Final Project
COMP 371 - Graphics

Submitted To:
Dr. Sudhir Mudur
&
Dr. Charalambos Poullis

Submitted By:
Earl Steven D. Aromin
40004997

12th of April, 2018

<u>Features from Assignment 1:</u>

Created 100x100 square grid

- Simple lines

Coordinate Lines

- Lines with colors (R,G,B) for (X,Y,Z)

Horse Model

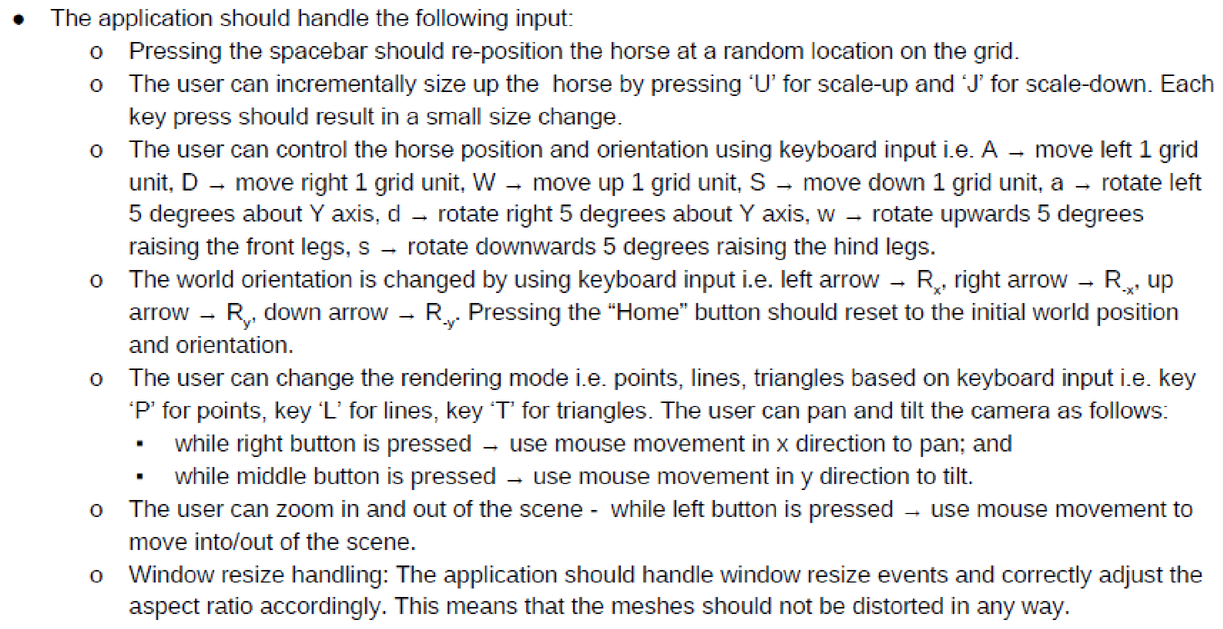
- Used triangles to create a cube
- Used a cube for each part, applying necessary transformations such as scaling, translating and rotating to create a magnificent steed.
- Used slightly different coloring for each part type

Rendering

- Window of 800x800 – can be resized without distorting image due to frame_size_callback(GLFWwindow* window, int w, int h)
- glEnable(GL_DEPTH_TEST) to remove hidden surfaces
- Double buffering supported

Input Interaction

- All mouse and keyboard inputs are processed as mentioned in the assignment



- The application should handle the following input:
  - o Pressing the spacebar should re-position the horse at a random location on the grid.
  - o The user can incrementally size up the horse by pressing ‘U’ for scale-up and ‘J’ for scale-down. Each key press should result in a small size change.
  - o The user can control the horse position and orientation using keyboard input i.e. A → move left 1 grid unit, D → move right 1 grid unit, W → move up 1 grid unit, S → move down 1 grid unit, a → rotate left 5 degrees about Y axis, d → rotate right 5 degrees about Y axis, w → rotate upwards 5 degrees raising the front legs, s → rotate downwards 5 degrees raising the hind legs.
  - o The world orientation is changed by using keyboard input i.e. left arrow → R_x, right arrow → R_{-x}, up arrow → R_y, down arrow → R_{-y}. Pressing the “Home” button should reset to the initial world position and orientation.
  - o The user can change the rendering mode i.e. points, lines, triangles based on keyboard input i.e. key ‘P’ for points, key ‘L’ for lines, key ‘T’ for triangles. The user can pan and tilt the camera as follows:
    - ▪ while right button is pressed → use mouse movement in x direction to pan; and
    - ▪ while middle button is pressed → use mouse movement in y direction to tilt.
  - o The user can zoom in and out of the scene - while left button is pressed → use mouse movement to move into/out of the scene.
  - o Window resize handling: The application should handle window resize events and correctly adjust the aspect ratio accordingly. This means that the meshes should not be distorted in any way.

