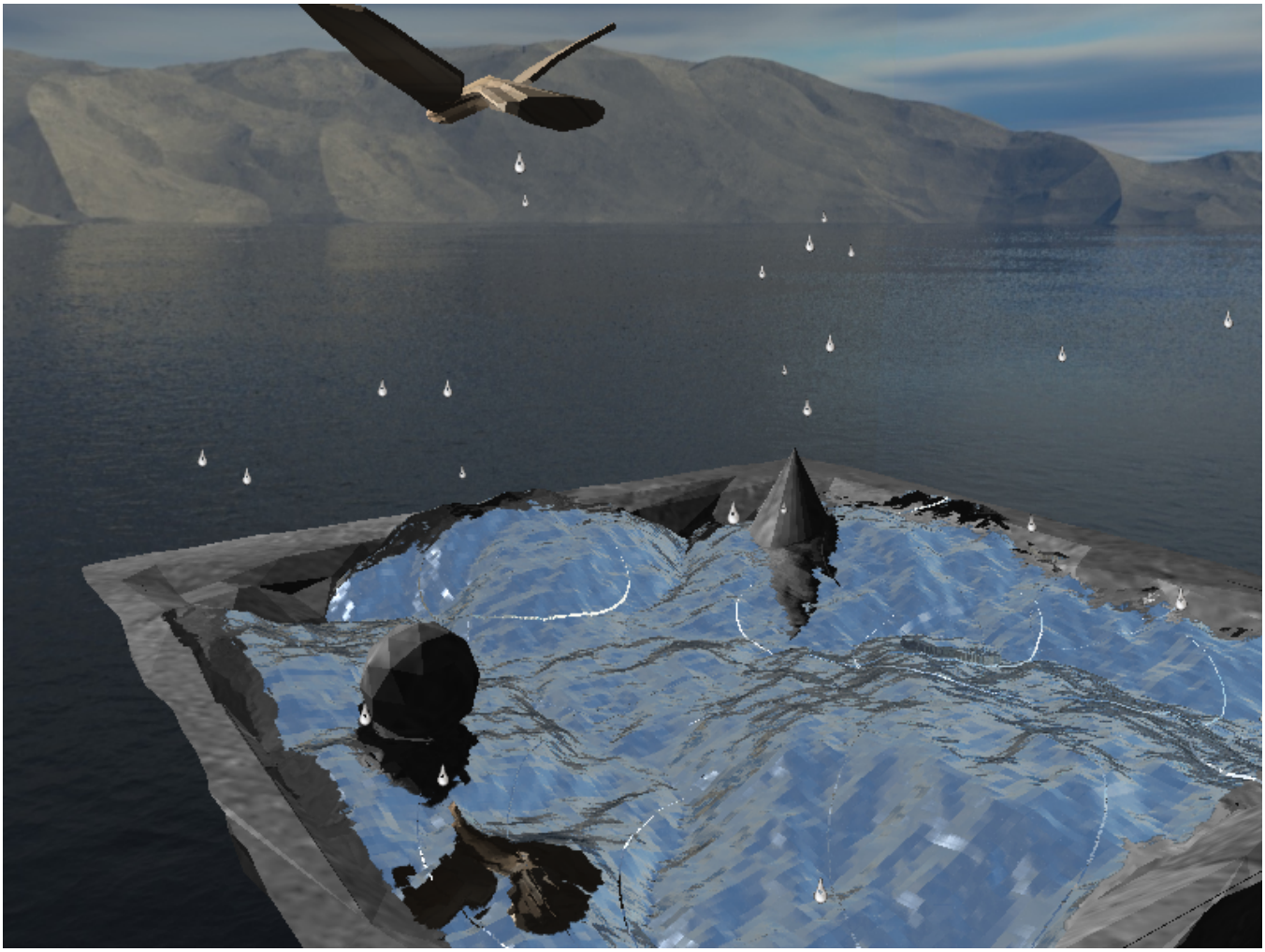
Report

系統概要

本系統使用 C++ 搭配 OpenGL 實作了一個充滿高質感光影效果的水面渲染項目。水面會隨著視角變化而動態調整水面深度表現，並結合光影反射、折射與泡沫的動態效果，營造真實的水體視覺。系統以 OpenGL 為核心，組合 GLSL 的 vertex 與 fragment shader 實現水面的動態程式渲染，模擬不同光照與觀察條件下的水面效果，將真實水面特性與視覺表現相結合，以達成教學與程式設計學習之目的。

Demo



水面

水面幾何結構與波浪模擬

水面是由一個橫向 1000 個、縱向 1000 個格點構成的平面網格。這些格點的 y 軸高度由一系列同步更新的波形高度圖（height map）根據時間變化，其值使用 FFTW 進行快速調和

計算，使用 [Tessendorf 2001] 中基於統計的 Phillips 頻譜，從而模擬水面的動態變化。後續疊加漣漪造成的高度變化後，在 vertex shader 中使用相鄰位置的偏移計算出法向量，用來計算光照效果。

光照與反射折射

水面渲染主要包含四個部份：物體反射、天空（背景）反射、折射以及 Phong Shading 中的 Specular Reflection。反射的部份先將鏡頭位置鏡射後對水面上的物體進行一次渲染並放入 Frame Buffer 中。背景反射根據視角反射的方向，在環繞場景的立方體 skybox 材質進行採樣。折射則將物體 y 座標進行壓縮後對水面下的物體再進行一次渲染放入 Frame Buffer。Specular Reflection 根據水面法向量與視角的夾角計算強度。最後再根據視角與法向量的關係計算 Fresnel 係數，依照不同比例混合折射與反射的效果。

