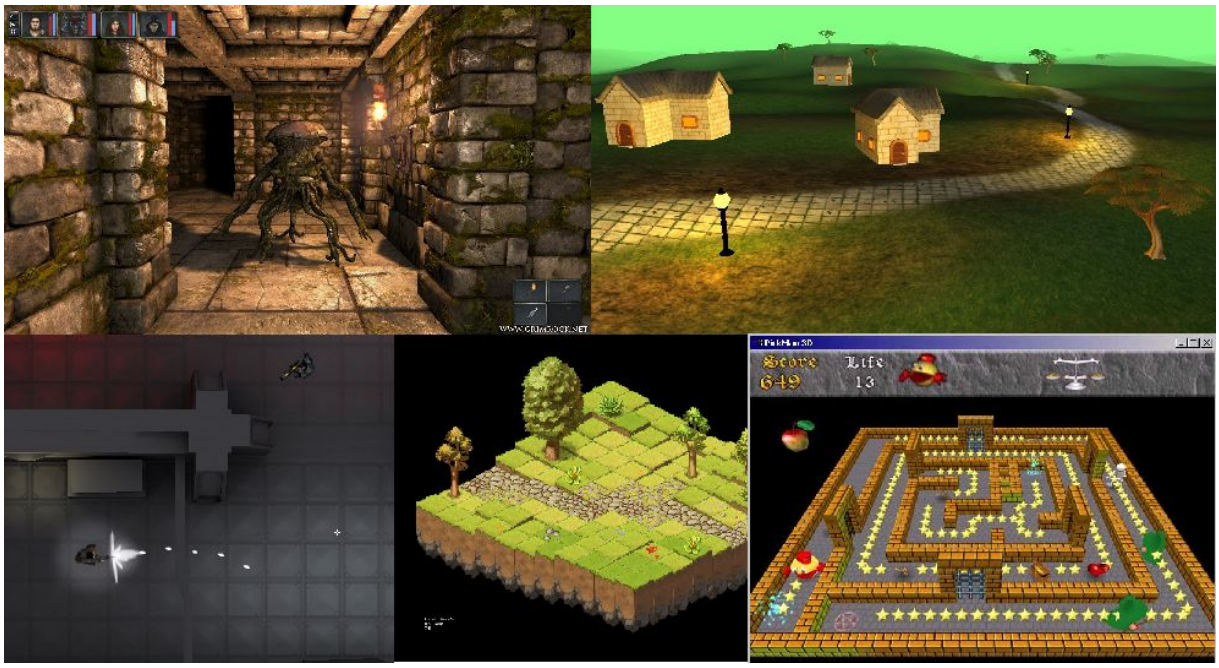
IN4152 - 3D Game Project



Introduction

In your final project the goal is to create a three-dimensional game. In this game you will use all of the past skills you have acquired from the previous practicals. Above are some examples of a game you can take inspiration from, but feel free to do something completely different as long as you make some implementation of the requirements outlined below:

Requirements

Your game should feature a **main character, modeled by you [Blender Practical]**, that can move around in 3D space using some input mechanism like the keyboard or mouse. The rotation and translation of the player should be accomplished using **matrix transformations [OpenGL Intro Practical]**. The view the player has on the scene is determined by setting **projection and view matrices**. You might want a first-person or third-person or birds-eye type of game, therefore the position of the camera is up to you.

The environment should consist of a 3D model **modeled by you [Blender Practical]**, including the house that you have built.

The environment and objects inhabiting it should be shaded in ways you have been taught during the course. This means you should implement the **(Blinn-)Phong Shading Model [Shading Practical]** with a diffuse and specular term. In addition at least one character should be shaded using the **Toon-Shading model [Shading Practical]**. In addition to the shading. You can optionally add textures to the scene that you can download from the internet or make yourself. The scene would look rather fake with just shading as objects don't cast shadows on each other. Therefore, you should simulate sunlight by applying **shadow mapping [Shading Mapping Practical]**. The shadows will probably look pretty rough, so add some kind of **filtering** (like PCF) **[Shadow Mapping]** to them to make them look better.

The game should contain at least one other character (possibly an end-boss) that consists of **several animated components [OpenGL Intro Practical]** (e.g., a robot arm or a snake, or many objects rotating like a solar system). The 3D models required for these other characters (and potentially your own character) can be downloaded from the internet. One character though should be an animated character using skinning, which you should produce by exporting your animated model into individual frames in form of .obj files **[Blender Practical]**

Finally, your main character should have one special power, which is X-Ray eyes. It allows to **[look behind the first occluding layer [Shadow Mapping]**. In this special mode, the shading should switch to the **X-Toon-Shading model [Shading Practical]** with a special effect for the distance.

For a bonus, add other cool graphics features, such as (but not limited to):

- Particle effects (explosions, magic spells, fire)
- Point-lights (like torches or light bulbs)
- The illusion of an infinite terrain by adding new tiles on the fly
- Post-processing effects
- More complex shading techniques, such as normal mapping
- Terrain based on a 3D height field
- Water surfaces that are procedurally generated (produce a vertex grid and move the vertices according to a combination of several sinus functions that you can evaluate in the vertex shader).
- Animated textures (by switching textures frame to frame)
- Zoom effects / perspective changes
- Changing lighting conditions (like a day-night system)
- A minimap (by rendering from a camera high up in the sky, and displaying it on a quad)
- An interactive user-interface
- Collision between the player and objects
- Generate a maze or dungeon for the player to move through
- Generate environment props like buildings procedurally.

Feel free to come up with your own extra effects and features, the above list is by far not exhaustive!

For the project you are not judged on the length, fun, functionality, or quality of the gameplay, but rather on the graphics techniques you used and how well they were applied. In short, a boring game can receive a good grade! Of course, a nice presentation, creativity, or breathtaking visuals are a bonus.

It is advised to work in **groups of two-three people**, three is the maximal number.

Please upload your contribution before the start of the last session, which will be scheduled on Friday 12th of June at 15:00 (during the practical session). The project upload should include: the **source code**, a **short report** (max 150 words) of the **implemented techniques each accompanied by one or multiple figures** and including a **list of the work done by each individual group member with a percentage indication** (not included in the word count). For each project, it is sufficient if ONE group member uploads all components.

Please also prepare a **5 minute presentation** (including a **live demo**) of your work to present during the last practical session, which will be recorded for internal grading purposes and not diffused online.

Framework

To help you get started we provide a framework similar to the ones used in the previous practicals. The framework is build using CMake in the same way as the previous ones. You are allowed to take code (such as camera controls) from any of the previous practicals. Feel free to extend the provided framework in any way you like but external libraries have to be indicated in the report and you will not recieve points for these elements. Hence, it is better to program all functionalities yourself.

Compared to the previously provided code there are three noteworthy changes:

- Meshes are now abstracted away into the `Mesh` class. Use `mesh.draw()` to bind the Vertex Array Object and draw the triangles.
- A similar abstraction class has been created for textures. Call `texture.bind()` to bind a texture at the given texture slot.
- The main loop and keyboard handlers have been moved into an Application class to make the code more maintainable.