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

`https://youtu.be/3X2_qD4iIy4` or

`https://drive.google.com/file/d/1YHvtwAg5F9scsN7r4panFjI6Sw0rcVp9/view?usp=sharing`

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/` – Change global camera lookat
- `Alt + Y/G/H/J/T/U` – Move global camera

Lights

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 illuminate the environment using *"Blinn-Phong model"*.

For placing the global lights we chose specific 3D coordinates that enable us to light up the scene properly while also enabling interesting shadows. Global lights are static, on the other hand, rider's spotlight follows the rider everywhere in the scene and bike headlight position + angle changes along with the bike. For limiting the throw of these spotlights we have limited the angle of light spread to 5° degrees instead of full 360° degrees. This is implemented via `includes/light.hpp`.

Shadow Mapping

We follow LearnOpenGL's shadow mapping tutorial along with class notes to implement shadow mapping from all 4 light sources. Since we generate maximum of 4 shadow maps (1 per light) we store them in a `GL_TEXTURE_2D_ARRAY` and sample from appropriate layer in the fragment shader. We combine the result of depth test at every fragment and also perform PCF for smoother shadows. We followed LearnOpenGL's shadow mapping tutorial for this. Main implementation can be found in `lighting_shading_fs.glsl`.

Cameras

We have employed three cameras – global camera, third person camera and first person camera. Global camera can be moved arbitrarily 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 above rider's neck at around eye level to emulate GoPro-like camera. Each camera can also be focussed at a particular point at any given time. This functionality is also used by spotlight to always focus on the rider. Implemented in `includes/camera.hpp`.

Textures

Figure 1 shows all the textures used in the assignment. We were unable to UV map all the complex geometry within time so we just UV-mapped a single quad in track, bike and rider. We also make sure to properly move this texture with bike and rider separately. The texture also interacts with all the lights. All the texture mapped geometry is drawn with `renderTexturedGL(...)` implemented in `includes/main.hpp`. Our skybox code is heavily inspired from the LearnOpenGL's cubemap tutorial although we use our own modeling machinery (`triangle.hpp`, `point.hpp`) in order to accomplish the goal. The skybox texture was taken from `http://humus.name/index.php?page=Textures` and is licensed under Creative Commons 3 license.

Figure 1: All the textures used in this assignment. Top row: Left – bike headlight, Middle – Torso back logo, Right – Track texture. Middle row: negative parts (XYZ) of skybox. Bottom row: positive parts (XYZ) of skybox.

Story of the animation

Josh is learning FMX riding and is practising his first show but somehow the bike is trying to sabotage his performance by not spinning the wheels. Even the GoPro comes loose during the show, but Josh manages to pull through. Will Josh continue learning FMX ? Our original idea was to make this as a meme but we did not have enough time to add music and make the quirks in the animation standout.

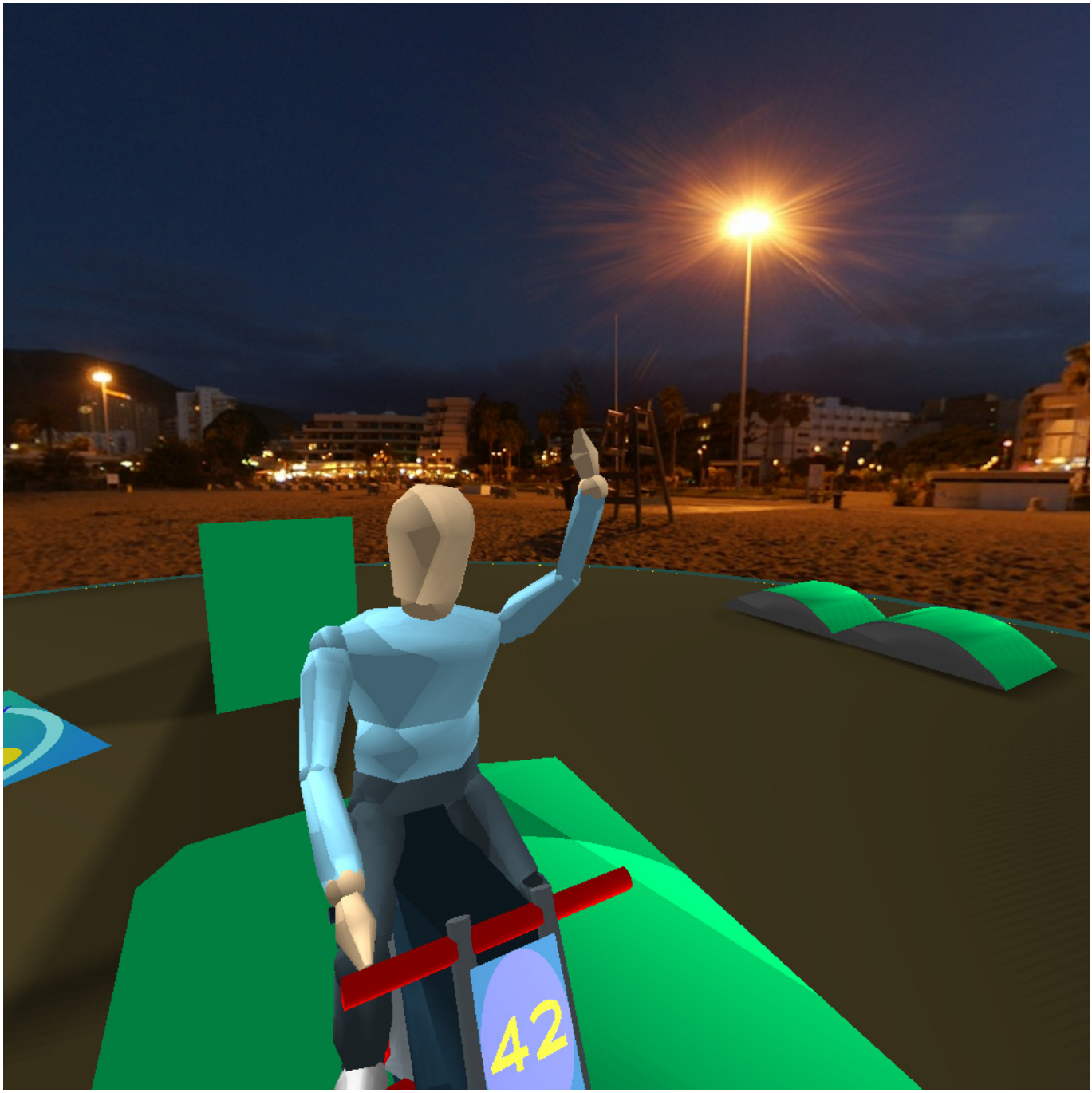

Figure 2: A frame from animation

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`
- Shadow mapping – `https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping`
- Lighting – `https://learnopengl.com/Lighting/Basic-Lighting`