Gator Engine
CIS 4914 Senior Project
Project Documentation

University of Florida

Submission Date: 4/23/2024

Project Advisor	*Professor Joshua Fox*	joshuafox@ufl.edu
Project Manager **Scrum Master**	*Jonathan Jean*	jonathanjean@ufl.edu
Development Team	*Jonathan Jean*	jonathanjean@ufl.edu
	Lily Reyes	l.reyes@ufl.edu
	Alec Dishchuk	adishchuk@ufl.edu
	Timothy Swango	timothyswango@ufl.edu
	Alexandra Cornide Huber	acornidehuber@ufl.edu

Extra Credit Sonar documentation submitted to Prof. Cheryl Resch

Abstract

This proposal presents the *Gator Engine*, a specialized 2D game engine designed to offer middle school students the opportunity to design and develop their own 2D side scrolling games. The engine attempts to bridge the gap between overly simplistic and complex game development tools. It aims to offer a graduated learning experience that is both engaging and educational. The engine starts with an easy-to-use drag-and-drop interface and scales to incorporate Lua scripting for more advanced learners. The proposal stresses the importance of game design in education, noting its potential to improve problem-solving abilities and computational thinking. It leverages research that advances that argument that students can achieve a deeper understanding of STEM concepts through game design. The Gator Engine offers a scaffolded learning environment that grows with the student's abilities. The ultimate goal of the engine is to foster a new generation of game innovators. It represents a commitment to improving STEM education by making it more accessible, appealing, and effective through the medium of game design.

Keywords

2D Game Engine, Educational Technology, Middle School Education, Game Design Learning, Computational Thinking, Drag-and-Drop Interface, Lua Scripting, Scaffolded Learning Environment, STEM Education , Problem-Solving Skills, User Interface Design, Entity-Component-System (ECS), Project-Based Learning (PjBL), SFML (Simple and Fast Multimedia Library), Box2D (Physics Engine), Sol2(Lua Binding Library).

Table of Contents:

Introduction

Problem Statement: The Challenge of Game Design in Education

The integration of game design into middle school curricula faces two primary challenges: the cognitive overload associated with complex game development tools and the lack of progression from basic to advanced skills.

Why It's a Problem

Existing tools either oversimplify the process, preventing skill advancement, or are too complex, causing frustration among young learners.

Why It's Worth Solving

Addressing this issue can transform educational outcomes, making STEM subjects more appealing and accessible. Effective game design learning tools can foster a new generation of creative problem-solvers and innovators.

Background | Related Work

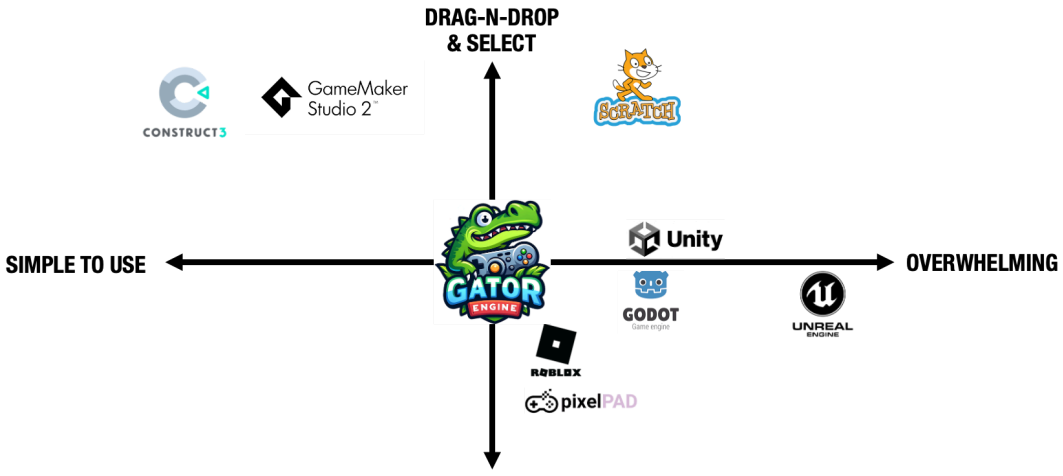
Several researchers and institutions have explored the benefits of game design in middle school education along with the challenges of integrating it into the curriculum. Bulut et al. (2022) demonstrate how a structured curriculum enhances creativity using the Torrance Test. Similarly, Hui and Mahmud (2023) show that game-based learning boosts cognitive and affective outcomes in mathematics. Additionally, Amador and Soule (2022) discuss Scratch's effectiveness in teaching coding through a user-friendly interface, easing cognitive load and fostering computational thinking.

