Computer Graphics, 2020

Team Project (T1): Your Own OpenGL Games

First, refer to the general instruction for assignment submission, provided separately on the course web. If somethings do not meet the general requirements, you will lose corresponding points or the entire credits for this assignment.

1. Goal: Creative OpenGL Games

Game is still one of the common applications of interactive computer graphics. Modern games appear realistic more and more, and special techniques are becoming increasingly important. While the majority of efforts to realize games rely on the great modeling of virtual worlds and physics simulation, we also need a good design of a game engine for real-time performance as well as effective user interfaces and experiences.

In this assignment, you are asked to implement a game that uses OpenGL for main graphics API. 3D games are the most appropriate application in this course, but you may implement 2D games as well. In general, 3D games would be more difficult than 2D games, because you need to load 3D models and locate camera movements properly. Though, I do not assign more weights on 3D games than 2D ones.

Scene Design It would be hardly possible to create a complete game in a limited time budget. Hence, choose a specific example of a scene, and implement everything necessary to the specific scene, including mouse/keyboard interaction, sound response, and special effects to emphasize and shine your games.

No Existing Engines and WebGL Do not use neither WebGL nor commercial/open-source engines (e.g., Unity3D, Unreal engines, SDL, or SFML) for your implementation. On the one hand, they greatly reduce the amounts of work, but on the other hand, they would not be useful for improving your implementation skills.

2. Grading

TAs and graduate students working for CGLab will also participate in the evaluation of the project.

Examples of evaluation items are listed in what follows.

- Creativity in the design and scenarios (30 pt)
- Fun and humor (10 pt)
- Fidelity of implementations:
  title screen (5 pt), ending/credit screen (5 pt), fullscreen graphical help (5 pt), 3D shading (5 pt), in-game reset/continue (5 pt), 2D/3D character animation (5 pt), text rendering (5 pt), textured 3D skybox (5 pt), bump mapping (5 pt), dynamic 3D camera (5 pt)
- Report (10 pt)

Extra points can be given for the advanced implementations:

- Toggling between windowed and fullscreen games (5 pt)
- Full-screen menu for working game options (5 pt)
- Your own hand-drawings for image sprites (5 pt)
- Particle systems (e.g., fire, water spray, snow; 10 pt)
- Reflection (5 pt)
- Refraction (5 pt)
- Use of a physics engine or your own physics (10 pt)
- Networked multi-player game (10 pt)
- Dynamically moving 2D NPCs with AI (10 pt)
- Animated dynamically moving 3D NPCs with AI (20 pt)

You get penalties for the following cases:

- Your game speed is machine-dependent (-30 pt).
- Your work is not demonstrated (-50 pt).
- Powerpoint (or similar) presentation is not prepared (-50 pt).

3. Oral Presentation

Your team needs to orally present your work. This presentation plays a significant role in rating your project, and hence, pay much attention to provide the best presentation to the audience.

- Students need to orally present the project.
- The presentation language should be English.
- The presentation time can be less than 5 minutes.
- The demonstration of playing the game should be included as well; show us impressive game play and features.

This presentation plays a significant role in rating your project, and hence, pay much attention to provide the best presentation to the audience. I strongly suggest that you let the most important/fluent student present the work.

Also, you need to demonstrate your implementation at the presentation. This is very important for your score. Please try to present impressive demonstration. Make sure your program runs in the test environments without problems; please test in advance for a fluent presentation.

3.1 Video Presentations only for 2020

In this year, your presentation needs to be online. Having offline presentation is the best, but the situation from the Covid-19 virus makes us to have the presentation online. Since having real-time streaming presentations are technically difficult, you need to submit a video clip that records your presentation. The collected videos will be broadcasted in real time. Then, short QnA is followed in the chatting window of the streaming application (e.g., Webex).

