High-level to Low-level of Abstraction: Teaching Computer Graphics and GPU Programming with a Game Engine

Mitja Hmeljak
Computer Science, Indiana University
mitja@iu.edu

Abstract

Real-time rendering of Computer Graphics algorithms is achieved through *shaders*, i.e. programs that run on GPUs (Graphics Processing Unit), for the parallel processing of vertices, fragments→pixels, etc.

Modern graphics APIs (Application Programming Interface) provide efficient access to GPU-based processing, but require a complex setup that represent a high entry barrier to students.

In designing an introductory Computer Graphics course, the debate is often about choosing between a top-down approach relying on fixed graphics pipeline functionalities, or whether to start by teaching GPU shader programming first.

This poster presents a set of Computer Graphics assignments that include both CPU-based and GPU-based implementations in the Unity game development engine.

Selected algorithms are shown first in the form of C# scripts that interface with the Unity scene editor, at a higher level of abstraction. The same concepts are subsequently revisited with the introduction of parallel processing and GPU programming concepts, by leveraging Unity's built-in interface to lower-level GPU shaders.

Introduction

Computer Graphics instructors face a dilemma, when designing an introductory Computer Graphics course:

- whether to take a top-down approach, presenting higher level concepts by relying on fixed graphics pipeline functionalities;
- or to start by teaching GPU shader programming first, thus also allowing for a more direct experience of current graphics APIs.

The question is especially relevant in degree programs that don't provide multiple Computer Graphics courses [5].

The amount of preparatory work that is required for writing GPU shader programs may be lessened by providing a software framework specific to the introductory Computer Graphics course [3]. Some textbooks provide a software library to allow most of the coursework to be implemented in GPU shaders [1]. Game development engines such as Unity are used in game development courses, as well as in introductory programming courses[6].

More recently, entire computer graphics courses have been redesigned to implement their coursework using Unity [2] [4].

Methods

"Fixed pipeline": teaching Computer Graphics with a Top-Down Approach

In designing an introductory Computer Graphics course,
- taking a top-down approach,
- presenting higher level concepts first,
- by relying on fixed graphics pipeline functionalities:

"Shaders first" approach: start by teaching GPU *Shader* Programming

In designing an introductory Computer Graphics course,
- start by teaching GPU shader programming,
- thus also allowing for a more direct experience of current graphics APIs
- (may have to start with a simpler version of the programmable pipeline)

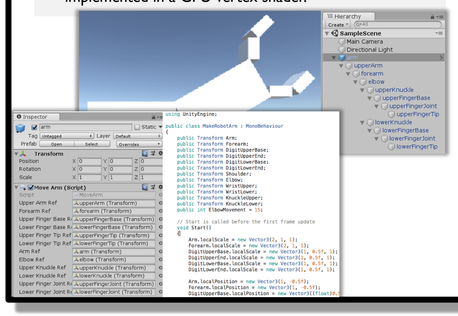
A different approach: use a Game Development Engine in a Computer Science course

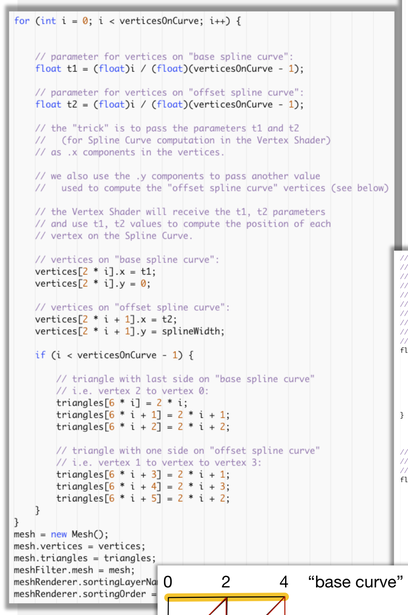
High-level to Low-level approach, with a Game Development Engine – assignments will include both:
- CPU-based implementations, and
- GPU-based implementations,

