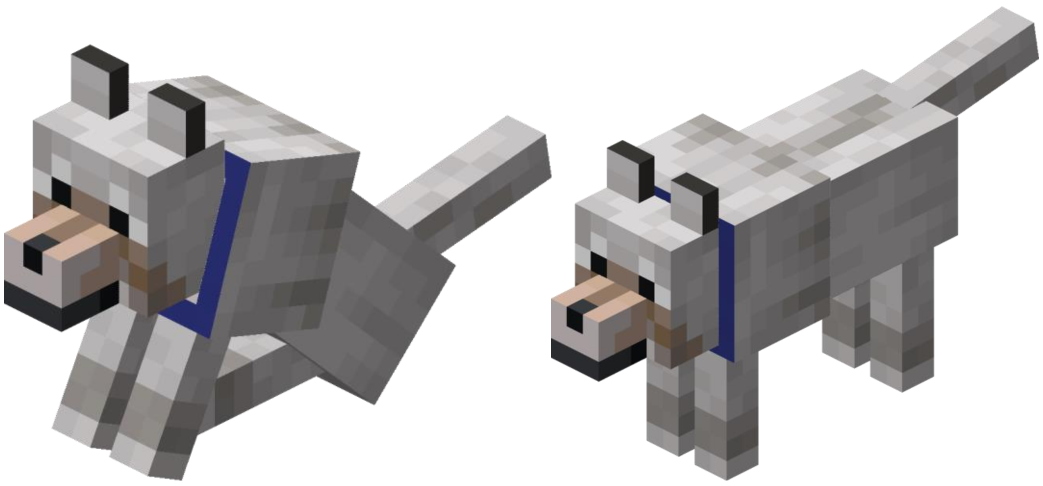
COMP2004 Computer Graphics Assignment

<u>“Rescue Mission” using OpenGL</u>
Theme : Save Sven from Water Sheep

1. INTRODUCTION

First of all, if the assignment theme does not make any sense to you, do not worry about it. Basically, I would like you to create a **simple game** where you have to rescue your wolf pet named “Sven” from the mastermind villain named “Water Sheep”.

Sven



(© Minecraft, Wiki link here)

Water Sheep

(© Minecraft, Wiki link here)

It is not necessary to design Sven and Water Sheep exactly like the pictures above, those are for your references. If you are curious about the origin of their names, feel free to search it in the Google.

The narrative looks like this:

- You will create a simple game arena such as abandoned ruins, spooky forest, forgotten tomb, villain headquarter, etc.
- The main character in the game is represented as the camera (first person view). If the main character is holding an object, it should be visible on the viewport.
- The arena will be darkish (not completely pitch black) and you only have one light source which you can pick up and bring with you such as torch or lantern which will illuminate the area around you.
- Sven is injured and sitting somewhere in the game arena. The objective of the game is to pick him up and bring him to the exit/safe zone.
- When you pick Sven up, Water Sheep will get angry and trying to chase the player down (basically coming towards the camera). Water Sheep will also release his minions OR activate traps to prevent you from escaping. I will leave this to your imagination. Some examples include: releasing zombies, spiders, and skeleton to chase you down. The traps might include trap hole, fire trap, spike trap, descending ceiling, etc.
- The player will be defeated if he is hit once by the Water Sheep or his minions or by the traps. When you are defeated, the camera position should appear laying down on the ground (unable to move anymore) and show the prompt “press **<R>** to restart the game”.
- If the player manage to reach the exit/safe zone with Sven , he wins the game.

2. REQUIREMENTS

In this assignment, you can unleash your imaginations and use OpenGL with C/C++ programming to produce a scene(s) that represents what you envision to be associated with the theme. These are the **general requirements** of the assignment:

- This assignment is focused on the technical skills on OpenGL programming. Artistic creativity is still encouraged, but not required. Please spend most of your effort learning OpenGL itself.
- Interactivity exists in OpenGL assignment. For example, you can explore the world by 'walking around' with **<W>, <S>, <A>, <D>** keys and move the camera with the mouse. Furthermore, the provided samples in the tutorial also shows an example of interacting with an object such as pressing a button. There are many other things you can try such as sitting on the chair, automatic door opening, opening a door, picking Sven up, and many more.
- It is possible to have multiple light sources in OpenGL. For simplicity, you only need to have one light source. However, the effect of ambient, diffuse, and specular has to be clearly visible.
- In order to make it more fun and testing your creativity, the only shape you can use for this assignment is **"box"** shape. Yes, you read that right, ONLY "box" shape. NO sphere, cylinder, cone, pyramid, torus, and any other polyhedron. You can apply any affine transformation to it (enlargement, shrinking, stretching, rotation, even shearing) and combine with other boxes to create anything. You can think it as if you are playing toy block (but only box). If you are familiar with video games, it is similar to Minecraft (https://minecraft.net/en-us/). You create the game only with boxes.