Contribution

The Gator Engine builds upon existing technologies by providing a 2D game design platform. It integrates a drag-and-drop interface with the advanced capabilities of Lua scripting, thus enabling students to transition from basic game design to more complex game development. This approach simplifies the learning curve associated with traditional game development tools and also enriches the educational experience by gradually introducing programming concepts.

Problem Domain

A Tool Uniquely Suited for Middle Schoolers



The Gator Engine is a 2D game design platform targeted to middle schoolers.

To go beyond simple tools like *TinyTap*, the engine provides scripting capabilities to help students transition from basic game operations to the nuances of Lua scripting.

In the scope of game development, the Gator engine goes beyond *Scratch*. It adds specific game development features that are easy to implement without coding, such as game object manipulation and 2D physics.

While *Unity* and *Unreal* are renowned for their capabilities, their complexity can be daunting for middle school students. The Gator Engine retains the core elements of game creation but simplifies them for young learners.

Roblox engages kids with its 3D world, but starting with 3D can be tough for beginners. The Gator Engine offers a friendly 2D space, making it easier to grasp game design basics before tackling the complexity of 3D environments like Roblox.

The Gator Engine is a game design tool that makes game design possible for young learners. It breaks down barriers for them, offering an easy start to game creation. As they advance, it offers them tools to delve into coding and learn even more game design techniques. More importantly, it engages and motivates them to develop computational thinking skills all while having fun.

Literature Review

Enhancing Middle School Education Through Game Design

Introduction

As educational technology evolves, incorporating game design into the curricula offers a unique opportunity for student engagement and learning in middle school education. This literature review synthesizes research findings to support the role of teaching game design to students, advocating for a specialized 2D game engine that balances ease of use with the depth needed for genuine learning experiences in game design and programming.

Purpose Statement

The purpose of this literature review is to identify and synthesize the current trends and developments in the use of game design as an educational strategy for middle school students, and to justify the need for a tailored 2D game engine that supports both foundational learning and the development of advanced skills.

Synthesis of Research Findings

Game Design as a Learning Medium

Research indicates that middle school students can grasp advanced computer science concepts when taught through game design using tools like Alice (Werner, Campe, & Denner, 2012). Simplified game engines, akin to fantasy consoles, provide a structured yet adaptable framework that matches the cognitive capabilities of younger learners (Arnold, 2022).

Engagement through Project-Based Learning (PjBL)

PjBL approaches have been successful in promoting student involvement in programming (An, 2016). By integrating PjBL, a 2D game engine can encourage students to create engaging game projects, such as side-scrolling and top-down games, which can demystify complex programming concepts.

Computational Thinking via Game Design

Exploring game design provides a fun way and engaging way for middle school students to approach problem-solving. Research has shown that when students create games, they're actually practicing how to break down complex problems and find solutions (Akcaoglu & Koehler, 2014).

A case study involving Taiwanese middle schoolers revealed that students who design games can learn to see the big picture — how individual elements interact to create a functioning system (Wu, 2018). They also become better communicators and strategists, skills that transfer well beyond the gaming context.

Multimodal Learning through Digital Games

Digital games serve as multimodal learning platforms that can cater to diverse learning styles by utilizing visual, auditory, and interactive elements (Garneli, Giannakos, Chorianopoulos, & Jaccheri, 2015). A well-designed 2D game engine can harness these elements to create an immersive educational experience.

