Big HW 2

ACG

Mini-game engine for RPG Game

General mentions

• For this project you will work either in teams of 2 people or alone.

• The homework will be uploaded on Moodle. **One upload per team** is enough!

• The homework must be submitted until Monday, the 24th of January 2022, 08:00. No late submissions will be accepted.

• The evaluation of the project will take place during the last lab of the semester (Lab 14). Presence at the lab will be **mandatory** for presenting your projects.

• The final submission will also contain a README file in which you will specify all the functional sections of the project. Additionally, if you have parts of the homework that don't work, you may offer ideas for a partial score.

• If you have any questions, **PLEASE USE** the special channel of your group on Teams. If you know how to answer a question asked by a colleague, you can reply to that post and you might get a small bonus if the answer is correct.

Attention! You are not allowed to reply with code fragments, you can only explain concepts / possible causes of error. Questions that seem to "reveal" too much from the solution of the homework will be removed ☺

Tasks

By using **the game engine framework from lab 9 (and uploaded in the description of Big Hw 2)**, implement advanced functionalities required for a 3D RPG game. You **MUST** keep all modern OpenGL concepts (this includes shaders, vbo, ibo, vao and so on). Deprecated OpenGL functions are not allowed in your code.

Create a 3D scene of your own liking. It should include at least:
- a terrain (mountains / hills), with an appropriate texture
- water (river, lake) with waves, with an appropriate texture
- multiple static objects of your choice with textures or colors that can make your game look nice: trees, animals, any other element of décor that you find suitable (pay attention not to use elements that are very high poly because of performance limitations)
- a dynamic effect / animation of your choice (e.g. fish jumping out of the water, a boat / ship navigating, day-night cycle etc.) – create the animation in your scene over time OR make it happen when the user presses some keys

----

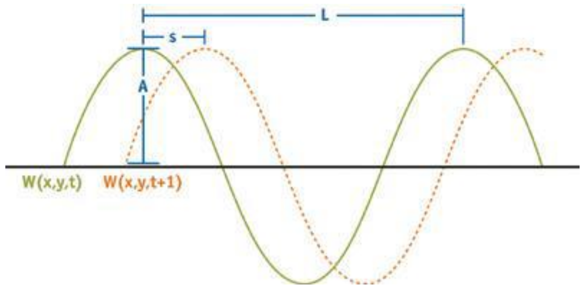
Possible solutions for simulating water:

1. Because we suppose that we work in a 3D space where y represents the height, we can deform this y coordinate based on a formula:

$$y(x, z, t) = 2 \times A_i \times \left(\frac{\sin(D_i \cdot (x, z) \times w_i + t \times \varphi_i) + 1}{2} \right)^2$$

   Where:

   - Ai = amplitude of the wave
   - Di = wave direction (depends on x and z)
   - Wi = wave frequency (w = 2π/L), where L = wave length
   - φi = phase (depends on speed, φ = S x 2π/L), where S = wave speed
   - t = time
   - We noted with • the dot product and with x the classical product (NOT the cross product)



This is called a directional wave. The wave direction stays constant (sent from the main program)

2. We can also have **circular waves**, where the direction should be calculated in **each vertex**:

$$D_i(x, z) = \frac{C_i - (x, z)}{|Ci - (x, z)|}$$

Where:

   - Ci = center of the wave
   - Di = wave direction (depends on x and z)
     (and introduced in the same formula as above)

More info here: https://developer.nvidia.com/gpugems/GPUGems/gpugems_ch01.html

3. Advanced effects: Gerstner waves / FFT (bonus)

Obs: You can choose how many waves you want in order to obtain a realistic effect. You can combine them as you wish (directional + circular, 2 x directional etc.), but you will get the max points if you have at least one type with a realistic effect.

Important! In order to have a realistic effect, the grid for the water should have at least 150 x 150 vertices.

Add a player with first person perspective in the scene (**FPS camera attached)**. We must be able to move realistically in our scene so make sure our player **cannot fly** (make it translate and rotate only on the 2 corresponding axis). Make sure to include also the **rotation** of the camera on the 2 axis.

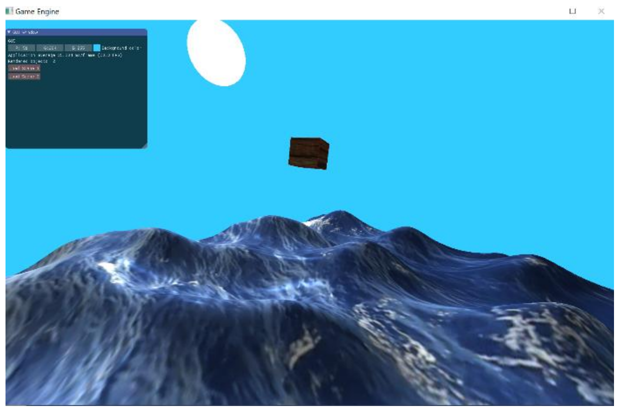
If the player moves in the scene and collides with an object (e.g. tree, stone etc.), a **collision** should be triggered which does not let us move in that direction anymore / display a message. You can use a very simple approach (e.g create bounding boxes or bounding spheres: https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection) or more complex ones, with an extra bonus (octrees, BSP trees etc. https://www.gamasutra.com/view/feature/3099/bsp_collision_detection_as_used_in_.php?print=1 and https://blog.kitware.com/octree-collision-imstk/).

Create a simple **2D GUI System**

- Display any information that you consider relevant (e.g. no of objects visible in the scene , window size etc.) in a 2D GUI window (example below)

- Include **at least** one action element (button, input text etc.) that performs a change in the scene (e.g. a button that will change the background if pressed)

- You can use any external library you want (e.g. Dear imgui)



1) (1.5p) Create a realistic scene, including mountains (0.5p) and various static objects as mentioned (1p).
2) (2.5p) Create the water effect (2p mathematical correctness, 0.5p if it looks good)
3) (2p) Implement the collision detection mechanism of the player and test it in your program.
4) (0.5p) Make the correct changes and add the FPS camera of the player in your scene.
5) (1.5p) Create the animation of your choice
6) (1p) 2D GUI system (0.5 action element which performs changes)

1p for the app rendering with no errors and no crashing

IMPORTANT: In order to see DIFFERENT effects on mountains, water and on the static objects, you should create different shaders for them, based on their behavior.

BONUS (0.5p skybox, others varying): Add a skybox in your scene or any other element which brings realism or fun ☺
BONUS (0.5p) for using a complex collision detection mechanism (such as BSP trees or octrees)
BONUS (varying): Implement water waves using an advanced model (e.g. Gerstner) and explain your method