Navigation Techniques

In this assignment multiple navigation techniques such as *Steering*, *Maneuvering*, and the *Navidget* technique have to be implemented.

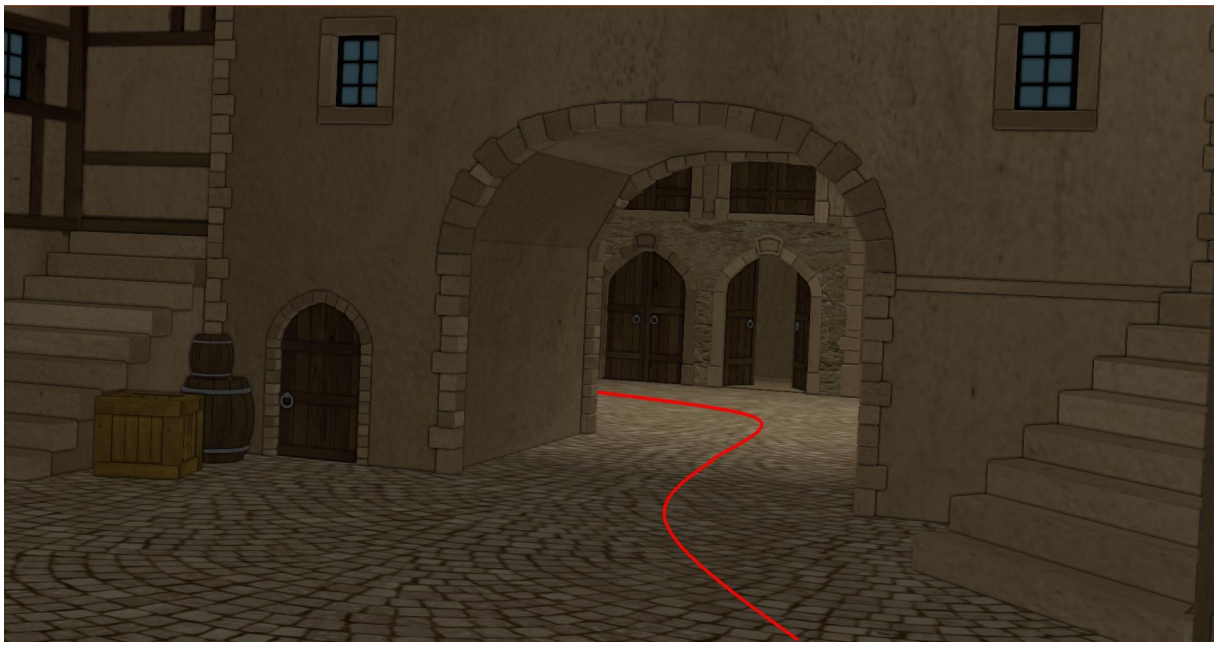

Figure 1: Steering through a virtual town.

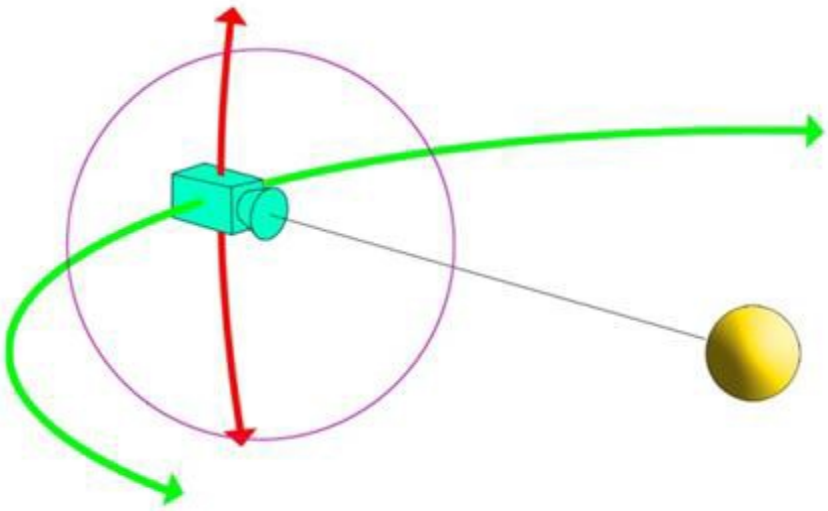

Figure 2: Viewpoint maneuvering relative to a predefined reference point.