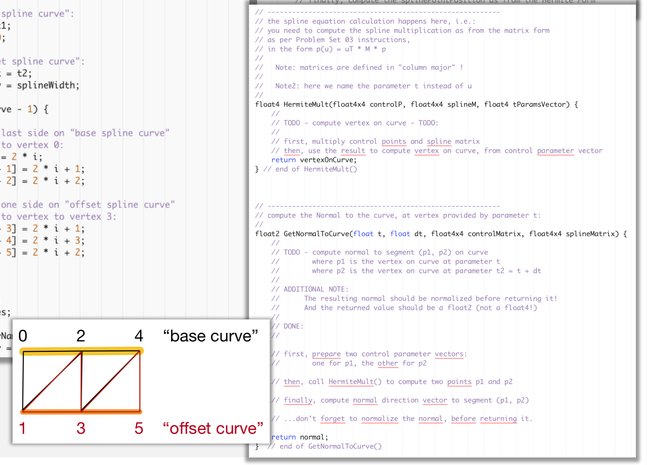
within Game Development Engine (e.g. Unity) runtime.

Each assignment introduces a specific concept or algorithm, for example:

Affine transformations and coordinate spaces
- first, transformations from object coordinates to world coordinates are implemented in the form of C# scripts that interface with the Unity scene editor.
- then, the same set of transformations is to be implemented in a GPU vertex shader.

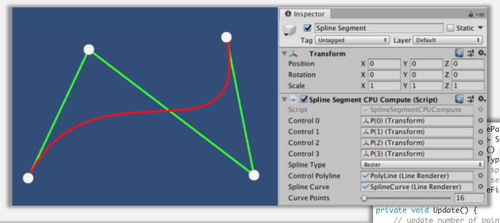






Assignments in Unity that include both CPU- and GPU-based implementations

Example assignment: ***Modeling curves and surfaces***
- Implementing spline equations in C# scripts, with input control points from Unity scene editor hierarchy.
- Curve modeling algorithms are then reimplemented in GPU shaders.



Implementation

- The complex setup that would be required by graphics APIs is provided by Unity's built-in interface to GPU shaders, while maintaining a higher level of abstraction in the Unity scene editor.
- By reimplementing details of the same algorithms in GPU shaders, students gain practice with parallel processing and lower-level GPU programming.

References

[1] Edward Angel and Dave Shreiner. 2012. *Interactive computer graphics : a top-down approach with shader-based OpenGL (6th ed.)*. Addison-Wesley, Boston, MA.

[2] Mitja Hmeljak and Holly Zhang, 2020. *Developing a Computer Graphics Course with a Game Development Engine.* In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (Trondheim, Norway) (ITiCSE '20). ACM, New York, NY, USA, 75–81. https://doi.org/10.1145/3341525.3387428

[3] James R. Miller. 2014. *Using a Software Framework to Enhance Online Teaching of Shader-based OpenGL.* In Proceedings of the 45th ACM Technical Symposium on Computer Science Education (Atlanta, Georgia, USA) (SIGCSE '14). ACM, New York, NY, USA, 603–608. https://doi.org/10.1145/2538862.2538892

[4] Gregory Smith and Kelvin Sung. 2019. *Teaching Computer Graphics Based on a Commercial Product.* In Eurographics 2019 - Education Papers, Marco Tarini and Eric Galin(Eds.). TheEurographicsAssociation. https://doi.org/10.2312/eged.20191031

[5] Jacqueline Whalley. 2020. *Critical Perspectives: The Shaders-First Debate.* ACM Inroads 11, 3 (Aug. 2020), 6–8. https://doi.org/10.1145/3410477

[6] Ursula Wolz, Gail Carmichael, and Chris Dunne. 2020. *Learning to Code in the Unity 3D Development Platform.* In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (Portland, OR, USA) (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 1387. https://doi. org/10.1145/3328778.3367010

Acknowledgements

- Holly Zhang(*) co-author in SIGCSE 2020 and ITiCSE 2020
- Rajin Shankar(*) implemented the first redesign of course assignments in Unity
- (*) both completed CSCI-B481 in Spring 2018 (when it was still based on OpenGL ES)
- Thanks to students taking Indiana University CSCI-B481 & B581, for their feedback and suggestions