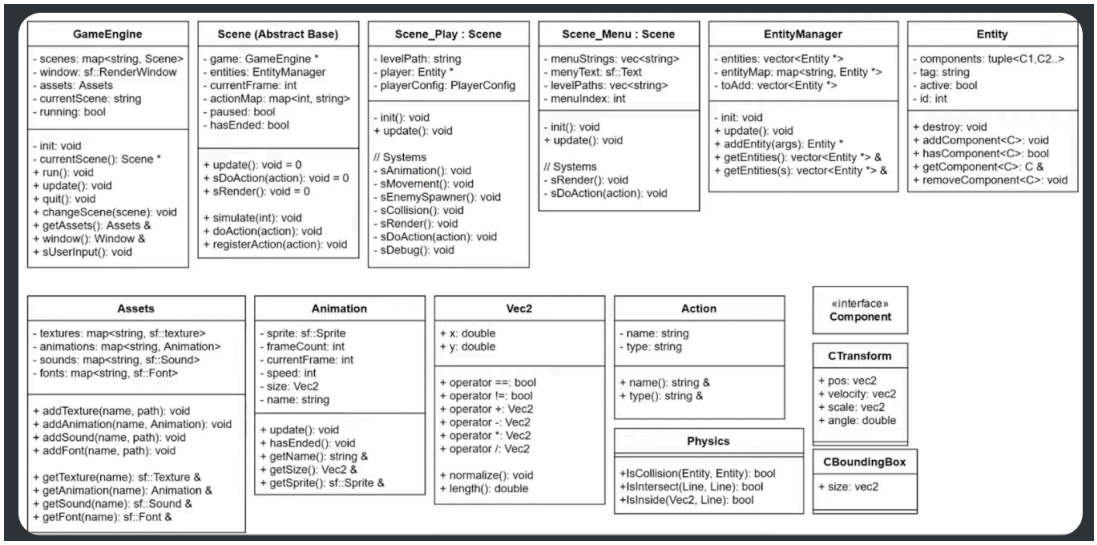
Conclusion

The reviewed literature underscores the importance of a 2D game engine designed specifically for middle school education and the benefits it can provide. This engine should offer a platform for students to learn and practice game design and programming within a supportive community, fostering both creativity and systematic thinking. The collected research indicates that a game design curriculum that uses a suitable game engine can be a powerful tool for learning, and aligns with the principles of reduced complexity and enhanced learning engagement.

Solution

Combining an ECS Architecture with Feature-Rich Library Choices

An ECS Architecture



Churchill, D. (2023, June 15). COMP4300 Game Programming [Video]. YouTube. https://www.youtube.com/watch?v=s99UDGdYIUE

This architecture was inspired by Dave Churchill's COMP4300 Game Programming Course. It emphasizes modularity and scalability. We chose ECS architecture as our foundation because it organizes game objects into entities, components, and systems. This framework enhances performance and flexibility by storing entities as unique IDs associated with a tuple of components, reducing the heavy inheritance hierarchies in object-oriented designs.

SFML Integration

SFML provides a suite of modular classes for handling windows, graphics, audio, and network. From a mathematical standpoint, SFML simplifies complex operations, abstracting the underlying computational geometry and linear algebra involved in rendering 2D graphics and managing audio processing.

Physics with Box2D

This adds a layer of physics simulation to animate the movement of game objects with real-world behavior and collision detection. The algorithms within Box2D apply principles of classical mechanics, using mathematical equations to simulate forces, velocities, and physical interactions between objects.

UI and Testing with Dear ImGui

ImGui powers our UI design and testing, streamlining the creation of in-game interfaces and debug tools for a responsive gameplay experience.

Lua Scripting with Sol2

Sol2 binds Lua to C++, providing a high-performance scripting environment within the game engine. This allows for complex behavior to be scripted mathematically and algorithmically in Lua, while executing within the engine's C++ framework.

Build Management with Cmake

CMake simplified our build process, providing a cross-platform system to manage the construction of the game engine smoothly. It enabled consistent builds across different environments, essential for collaborative development.

Testing Code Quality with Sonar Cloud

SonarCloud played a key role in our quality assurance strategy. It analyzed our codebase, detecting bugs and vulnerabilities early and ensuring adherence to coding standards for a clean, efficient code.

Version Control with GitHub

GitHub hosts or Gator Engine code and serves afor our version control. It tracks changes and coordinates work among our development team. It also automatically tests with Sonar Cloud.

