**CE301**

**Background reading for ‘Creating a 2.5d Survival Game with tailored Engine’**

Table of Contents

[Section 1 Game Engine reading 3](#_Section_1_Introduction)

[Section 2 Game Application reading 4](#_Section_2_Instructions)

[Section 3 Universal reading 5](#_Section_3_Game)

[Section 4 5](#_Section_4_Game)

[Section 5 5](#_Section_5_Class)

Section 6 5

# 1 Game Engine reading

## General

[1] Gregory, J., 2019. *Game Engine Architecture*. 3rd ed. Boca Raton, Florida: CRC Press.

Game Engine Architecture provides a sound background into the main elements and sub-systems typically provided by game engines and has been referenced to throughout the design of the game engine throughout. Descriptions on the implementations are more abstract giving a depth of knowledge in the methodologies, rather than direct code implementation. Although some code examples are offered these are minimalistic.

[6] Rojas, J., 2013. Getting Started with Videogame Development. *2013 26th Conference on Graphics, Patterns and Images Tutorials*,.

This journal further provided a high-level overview of some of the components involved with game-engines, from a general engine perspective. This provided knowledge into methods such as scene graph representation, spacial partitioning etc. Although no implementation code is given, this provides more abstract methodologies.

[7] IEEE, 2012 World Automation Congress - Design and implementation of three-dimensional game engine. Puerto Vallarta, Mexico, Mexico: IEEE, 2012.

This journal supplemented game engine design architecture from a 3d aspect, including background information on frustrum culling, billboard sprite drawing, and insight into into special effects such as particle systems. Although no implementation has been done relating to these, it provides the information to do so when the time comes.

## Event subsystem

[1] Provided an overview of the event subsystem, p114-134, in which a publisher-subscription style design pattern in used. Classes can either publish events to be distributed to subscriber classes or subscribe to receive events, where events are categorized through either enumeration or a type identifier to specify the category of events. The text given offers examples, although in this project a custom design has been used while still maintaining the general design pattern.

[8]P. Th. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” ACM Computing Surveys, vol. 35, no. 2, pp. 114–131, Jun. 2003.

‌Gave insight into a high level architecture of the publish/subscribe design pattern. Helped to realise how to categorise events and only send events through the publish/dispatch to subscribers who have selected interest in their categories.

## 1.3 Renderer

[4] de-Vries, J., 2020. *Learn Opengl, Extensive Tutorial Resource For Learning Modern Opengl*. [online] Learnopengl.com. Available at: <https://learnopengl.com/> [Accessed 12 October 2020].

LearnOpenGL.com provided a rich depth of knowledge in the implementations of rendering objects to screen in modern opengl. The website starts out with a basic tutorial exploring drawing your first triangle, to more advanced topics such as using shader storage buffer objects, a memory storage buffer area specific to shaders, or instanced based rendering, drawing the same set of geometry data multiple times in conjunction with arrays of instance specific data held as elements.

For any rendering-based query, or for information on how to draw anything to screen, this is the page most frequently visited. Any code utilized from the examples has been refactored and abstracted away only having been used to grasp the methods given.

[5] Kessenich, J., Sellers, G. and Shreiner, D., 2017. Opengl Programming Guide. 9th ed. Crawfordsville, Indiana: Pearson Education, Inc.

OpenGL programming guide, the ‘red book’, has been a supplementary textbook used to further advance knowledge in the rendering pipeline. While LearnOpenGL provides the backbone, this textbook gives insight into the newer features of OpenGL 4.5

[15]"OpenGL Programming/Modern OpenGL Tutorial Text Rendering 02 - Wikibooks, open books for an open world", *En.wikibooks.org*, 2020. [Online]. Available: https://en.wikibooks.org/wiki/OpenGL\_Programming/Modern\_OpenGL\_Tutorial\_Text\_Rendering\_02. [Accessed: 15- Oct- 2020].

Resource used to help understand building a texture atlas in order to render text to screen.

[16] J. de Vries, "LearnOpenGL - Text Rendering", *Learnopengl.com*, 2020. [Online]. Available: https://learnopengl.com/In-Practice/Text-Rendering. [Accessed: 15- Oct- 2020].

A further resource showing how to render text to a screen, read in order to understand how to render glyphs

# 2 Game reading

## 2.1 General

[8] Noor Shaker, J. Togelius, M. J. Nelson, and Springer International Publishing Ag, Procedural Content Generation in Games. Cham Springer International Publishing Springer, 2018.