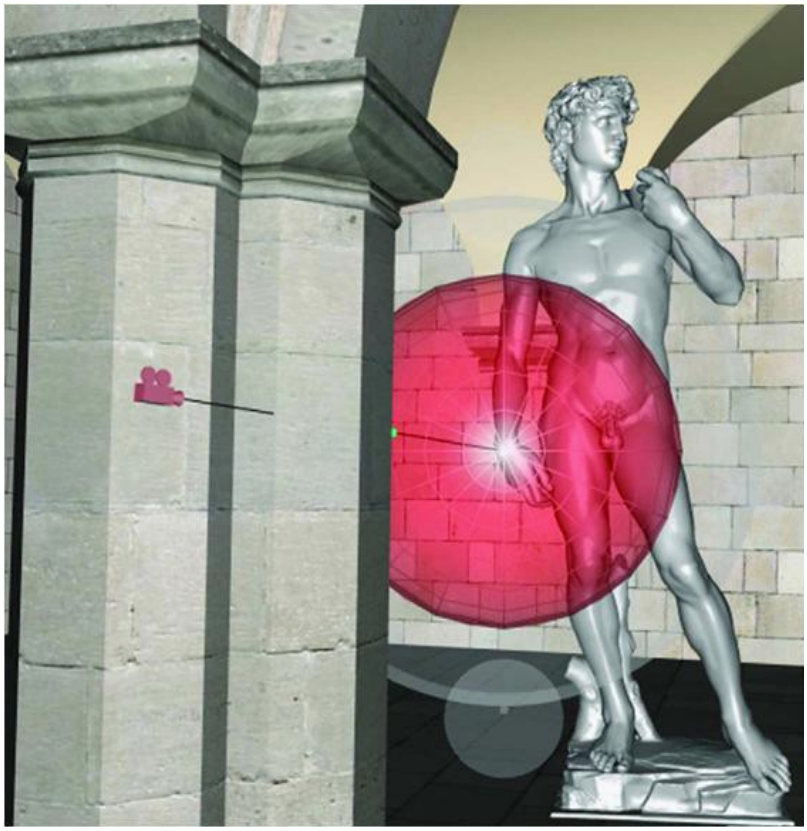

Figure 3: The "Navidget" navigation technique.

Figure 4: Navidget - definition of the reference point in the scene.

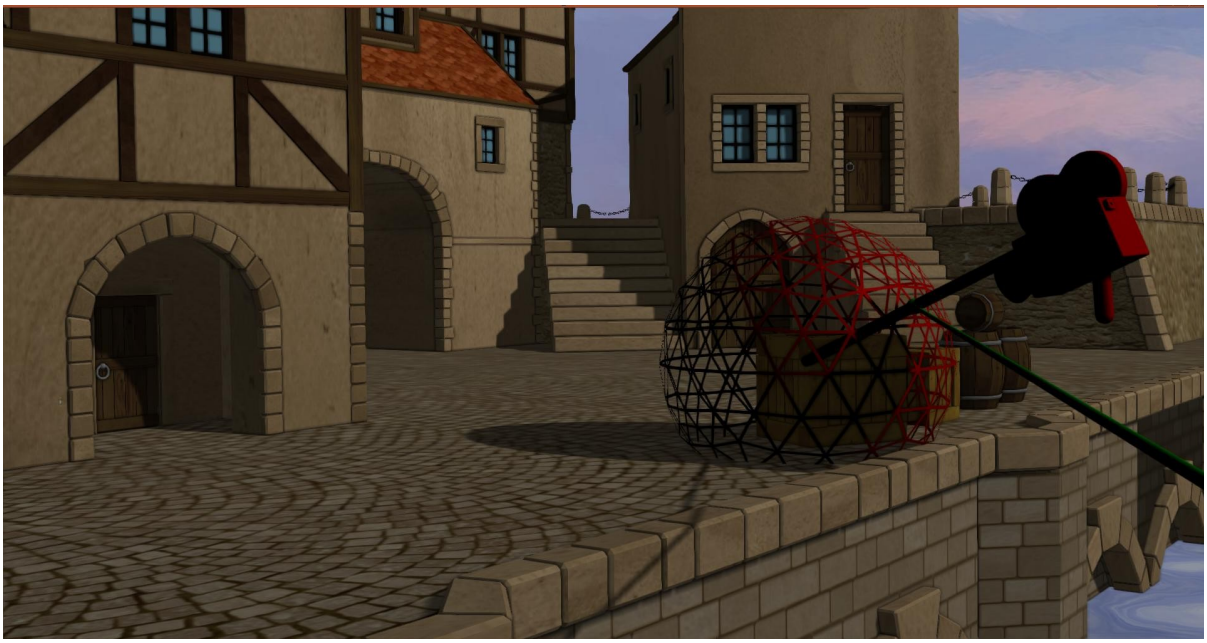

Figure 5: Naviget - definition of the camera position and orientation on the sphere widget.

