Declaration

> The entire code of this assignment is purely our own work and we have not taken any assistance from other students or copied the code from internet and at any point of time we both will be able to explain any part of the code that we have written.

1. Ashutosh Sathe (21q050012)
2. Animesh (21q050015)

Rendered video link

Controls

Hierarchy controls

- `Alt + Left/Right` – Move to the left or the right sibling of current node
- `Alt + Up` – Move to the parent of the current node
- `Alt + Down` – Move to the first children of the current node
- Numbers 1 through 6 – Select a particular degree of freedom of the current node
- Left bracket (`[`) – Increase the parameter of the selected degree of freedom
- Right bracket (`]`) – Decrease the parameter of the selected degree of freedom
- Period (`.`) – Move to the next animation entity (a parameterized hierarchy node)
- Comma (`,`) – Move to the previous animation entity (a parameterized hierarchy node)

Rider and bike combined movements

- `7` – Moving both rider and bike backward in the direction of rider's view
- `8` – Moving both rider and bike forward in the direction of rider's view
- `9` – Rotating both rider and bike anticlockwise along the selected dof axis
- `0` – Rotating both rider and bike clockwise along the selected dof axis

Light controls

- `F1` – Toggle global light 1 on/off
- `F2` – Toggle global light 2 on/off
- `F3` – Toggle rider spotlight on/off
- `F4` – Toggle bike headlight on/off

Camera controls

- `b` – Selecting global camera
- `n` – Selecting third person camera
- `m` – Selecting first person camera
- `Y/G/H/J/T/U` – Global camera movement

Details of light

We have included 4 lights in the scene, global light 1, global light 2, rider's spotlight and bike headlight. First two lights are point lights which are used to lights up the whole FMX track, whereas the latter two light are spotlights (i,e, whose spread is limited by an angle range). All the lights are modelled using the `Blinn-Phong model`.

For placing the global lights we have chosen some 3D coordinates enables us to light up the scene properly, thus global lights are static. On the other hand, rider's spotlight follows the rider everywhere in the scene and bike headlight position and angle changes as the bike changes the same. For limiting the throw of these spotlights we have limited the angle of light spread to 5 degrees instead of full 360 degrees.

Shading and shadows implementation

We have used the `phong shading` to render the whole scene as it gives the best looking results without need of much tesselations. The colours calculated for every fragment also considers the shadows generated due to the entities in the scene.

For implementing the shadows we have perfomed shadow mapping using textures. It is done for every light source which makes the scene realistic with multiple shadows. We have created a vector for storing the shadow maps corresponding to every light source which later are considered for performing the depth tests repeatedly for every light. Finally the colour of every fragment is calculated by considering each shadow term.

Cameras

We have employed three cameras. i.e. global camera, third person camera and first person camera, to capture the scene from multiple point of views. Global camera can be moved in the world using controls specified above. Third and first person cameras are positioned dynamically. The third person camera tracks the rider automatically whereas the first person camera is positioned in a way to emulate the rider's eyes.

Modeling the bike

Modeling the bike body was challenging due to its irregular shape. We first drew left hand side view of the bike on a graph paper. Labeled each point and then manually generated triangles for this side. Then we ported these triangles into OpenGL and kept transforming it till we were happy with the overall shape of one side. Then we simply reflected the shape about Z-axis to get the complete body. Other parts of the bike were modeled using SPP. Figure 4 shows the overall hierarchy structure. Similar to rider, for successful execution, the generated executable `a2-model-bike` must be executed from the folder containing the parameters of various SPPs (i.e. folder containing `bike_parts` directory) used for modeling the bike.

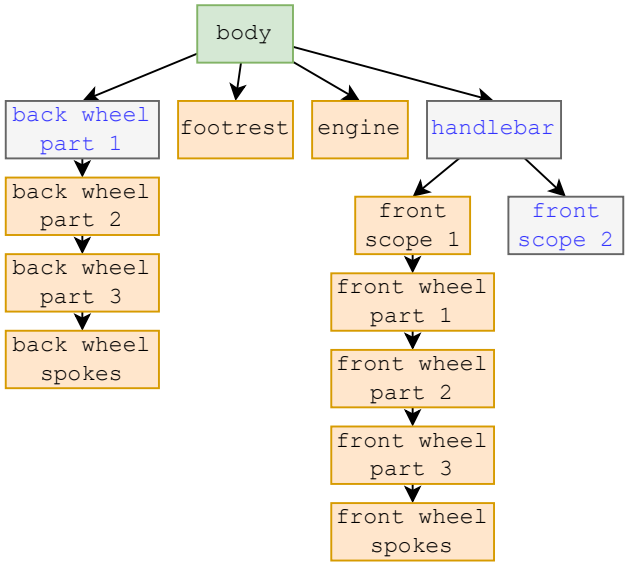

Figure 1: Hierarchical model of the bike. Green node is root. Orange nodes are "fixed". Blue text nodes have 1 degree of freedom.

Modeling the track

Similar to bike body, we drew obstacles on graph paper, labeled each point and generated triangles foe each side. Then we ported them to OpenGL and applied transformations to get the final track. We have roughly scaled the obstacles to be sensible with respect to bike and the rider, however, we will finetune the scale of bike + rider combo a bit before next assignment depending on how it looks through the camera. A lot more detail in the arena – such as the actual path that bike follows, roughness of the surfaces etc. will be brought via textures and lighting in the next assignment.

References

- OpenGL Tutorials – Tutorial 07 – `https://github.com/paragchaudhuri/cs475-tutorials`
- Hierarchical Modeling – `http://graphics.cs.cmu.edu/nsp/course/15-462/Spring04/slides/05-hierarchy.pdf`
- Enabling `glVertexAttribPointer` such that could make arranging a single VBO for hierarchical modeling easy – `https://stackoverflow.com/a/39684775`
- Track obstacle inspiration – `https://www.youtube.com/watch?v=VC1FeM9QuEg`