2024. 27 Game Engine Statistics About the Rise of Gaming. https://learn.g2.com/game-engine-statistics. (Accessed on 08/06/2024).

US News. 2024. 2024 Best Undergraduate Game Design Programs - US News Rankings. https://www.usnews.com/best-colleges/rankings/computer-science/game-simulation-development. (Accessed on 08/06/2024).

Robert Nystrom. 2014. *Game programming patterns.* Genever Benning.

Randy Pausch, Tommy Burnette, AC Capeheart, Matthew Conway, Dennis Cosgrove, Rob DeLine, Jim Durbin, Rich Gossweiler, Shuichi Koga, and Jeff White. 1995. Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications* 15, 3 (1995), 8–11.

Mohan Rajagopalan and David I Schwartz. 2005. Game design and game-development education. In *Phi Kappa Phi Forum*, Vol. 85. Honor Society of Phi Kappa Phi, 29–33.

Pallavi Rao. 2024. Charted: Video Game Industry Revenues By Year & Platform. https://www.visualcapitalist.com/video-game-industry-revenues-by-platform/. (Accessed on 08/06/2024).

Michael Shah. 2024. There Can Be (at least) Two Introductory Graphics Courses: Teaching Introduction to Non-Interactive Computer Graphics. In *ACM SIGGRAPH 2024 Educator's Forum* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3641235.3664429

Akenine-Möller Ström, Åström. 2025. Immersive Math. https://immersivemath.com/ila/index.html [Online; accessed 2025-02-17].