WORKSHEET 4: GRAPHICS & VFX

Version 1.0
Computing
COMP270

Brian McDonald, Kate Bergel

Introduction

VFX in video games is a combination of art and programming practice, for which it is useful to have an understanding of the graphics pipeline and how to exploit its features to achieve interesting visual effects in an efficient way. In this worksheet, you will implement a VFX technique which will demonstrate your understanding of this process along with the calculations required to generate/manipulate 3D geometry and implement a shader.

Begin by **forking** the following BitBucket repository, which is empty except for a .gitignore file for Unity and Unreal projects in a variety of IDEs (you may need to amend this depending on your project setup):

https://gamesgit.falmouth.ac.uk/projects/COMP270/repos/comp270-worksheet-4

"Today, with computer-generated visual effects, everything is possible. So we've seen everything. If it can be imagined, it can be put on screen."

— Gabriel Campisi

Complete the task described below, remembering to **commit** your work regularly.

This worksheet has only one task, with multiple components; you may approach the work and structure your code in whichever way you think is most appropriate. You are recommended to use your implementations from the workshop exercises as a basis.

Task

For this worksheet, you should **implement** a basic VFX technique, which **must include** a procedural mesh element. You are advised to start by implementing a basic operation and iterating on it to increase the sophistication. You are free to choose any kind of effect, as long as it demonstrates your ability to manipulate data as it passes through the graphics pipeline in order to alter the appearance of an object, and you may implement your solution in either Unity or Unreal.

The following are some ideas for VFX techniques you could implement:

- Fireball
- Deflector Shield
- Liquid
- Flames
- Basic Animation (waving flags)

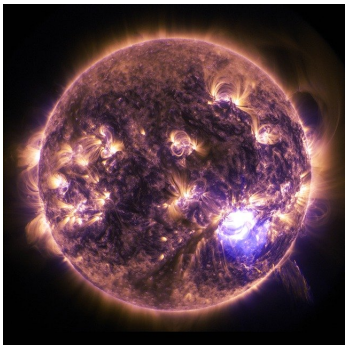

Image by skeeze from Pixabay

Marking Rubric

All submissions and assessment criteria for this assignment are individual. Allowances will be made for graph layout when this is not controllable.

Criterion	Weight	Near Pass	Adequate	Competent	Very Good	Excellent	Outstanding
PROCESS: VFX Technique	60%	There is no VFX Technique implemented, or the mesh was not generated in code.	The VFX technique is very basic (e.g. some simple animation) and the procedural mesh is a basic primitive.	The VFX technique shows efforts towards sophistication, but the procedural mesh is a basic primitive.	The VFX technique is moderately sophisticated, with some effort to link it into a game context (e.g. Fireball is cast by a player character). The procedural mesh is a simple extension of a basic primitive.	The VFX technique is sophisticated, well thought out, and is linked to a game action. The procedural mesh is a more complex extension of a basic primitive.	The VFX technique is extremely sophisticated, well thought out, and would fit well into a game. The procedural mesh is a complex and original shape.
PROCESS: Sophistication and Maintainability	40%	The graph is extremely messy or unintelligible, with obviously unnecessary nodes. The code is only sporadically commented, if at all, or comments are unclear. Few identifier names are clear or inappropriate. Code formatting hinders readability.	The graph is reasonably well laid out, but the purpose of some nodes is unclear and changing values results in no discernible difference to the final result. The code is reasonably well commented. Some identifier names are descriptive and appropriate. An attempt has been made to adhere to a consistent formatting style. There is little obvious duplication of code or of literal values.	The graph is well laid out, with the purpose of each node being reasonably clear, but changing some of the values results in undesirable variations to the effect. The code is well commented. Most identifier names are descriptive and appropriate. Most code adheres to a sensible formatting style. There is almost no obvious duplication of code or of literal values.	The graph is very well laid out and the purpose of each node is apparent. Some values may be adjusted to produce interesting variations on the effect. The code is reasonably well commented, with appropriate high-level documentation. Almost all identifier names are descriptive and appropriate. Almost all code adheres to a sensible formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered".	The graph is very well laid out and the purpose of each node is apparent, and good use is made of functionality to rename and/or annotate nodes where available. Many values may be adjusted to produce interesting variations on the effect. The code is very well commented, with comprehensive appropriate high-level documentation. All identifier names are descriptive and appropriate. All code adheres to a sensible formatting style. There is no obvious duplication of code or of literal values. Most literal values are, where appropriate, easily "tinkered".	The graph is extremely well laid out. Extensive and appropriate use is made of functionality to rename and/or annotate nodes where available. Most values may be adjusted to produce interesting variations on the effect. The code is extremely well commented, with comprehensive appropriate high-level documentation. All identifier names are descriptive and appropriate. All code adheres to a sensible formatting style. There is no duplication of code or literal values. Nearly all literal values are, where appropriate, easily "tinkered".