The suggested guideline for the video clip formats is like this:

- Video length: 3–5 minute
- Container: mp4 or mkv
- Codec: X264 or H264
- Resolution: 1920×1080
- BPS: 1200 Kbps (cbr or vbr)
- Frame rate: 24

You can use any recording tools, but my recommendation is the use of Open Broadcasting Systems (OBS), which is a open-source software for the real-time streaming and recording. You can download the application from https://obsproject.com/. I do not explain how to use it, but it is not that difficult. You can share your computer screen and your video as well. Please try to embed your webcam shots in the video, as I do in the lecture.

4. What to Submit

- Team name (or ID) and members
- A text file that indicates a URL to download your video; do not directly upload the video, because iCampus will complain about this.
- Source, project, and executable files
- A PDF report file: TEAMID-TEAMNAME.pdf. Please explicitly indicate which features you implemented in the game not to miss corrsponding credits.
- Compress all the files into a single archive and rename it as TEAMID-TEAMNAME.7z.
- Use i-campus to upload the file. You need to submit the file at the latest 23:59, the due date (see the course web page).

5. Using Third-Party Libraries

5.1 Policy on Open-Source Libraries

Obviously, you cannot make all the things necessary for the project. So, you are basically allowed to use open-source libraries for non-trivial functions. More specifically, you can use the projects in GitHub or similar public open-source repositories for this course. The functionalities are only limited to handling low-level functions, which you cannot easily implement.

5.2 Example Libraries

Examples include 3D model loaders, image loaders, 3D animation libraries, Physics simulation libraries, sound helpers, GUI library, text rendering libraries, and similar stuffs.

Loading external 3D models Your own program-based modeling will be limited to a sphere in practice. For sophisticated games, you need to load existing external 3D models and textures. A lot of free 3D models and texture images can be found on the web. When you pick up such models, please check their licenses carefully; use non-commercial models that allows academic/personal uses.

There are 3rd-party libraries to load such models (e.g., *.3ds, *.obj). A common example is the ASSIMP library (https://github.com/assimp/assimp), but it would be too complex to use. LIB3DS (https://code.google.com/archive/p/lib3ds/) or TinyOBJ (https://github.com/syoyo/tinyobjloader) can be easier alternatives for 3ds and obj models. Learning how to compile and use them is also a part of this assignment, which is a common practice in the industry. Their repositories typically contain example codes for OpenGL as well.

Text rendering with truetype fonts Text rendering is also a crucial part of the game, which shows statistics and states of the game. The text rendering can be done in three steps: 1) loading of truetype fonts, 2) baking the characters into a OpenGL texture, and 3) repeated rendering of characters in the GL texture based on an input text. You use a third-party library only for the first step (i.e., loading of external fonts).

Freetype (https://www.freetype.org/) is a common option for font loader, but is often too heavy and overkill. A simpler effective alternative is stb_truetype.h (https://github.com/nothings/stb/blob/master/stb_truetype.h), which is a single-header library; example codes are also embedded in the header file itself. I recommend you using the latter for a compact implementation.

Sound rendering Sound rendering is another crucial component for exciting games. There are tones of libraries for the sound rendering. You may search over the web. Examples include OpenAL (https://www.openal.org/) or irrKlang (https://www.ambiera.com/irrklang/), which is free for non-commercial use. Personally, I suggest to use irrKlang over OpenAL.

Physics Engine When you implement a 3D game, a physic engine is often necessary to handle many physical events, including collision, gravity, and shooting. There are many physics engines on the web. A famous one is the bullet physics engine (http://bulletphysics.org). Using a physics engine is challenging, so when you really need it, try to include it.

OpenGL-Based Embedded GUI When you have many parameters or want intuitive control over certain effects, graphical user interfaces (GUIs) are help much. There are tons of GUI toolkits, ranging from native to embedded GUI. One of such examples, which can be easily integrated with the OpenGL backend, is “dear imgui” (https://github.com/ocornut/imgui).

5.3 Available Samples on Couse Webpage

You may use the course samples provided in the course web page, such as 3D model loader, text rendering, sound rendering, and particle systems. Also, refer to the accompanying documents for usage.