Program Requirements:

1. At least 6 composite objects and 6 different image textures. Since you can only use boxes, be creative on combining these boxes to create complex objects (*e.g.* table on tutorial Sample_1). Sven and Water Sheep should be among these 6 objects. Please do not just create a simple white box and call it Sven . At least, make it look reasonable with legs and head.
2. Only one light source is required. You can create something like a simple lantern/candle/torch as the light source. You should be able to carry this light with you. Pressing key **<F>** allows me to pick up the light source (if close enough). Pressing the same key again will put the light source on the floor.
3. Implementing a simple light source attenuation (lecture 7). As the distance to the light source is increasing, the brightness should be reduced gradually. Pressing the key **<K>** will reduce the brightness radius of your light source, while key **<L>** will increase the brightness radius of your light source. You can derive your own formula for the light source attenuation, but describe it in your report.
4. The effect of the light source should be reasonable. For example, metal surface and waxed apple should be shiny, while wooden plate should be dull.
5. As you have noticed from the examples, you can only move forward, backward, left, and right with **<W>**, **<S>**, **<A>**, and **<D>** keys. Please extend that example so that the main character can look around with the mouse movement. However, you are not supposed to fly around. (e.g going forward while looking at the sky is not supposed to make you fly up)

6. Pressing the keys **<spacebar>** shall make you “jump” (like in video games).
7. Pressing the key **<P>** shall toggle the switch between perspective and orthographic projection.
8. At least 2 animations should present in your created scenery. One should start playing the moment the program start rendering, while the other one only plays when you pick Sven up (by pressing **<E>** when he is close enough).
9. Pressing the key **<R>** should restart the game. It will return the position of all the objects in the game back to the original position.
10. Since this is a darkish environment, it might be a bit too dark for me to assess everything. So, please implement a functionality to toggle between the “darkish scenery” and “everything bright scenery” with the key **<O>**. For good examples, you can see the sample_1 and sample_2 I created for the tutorial. Sample_1 just simply disregards light sources, so it looks bright on every surface. Meanwhile, sample_2 is dark until you turn the light on.
11. You need to implement a simple “hitbox” to trigger some events such as *hit by the zombies* or *being close enough to Sven to pick him up*. One of the simplest way to do this is by utilizing the distance from the camera to the object of interest. When the distance is lower than a threshold, it can be regarded as “collision”. Please observe Sample_1 in the tutorial to see how this idea is applied to press the red button.
12. The difficulty level of the game is up to your decision. You can make it extremely easy or extremely difficult to beat, as long as we can assess it properly.

T**he final version of your program must work on the lab workstations and be presented and assessed in the Lab**. No technical support will be provided for your work on your own PC. Please ensure that clear reference is given if you do have to use some external codes and resources.

3. SUBMISSION

This assignment is due on the **28th October 2019, Monday at 9:00am**. You must submit two files:

- A short report of up to 5 pages in PDF format. In the report, you must list the implemented functions and briefly explain the main algorithms you used for modelling, rendering and animation. Please provide specific description on YOUR OWN work, not just a list of what are required. Please describe any simple or composite objects you use to model YOUR OWN objects, and what kind of surface finishing you have employed for each of them. If you use any external tool to create a complicated object or finishing, please provide a brief description and the usage of the tool. You are also required to describe the animation you have produced by providing the design ideas and their implementations. The report will contribute to 20% of the assignment marking.
- A compressed zip file containing:
  - ❖ The whole folder containing your source code directories, texture images, and relevant libraries;
  - ❖ A simple readme.txt which briefly describes each source file and the way to compile and run your code(s) (should be similar to compiling your tutorial codes);
  - ❖ The completed and signed Declaration of Originality. By submitting the sheet, you declare that the work submitted are solely your own work.

Both of these files should be uploaded to the Blackboard (Assessment → OpenGL Assignment Submission 2019) before the deadline. **Please make sure the completeness and correctness of your files before your submission**. **All late submissions will not be accepted.** No hard copy submission is required.

In addition, you should leave a copy of the above files (not zipped) and **the executable file** in your main directory under */CG/assignment/*. Please note that the time-stamp for the source files / executable file must be before the deadline.

During the practical sessions in Teaching Week 12 (Starting on 28th October 2019), you must first show the time stamp of your executable file to your tutor. You must demonstrate and be ready to answer any question relating to your program. You will be asked to make slight changes to the program to clarify your understanding of OpenGL graphics programming or to prove your authorship if any doubt arise. **Failing to turn up at the demonstration in your own enrolled practical session will result in a ZERO for the assignment, and may result in a F-IN for the whole unit.**