Project Management with Github Project Management Tools

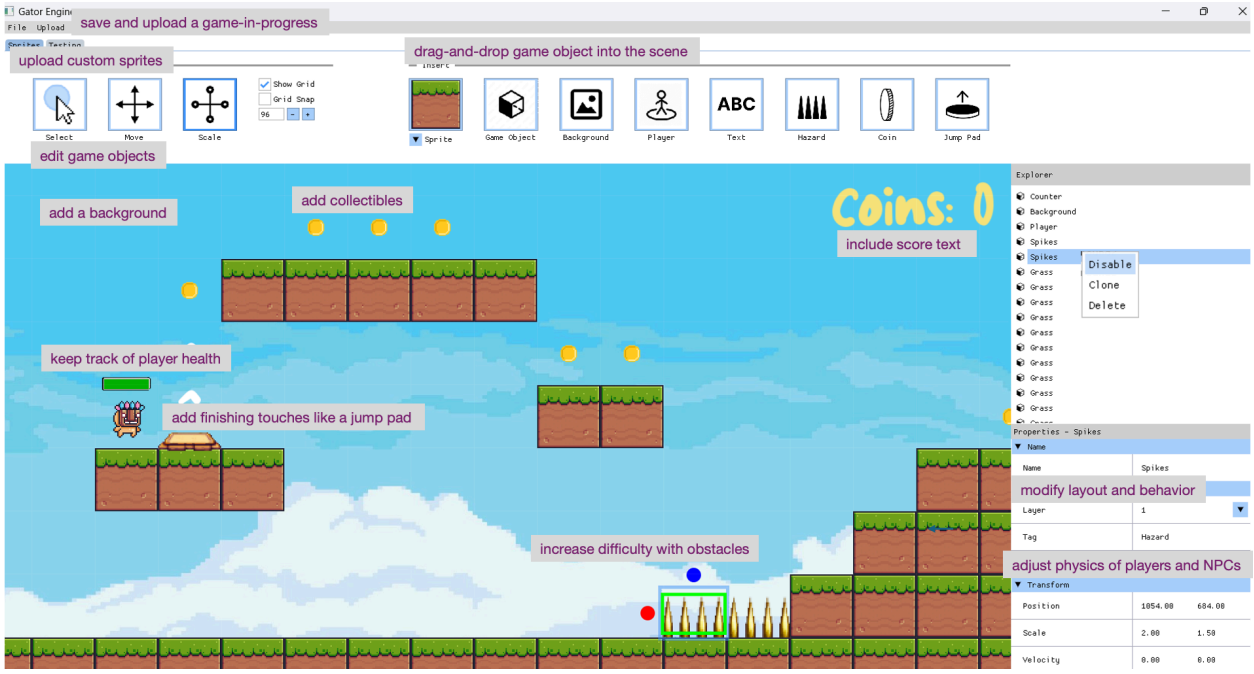
It was used for overall project planning, sprint planning, tasks organization, and progress tracking.

GitHub Wiki as a Learning Hub

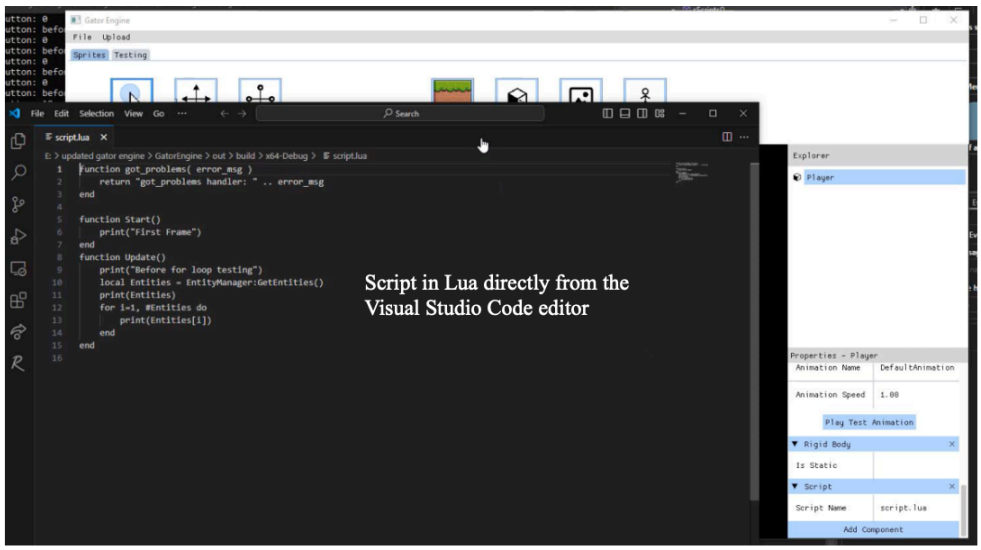
Our GitHub Wiki hosts the game engine's tutorial and reference section, offering an easily accessible and editable documentation space. It includes hyperlinks, images, and GIFs, making the learning process more intuitive and visually engaging.

Results

A friendly user interface with drag-and-drop, drop-down menus and input boxes.