水深計算與調色

水深資訊在本系統中扮演著兩種角色。首先，我們將 screen-space depth 還原為 view-space depth，並以攝影機位置到水面 fragment 的距離計算水深，用來決定水下區域的折射顏色與深淺混合比。透過此距離計算出的 depthFactor，將原始的 refraction.rgb 與一組藍綠深水色進行混合，使水面在視覺上產生由淺至深的自然過渡。

白沫效果（Foam）

泡沫的呈現結合了動態與靜態圖樣。我們首先引入靜態泡沫圖 foamMap 作為基本泡沫形狀參考，再以一張 1000x1250 的 JPEG 泡沫圖 foamTexture 提供細節與動態流動感。此動態貼圖以 uv 倍頻與時間偏移實現動畫效果，模擬自然漂移泡沫，並透過 fragment shader 使用垂直方向的水深（由 waterHeight 減 terrainHeight 而得）決定泡沫出現的位置。

在 shader 中，我們不再使用 view-depth 計算泡沫，而是根據世界座標的 terrainHeight 與水面高度 waterHeight 推導出 verticalDepth，使用 smoothstep 與 pow 函數使泡沫出現在靠岸且水較淺的地區，達成海岸線泡沫聚集的真實視覺效果。

雨滴干擾模擬

最後，系統又把雨滴作為一個動態操作加載，將雨滴擊中水面的地點及時間記錄為 (x, y, t)，在 vertex shader 中計算使用阻尼模擬雨滴對各個位置的水面形狀造成的影響，並與先前 FFT 模你的高度進行疊加，模擬漣漪的波紋效果。為了提昇法向量的計算速度，每個雨滴對其造成的效果不會直接進行計算，而是在高度全部疊加完之後使用 OpenGL 提供的 dFdx 等函數進行計算。

其他物件

雨滴

雨滴使用一個資料結構維護多個質點，隨著時間經過會在水面上方高處隨機產生一些質點，每個質點會以固定速度下墜，並在墜落到水面高度後消失，變成影響水面的漣漪。在渲染每一幀時，每個質點會作為一個矩形傳入 shader，為了在兼顧效率的同時得到較好的視覺效果，我們使用半透明的雨滴圖片當作 texture 直接採樣，並把雨滴的矩形旋轉到面向鏡頭的角度保持它不要變形。

水池

水池的部份我們使用了 [WebGL + Rust: Basic Water Tutorial] 裡使用的模型。我們先使用網路工具將此模型從 `.blend` 轉型成 `.glb` ，之後再使用 tinygltf 來匯入 `.glb` 的模型。而水池是靜態的，所以只要使用 tinygltf 將模型內的 vertex 座標、normal 與 texture 座標等資料取出來，再使用一個簡單的 shader 讓它能正確顯示出來即可。

鳥

鳥的部份我們一樣使用了 [WebGL + Rust: Basic Water Tutorial] 裡使用的模型。不同於水池，鳥是有動畫的，除了前面提過的資料以外，還需要先取出 joints、weights 與 AnimSampler 等資料。當我們想要畫某一時間鳥的樣貌時，我們會先用時間作為參數，使用插值法計算出每一條骨骼的變換矩陣。再來會建造場景樹，並對每個節點都算出它會根據什麼矩陣變換，最後對於每個 vertex 再根據他所使用的關節與權重做線性結合，即可計算出在不同時間的鳥的樣貌。

Work Distribution

- 林伯禧（B11902112）：水面基本波浪形狀、反射與折射效果、雨滴下落模擬
- 盧泓宇（B11902053）：水深計算與調色、白沫效果、天空盒
- 吳柏燁（B11902115）：模型引入與調整、其他物件

Reference

1. Johanson, C. (2004). *Real-time water rendering: Introducing the projected grid concept*. Lund University. https://fileadmin.cs.lth.se/graphics/theses/projects/projgrid/
2. 毛星云. (2021, December 15). *真实感水体渲染技术总结*. 知乎. Retrieved May 29, 2025, from https://zhuanlan.zhihu.com/p/95917609
3. Chinedu. (n.d.). *WebGL + Rust: Basic Water Tutorial*. Retrieved May 29, 2025, from https://chinedufn.com/3d-webgl-basic-water-tutorial/
4. Tessendorf, J. (2004). *Simulating ocean water*. Clemson University. https://people.computing.clemson.edu/~jtessen/reports/papers_files/coursenotes20

04.pdf
5. Syoyo. (n.d.). *Tinygltf* [Computer software]. GitHub. Retrieved May 29, 2025, from https://github.com/syoyo/tinygltf
6. 麒麟子MrKylin. (2022, March 4). *3D渲染技术分享：用实时反射Shader增强画面颜值*. 知乎. Retrieved May 29, 2025, from https://zhuanlan.zhihu.com/p/475696435
7. 麒麟子MrKylin. (2022, March 24). *3D渲染技术分享：实时水面渲染方案(反射、折射、水深与水岸柔边)*. 知乎. Retrieved May 29, 2025, from https://zhuanlan.zhihu.com/p/486631970
8. u010206379. (2023, May 19). *Cube Map 系列之：手把手教你 实现天空盒（Sky Box）*. CSDN. Retrieved May 29, 2025, from https://blog.csdn.net/u010206379/article/details/130764958