Features from Assignment 2:

Hierarchical Modeling

- Achieved with CubicPart (child of Part) class. Has pointer to parent and a vector of pointers to corresponding children.
- Transformation from parent cascades down to children with the children themselves having their independent transformations.
- Has pivot point around which the CubicPart will rotate.
- Appropriate textures are also added for horse and floor

Lighting

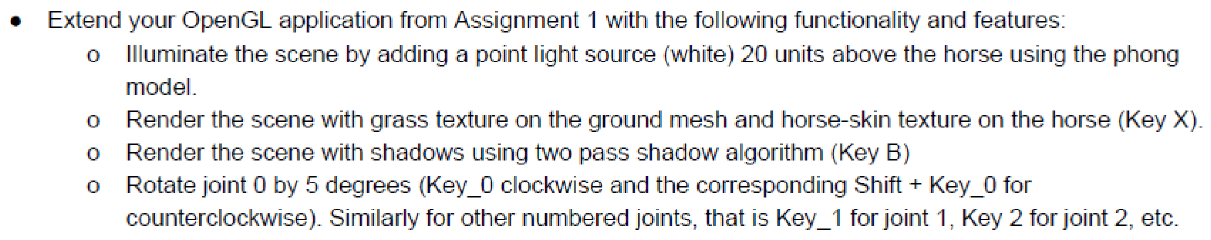
- Phong model achieved by having an ambient with diffusion and specular added. Light source is a point light above the origin +20y

Shadows

- Two-pass for shadow mapping. Takes a perspective view from light source looking at origin and stores the values in the depth buffer on the first pass. When rendering from the camera's perspective the texture is mapped on the occluded (after comparing depth values)

Input Interaction

- Along with Asg 1's key bindings, these are also implemented



- Extend your OpenGL application from Assignment 1 with the following functionality and features:
    - o Illuminate the scene by adding a point light source (white) 20 units above the horse using the phong model.
    - o Render the scene with grass texture on the ground mesh and horse-skin texture on the horse (Key X).
    - o Render the scene with shadows using two pass shadow algorithm (Key B)
    - o Rotate joint 0 by 5 degrees (Key_0 clockwise and the corresponding Shift + Key_0 for counterclockwise). Similarly for other numbered joints, that is Key_1 for joint 1, Key 2 for joint 2, etc.

Extra: Walking Animation

- Inspired by keyframe animations, a walking animation is implemented (gallop())
- Used glfwGetTime() %n for n frames and set specific rotations for each joint
- Expanded upon in final project

Final Project Features:

Lighting Changes

- Set light position to 110 above the origin.

Troop

- 20 Horses with randomized orientations and position on ground.
- <Rotations and moving in animation section>
- Used a Bounding_Sphere to implement collision. Follows the horses' location and does a routine check for proximity each frame. To better illustrate the collision in effect, the horses that are set to **stopped** are doubled in size.

Animations Available

- walko() and stoppo() for assignment 2's walking animation (not used since it doesn't flow well)
- stroll() which takes a random number and makes the horse move forward by it (5-15)
- choose_direction() which has the same randomiser as stroll, but instead is used to make the horse rotate left and right by 15 degrees (a coin flip is done each time to randomise)
- walk() a supplementary animation that is designed to be with stroll
- eat() an idle animation in which the horse bends down and eats

Animation Implementation

- Toggled by 'H' (only other new hotkey)
- Animation speed is determined by horse's speed which is randomized within its constructors.
- Two types of animations shown:
  - eat,choose_direction and stroll are main ones, meaning only 1 of each can be running at the same time, since they depend on the 'frame' variable.
  - walk is not designed to have a predetermined start or end, it just loops. Not being dependent on the current 'frame' value it is used to supplement stroll, which only moves the horse forward
- Frames: To have a more precise animation, an int frame is introduced.
  - Instead of relying solely on the glfwGetTime()%n, which breaks the flow of an animation, it is used to signal a 'tick'.
  - Since we're still using integer values of glfwGetTime(), we can still affect tick speed with the horse speed. Frame increments when current_tick is different from old_tick, regardless of difference to ensure that all frames are visited and there is no skipping.

**External sources are referenced in the 'ReadMe' and 'EXTERNAL SOURCES'