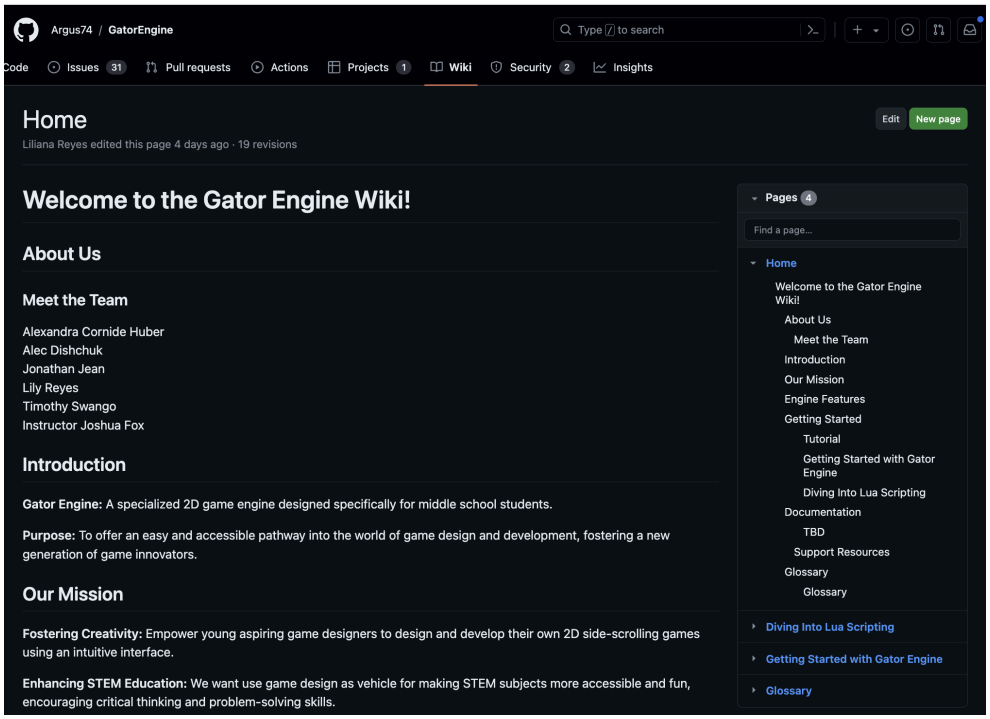


A scripting editor using the easy to learn and friendly Lua language.



← Launch the script editor from the UI

An online tutorial easily accessible in our Github Wiki that includes images, gifs and hyperlinks.



Conclusions

Throughout the development of the Gator Engine, we encountered and overcame technical challenges that contributed to our professional growth and the engine's robustness.

Challenges and Learnings

- Code Integration: We navigated the complexities of merging code from the different members of the team, addressing inconsistencies and redundancy to create a workable engine without code conflicts.
- Build Process Mastery: Adopting CMake was initially challenging due to its steep learning curve, but it ultimately streamlined our build process across multiple platforms.
- Quality Assurance: Diligent debugging with SonarCloud was imperative in maintaining high code quality, though it proved to be a meticulous and time-consuming task.

These challenges highlight the importance of rigorous testing, consistent coding standards, and the selection of appropriate development tools—lessons that are invaluable for future software engineering endeavors.

Future Directions

Given these experiences, future work may involve:

- Enhanced Documentation: Creating more comprehensive documentation to ease future code integration and onboarding processes.
- Build Automation Enhancement: Further automating the build process to reduce complexity and improve developer experience.
- Advanced Debugging Techniques: Implementing more sophisticated debugging tools or practices to streamline the identification and resolution of code quality issues.

These insights into the challenges faced provide context for our project's achievements and underscore the ongoing need for refinement and learning in the field of software development.

Advantages and Disadvantages

Our engine offers a balance between simplicity and advanced features, yet there's room for growth. While the engine can be used for educational purposes, I does not yet meet the requirements for commercial game development.

Standards and Constraints

The Gator Engine was developed adhering to established software development standards to ensure quality and interoperability.

Standards

Programming conformed to C++17, the ISO/IEC standard at the time of development. SFML, Box2D, ImGui, and Sol2 version 3 libraries were utilized, all compatible with this standard. CMake version 3.15 facilitated build management, while SonarCloud enforced code quality standards.

Constraints

The engine was designed to be compatible with Windows and macOS, ensuring a GUI refresh rate capable of supporting smooth gameplay. The choice of libraries and tools was constrained by the requirement to keep the development environment open-source and accessible to educational institutions.