‌This textbook provides detailed abstract approaches for a breadth of procedural content generation ranging from the definition, search methods, and evaluation functions to landscape generation and story. It gives great insight into the abstract implementation of the diamond-square and the Perlin noise algorithms to generate height maps used in terrain generation. This book will provide a wealth of resource for generating content.

[19] S. Lembcke, "Adding some perspective to your Unity 2D game.", *Gamasutra.com*, 2020. [Online]. Available: https://www.gamasutra.com/blogs/ScottLembcke/20170825/303794/Adding\_some\_perspective\_to\_your\_Unity\_2D\_game.php. [Accessed: 15- Oct- 2020].

Article written by a programmer of the game ‘Verdant Skies’ in which they implement a 3D perspective camera to render the ground, and detail a method to render 2D sprites by assigning the z-value of the projection matrix to 1 so that no perspective is applied.

## 2.2 Procedural terrain generation

[9] J. Doran and I. Parberry, “Controlled Procedural Terrain Generation Using Software Agents,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 2, pp. 111–119, Jun. 2010.

‌Provides an AI agent based approach to generating landscapes inclusive of hills, rivers, mountains etc. Although no direct implementation is provided, an overview of their works and pseudo code is offered. This approach gives a highly customizable method to generate landscapes given varying parameters such as number of agents, token usage for terraforming, child agent spawn rates etc.

[10] F. Bevilacqua, C. T. Pozzer, and M. C. d’Ornellas, “Charack: Tool for Real-Time Generation of Pseudo-Infinite Virtual Worlds for 3D Games,” 2009 VIII Brazilian Symposium on Games and Digital Entertainment, 2009.

A paper detailing a complete world generation through the tool created ‘Charack’ which utilizes Perlin noise and fractal-based systems for realistic coastlines

[17]"Making maps with noise functions", *Redblobgames.com*, 2020. [Online]. Available: https://www.redblobgames.com/maps/terrain-from-noise/. [Accessed: 15- Oct- 2020].

This website shows various applications of using noise functions to generate maps, gives a good insight into how this may be applied to this project.

[18] A. Patel, "Polygonal Map Generation for Games", *Www-cs-students.stanford.edu*, 2020. [Online]. Available: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/. [Accessed: 15- Oct- 2020].

A further website detailing how to generate a world using noise functions. This websites implementation uses Voronoi graphs to create the map, but does also deal with biomes, rivers, roads, and moisture to assign forest areas or deserts. Comes with a lot of further research material links also.

# 3 External Libraries

[2] Geelnard, M. and Löwy, C., 2020. *GLFW*.

The GLFW library is used to provide a window, attach a context for OpenGL in which to render graphics, and record input related events such as keyboard presses and mouse presses. This library comes with numerous initialization settings which is utilized in the Window.h/.cpp class upon the programs beginning execution. Also provided is the ability to add callback functions enabling a small block of code, or a function pointer, to be attached to associated event types enabling the creation of a publish/subscription-based event handler class.

[3] Herberth, D., 2020. *Glad*.

The Glad library is a wrapper for low level OpenGL based functions allowing for the rendering of geometry-based objects. This is used throughout the rendering process to draw graphics onto the window context, primarily used in the ‘GE::Graphics’ namespace within the game engine.

[11] Riccio, C., 2020. *GLM*. G-Truc.

GLM (OpenGL Mathematics) is a math-based library containing classes and functions relating to vector and matrix mathematics initially aimed at OpenGL, alongside being able to create Orthographic and Perspective transformation matrices in the renderer. The suite comes with all mathematical functionality required to calculate rendering based expressions.

[13] Melman, G., 2020. *Spdlog*.

Spdlog is a fast logging utility library allowing for custom timestamped logging through use of regular expressions and colour coded messages dependant upon severity of message type. This is used throughout the Game Engine and Application classes for debugging purposes.

[14] Barrett, S., 2020. *Stb*.

Stb is a suite of precompiled header only libraries which provides common OpenGL rendering based functionality. Used within this project is namely stb\_image.h which is used to read pixel data from a given file. This data is then uploaded to the GPU memory and bound before usage.

[20] D. Turner, R. Wilhelm and W. Lemberg, *FreeType*. 2020.

FreeType is a font loading utility used to generate bitmaps and meta data from a ttf font file. This is used for text rendering within the game engine

[21] O. Cornut, *ImGui*. 2020.

ImGui is fast and simple graphical user interface library adding functionality for creating a basic user interface. This is used within the project to create a testing sandbox where various features can be displayed on screen alongside customization of input parameters to the associated functions dynamically.