IN4152 Final Project Description 2023

In your final project the goal is to create a three-dimensional game.

The theme of the game is up to you. You can think of completely different scenarios, such as a game with bears chased by bees in search of honey… or a water world with mermaids trying to protect their kingdom from sharks with lasers… or a racing game with special vehicles ...

The most important is that you use all the skills that you acquired during the practical sessions and the knowledge from the lectures. For this, you can integrate the code directly from the practical into the provided framework. A few points you will need to implement from scratch using the lecture material. Nevertheless, also here, you have the freedom to decide to start directly from your practical code, if you prefer. The important requirements are outlined below.

Requirements

Modeling: your game should feature a **strong** main character, modeled by you [Blender Practical], that can move around in 3D space using some input mechanism such as the keyboard or mouse.

Transformations: The controls of player should be accomplished using matrix transformations [OpenGL Intro Practical]. The viewpoint in the game should use projection and view matrices. There should be at least two different viewpoints. For example, a top view and a third-person camera that follows the character. The position and movement of the camera is up to you. You can also create a cutscene to add new viewpoints.

Environment: The environment should consist of a 3D model modeled by you [Blender Practical], including the house that you have built. You can enrich your game by using 3D models downloaded from the internet (make sure you have the correct rights to use the models).

Shading: the environment and objects inhabiting it should be shaded in ways you have been taught during the course [Materials and Textures]. This means you should implement at least the (Blinn-)Phong Shading Model [Shading Practical] with a diffuse and specular term. Advanced shading solutions with more complex material models [Advanced Materials] are very welcome.

Toon Shading: Your character should have some special powers. While using the special power the character should change his shape, for example, turn into a different model or grow/shrink. When the special power is applied, shading should switch to the X-Toon-Shading model [Shading Practical] (applied for selected objects or the entire scene). The second axis of the X-toon could be linked to a timer for how long the special power lasts – or something else.

Textures: add textures to the scene. These can be made by yourself or downloaded from the internet (again, make sure you have the rights). Employ these textures also for simple material definitions (control the shininess or use them as normal map...) [Material and Textures].

Illumination: to increase realism, you should include light sources and apply shadow mapping [Shading Mapping Practical]. The shadows will probably look pretty rough at first, so add filtering (like PCF) [Shadow Mapping Practical] to make them look better. Implement at least one special light source which emits light only inside a cone using a light falloff [Shadow Mapping]. Integrate a mechanism to determine whether a player is receiving light from this cone (i.e., is seen by the monster or in recharge zone) and integrate it in your game.

Transparency: Add transparency effects to the scene (at least two layers of transparency). For example, involving a smoke effect by rendering view aligned quads with a texture on top that are drawn with semitransparency (texture sprites). To sort the quads correctly, make use of the peeling strategy [Shadow Mapping].

Animation: The main (or another) character should rely on skinning, which you should produce by exporting your animated model into individual frames in form of *.obj* files [Blender Practical]. The game should contain at least one model 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, or some vehicle composed of multiple parts). Opt for at least four hierarchy levels.

Paths: At least one feature of your game should make use of smooth paths using Bézier curves [Animation lecture]. It can be the movement of a model, a light source, or a camera path at some point, for example, during a cutscene. Use at least one Cubic Bézier curve for the path.

For an additional bonus, add other cool graphics features, such as:

- Path with composite Bézier curve
- Move at constant speed along Bézier curve
- 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!

Hand-in and Rules

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, gameplay, creativity, or breath-taking 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 13:00 on Thursday 6th of April (see Brightspace). The project upload should include: the source code, a short report (max 150 words) of the implemented techniques each accompanied by one or multiple screen shots to well illustrate the method and 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.

In case of strong discrepancies in the group, we will opt for an individual grading based on the provided overview.

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.