Acknowledgements

We extend our thanks to Prof. David Churchill for his informative course, which he has generously shared online. His teachings have been instrumental in our development. We are also grateful to our advisor, Prof. Joshua Fox, who provided invaluable guidance, sage advice, and engaging stories throughout this project. Their contributions have profoundly impacted our work and learning experience.

References

Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers & Education, 75*, 72-81. https://doi.org/10.1016/j.compedu.2014.02.003

Amador, J., & Soule, H. (2022). Game-based learning: Coding with Scratch. In *Technology and the Curriculum: Summer 2022*. Pressbooks. https://pressbooks.pub/techcurr20221/chapter/game-based-learning-coding-with-scratch/

An, Y.-J. (2016). A case study of educational computer game design by middle school students. *Educational Technology Research and Development, 64*(4), 555–571. https://eric.ed.gov/?id=EJ1108010

Arnold, M. (2022). Tiny game engines. *Read Write Code*. [Blog post]. Retrieved from https://readwritecode.blog/tiny-game-engines-561e3396a599

Bulut, D., Samur, Y., & Cömert, Z. (2022). The effect of educational game design process on students' creativity. *Smart Learning Environments, 9*(8). https://doi.org/10.1186/s40561-022-00188-9

Common Sense Education. (n.d.). Best Tools to Make Games. Retrieved January 20, 2024, from https://www.commonsense.org/education/lists/best-tools-to-make-games

Garneli, V., Giannakos, M. N., Chorianopoulos, K., & Jaccheri, L. (2015). Serious game development as a creative learning experience: Lessons learnt. In *2015 IEEE/ACM 4th International Workshop on Games and Software Engineering* (pp. 36-42). IEEE. https://doi.org/10.1109/GAS.2015.14

Hii, H. B., & Mahmud, M. S. (2023). Influence of game-based learning in mathematics education on the students' cognitive and affective domain: A systematic review. *Frontiers in Psychology, 14*. https://doi.org/10.3389/fpsyg.2023.1105806

Raffaillac, T., & Huot, S. (2019). Polyphony: Programming Interfaces and Interactions with the Entity-Component-System Model. In *Proceedings of the ACM on Human-Computer Interaction,* 3(EICS), 1-22. https://doi.org/10.1145/3331150

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for Agile methodology. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*. https://doi.org/10.1109/CCAA.2017.8229928.

Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)* (pp. 427–432). ACM. https://doi.org/10.1145/2157136.2157263

Wu, M. L. (2018). Educational game design as a gateway for operationalizing computational thinking skills among middle school students. *International Education Studies, 11*(4), 15. https://doi.org/10.5539/ies.v11n4p15

Biography

Jonathan Jean

Hi, my name is Jonathan and my main interests are in computer graphics and computer graphics education. As a part-time tutor, I am constantly researching ways to more effectively help my young students understand graphics technologies and concepts. I am looking forward to learning different strategies for sharing graphics programming with future generations.

Lily Reyes

After earning a Master's degree in History from Princeton University in 2010, I relocated to New York City, where I embarked on a career in Digital Advertising. I worked as a Director of Strategy and Analytics at companies such as R/GA, Code and Theory, and IBM. In 2020, I returned to academia to pursue a degree in Computer Science, focusing on coding and game design. Outside of my professional and academic pursuits, I enjoy hiking, long-distance running, gardening, and caring for my pets. After I graduate from the University of Florida, I will pursue graduate studies in Computer Graphics at the University of Utah, Kahlert School of Computing.

Timothy Swango

In the summer of 2023, I worked as a software engineering intern at Walmart Global Tech. After graduating, I'm returning to the San Francisco Bay Area to start my career at Walmart Global Tech as a full-time software engineer. Apart from schoolwork, I enjoy creating video games or other ideas that I find brewing around that could be interesting to make. I also love traveling to new places, going to the gym, hiking, snowboarding, and golf.

Alexandra Cornide Huber

I'm a to-be software engineer with an interest in graphics programming. Previously, I interned at NASA working on the Launch Control System, then at Microsoft working on Babylon.js. I will be returning to Microsoft in the fall to work full-time. When not at school or work, I enjoy game development, traveling, brewing, and scuba diving.

Alec Dishchuk

This summer after graduating I will return to World Kinetic Energy as a full-time Cloud Software Engineer, where I've spent my previous two summers interning as a Cloud Infrastructure engineer and Application engineer. My career goal is to get into game development, and to create something that everyone would love to play. In my free time I enjoy playing video games, working out, and spending time with friends and family.