Figure 6: Navidget - final camera position and orientation after the viewpoint transition sequence.

How to start?

- Copy the 08_navigation folder from /opt/vr_exercises/WS_15_16 to a local repository.
- Execute the application by running ./start.sh in a terminal.
- Start the ART or PST tracking system.
- The Samsung Stereo-TV set has to be set to stereo mode for every session using the remote control (settings are not saved when turning off)
    - Menu -> Setup -> DLP 3D / Dual-View -> ON-STD GLS
- Mitsubishi Stereo-TV usually is set to stereo mode (settings are saved between sessions)
    - Menu -> 3D Mode -> ON Standard
- Verify functionality of the wired shutter glasses. Perception of flicker when looking into a light source (e.g. room light).
- Proceed with the assignments.
- Turn of displays and tracking systems when finished.

Assignment Tasks (graded):

The maximum number of students per group is two. The presentation of the results has to take place until **Friday 5.02.2016**. Make individual appointments for your presentation.

1. [**30%**] Implement a steering-based navigation technique that enables translation and rotation of the camera on all three spatial axes. The scenario is the exploration of a virtual town (Figure 1). As an input device the Spacemouse should be used in combination with rate-control as transfer function type. A non-isomorphic input mapping should enable both rapid and precise camera movements. Apply adequate filter mechanisms to separate the input channels from each other. It should be possible to look-around at a certain place (e.g. head & pitch rotation) without inducing any translations. For distance travel on the other hand, translation (z-axis) and rotation (head & pitch) have to be used in combination. All input values have to be mapped in the actual camera coordinate system. The rotation center for navigation should lie in the users head. Therefore the headtracking matrix of the users is already forwarded into the Navigation class with a field connection.
2. [**25%**] Extend the camera navigation with maneuvering capabilities (Figure 2). Therefore set the rotation center of the camera to a predefined reference point. An intersection ray is already

implemented, which returns the intersection point of the tracked pointer with the underlying scene geometry. The intersection is triggered by pressing the primary pointer button (button layout may differ between different pointing devices). The intersection point is visualized with a sphere at the respective spatial coordinates and represents the external rotation center of the camera. While holding the button pressed, the viewpoint maneuvering mode is enabled. Releasing the button switches the navigation back to steering mode. The three rotational degrees of freedom should be mapped with respect to the external rotation center. For instance, inducing y-axis rotation input should lead to a left- or right-sided lever movement of the camera around the defined reference point. X and y translations can be ignored in this mode. Inducing translations on the z-axis should increase/decrease the distance towards the reference point. Keep in mind that a found intersection point is expressed in world coordinates. Possibly transform this point into the navigation coordinate system to be used as a rotation center. Besides the town scene (key 1) a virtual car model can be loaded (key 2) as a maneuvering scenario.

3. [**45%**] Implement the “Navidget” technique (Figure 3) as a representative of target-based navigation techniques. The functional principles of the “Navidget” navigation technique are explained in the lecture notes (3D Navigation) and in the paper of Hachet, M., Decle, F., Knödel, S., and Guitton, P. 2009. **Navidget for 3D interaction: Camera positioning and further uses**. Int. J. Hum.-Comput. Stud. 67, 3 (Mar. 2009) and in the paper Knödel, S., Hachet, M., Guitton, P. 2008. **Navidget for immersive virtual environments**. Proceedings of the 2008 ACM symposium on Virtual reality software and technology. Both papers are included in the repository. An intersection ray is already implemented, which returns the intersection point of the tracked pointing device with the underlying scene geometry. The intersection is triggered by pressing the secondary button on the pointer. After the reference point is set (Figure 4), the position and orientation of a virtual camera avatar can be defined by moving the ray over the surface of a transparent sphere, which surrounds the reference position (Figure 5). The node *navidget_node* can be used to set the position and orientation of the attached *navidget_sphere_geometry* and the *navidget_camera_geometry* node. The function *get_rotation_matrix_between_vectors(VEC1, VEC2)* in the *Navigation* class returns a rotation matrix that rotates one vector into the direction of another vector. Use this function to determine the orientation of the *navidget_node* and thus all its sub nodes. Releasing the button then triggers a transition sequence, which

animates the actual camera position and orientation towards the predefined camera avatar (Figure 6). The navigation matrix should be animated over a period of time using linear interpolation of positions (use function lerp_to() on a vector) and spherical linear interpolation (use function slerp_to() on a quaternion) for interpolation of orientations.

4. Prepare a short presentation (10-15min) of your implementation using one of the stereoscopic display environments available in the VR-Lab with a tracked pointer. Demonstrate the working prototypes as well as relevant code snippets.