Virtual Reality
Winter Term 2019/2020

Assignment 3 - Input Devices and Transfer Functions

The choice of suitable input devices and transfer functions is crucial for any interaction technique in virtual environments. In this assignment, you will investigate position, rate, and acceleration control mappings applied to the inputs of an isotonic and an elastic device for a simple manipulation task. Furthermore, you will implement basic automated animations of virtual objects towards targets selected by the user.

Group work in pairs of two is encouraged. You are required to submit this assignment by **28 November 2019, 11:55 pm** on Moodle. Furthermore, you will be asked to present and discuss your results in the lab class on **29 November 2019**. Please register for an individual time slot with the teaching assistants on Moodle (one per group). This assignment contains tasks worth a total of **16 points** and will be weighted by **1/6** for your total lab class grade.

Getting Started

Download the source code package from the assignment page on Moodle and extract it to your local hard drive. You can start the application by typing `./start.sh` on a terminal in the extracted directory. This will set all environment variables correctly and execute the file `main.py` using *Python3*.

The virtual environment of this assignment contains the model of a bird that you will move across the screen using different input devices and transfer functions (see Figure 1). As a representative of isotonic input devices, you will use the standard desktop mouse attached to your PC. The role of the elastic input device will be taken by a space navigator[1], a joystick for navigating in three dimensions.

[1] https://www.3dconnexion.de/spacemouse_compact/

To complete the exercises, modify the provided source code files with respect to the given instructions, compress the directory to a .zip file, and upload it back to Moodle. Please do only insert code between the corresponding `# YOUR CODE - BEGIN` and `# YOUR CODE - END` comments. Additional code outside of the marked areas will not be considered for grading.

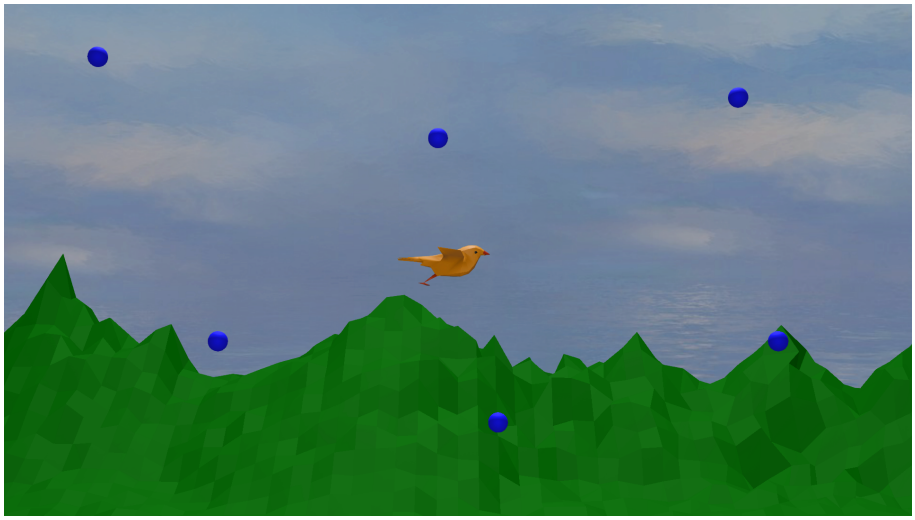

Figure 1: This assignment features a virtual bird model that should be moved across the screen to collect all of the blue spheres with different input devices and transfer functions for its manipulation.

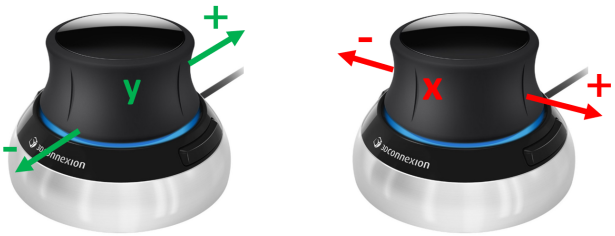

Figure 2: Axes of the space navigator to be used in this assignment.

Exercise 3.0 (no grading)

The class `BirdControls` in the file `BirdControls.py` is responsible for retrieving the inputs of all involved input devices and for using them to manipulate the position of the virtual bird model. You will complete all the coding exercises of this assignment within this class. For applying movement to the bird, you can access its scenegraph node by using the member variable `self.bird_node`. The member variable `self.bird_start_mat` contains the initial transformation matrix of this node. For this assignment, only the x and y components of the bird's translation need to be manipulated (see Figure 2 for the axes on the space navigator).

When launching the assignment code, a position-control mapping of the isotonic mouse is already implemented. Investigate how the inputs are mapped to the bird in the function `apply_isotonic_position_control_input(x_input, y_input)` and try to collect all of the blue spheres in the scene with the bird. Your total time will be printed on the console once the last sphere is collected. You can reset the scene by pressing `1` on the keyboard.

Using a position-control transfer function means that the input values are directly mapped to the position of the virtual object. As a result, moving the mouse in a certain direction results in a direct application of this displacement to the bird. However, the discrete and dimensionless mouse inputs need to be scaled appropriately to achieve a usable result. When the mouse is not moved, the virtual object also stays still.

Exercise 3.1 (2 points)

Implement the function `apply_isotonic_rate_control_mapping(x_input, y_input)` such that the input values of the mouse are applied to the bird using a rate-control transfer function. When running the application, use the key `2` to activate and test your implementation.

Using a rate-control transfer function means that the input values are directly mapped to the velocity of the virtual object. As a result, moving the mouse in a certain direction results in a change of the bird's velocity, which is in turn applied to a change of the bird's position every frame. When the mouse stops moving, the bird keeps moving with the previously defined velocity. To stop the bird, the mouse needs to be moved back to its original position. You can use the member variable `self.velocity` to store the bird's velocity over frames. Make sure to apply a suitable scaling factor to the input values.

Exercise 3.2 (2 points)

Implement the function `apply_isotonic_acceleration_control_mapping(x_input, y_input)` such that the input values of the mouse are applied to the bird using an acceleration-control transfer function. When running the application, use the key `3` to activate and test your implementation.

Using an acceleration-control transfer function means that the input values are directly mapped to the acceleration of the virtual object. As a result, moving the mouse in a certain direction results in a change of the bird's acceleration, which is applied to a change of the bird's velocity every frame, which is in turn applied to a change of the bird's position every frame. When the mouse stops moving, the bird keeps getting faster with respect to the previously defined acceleration. To stop the bird, the mouse needs to be moved in the inverse direction for the same amount of time. You can use the member variables `self.acceleration` and `self.velocity` to store the bird's acceleration and velocity over frames. Make sure to apply a suitable scaling factor to the input values.

Exercise 3.3 (2 points)

Implement the function `apply_elastic_position_control_mapping(x_input, y_input)` such that the input values of the space navigator are applied to the bird using a position-control transfer function. When running the application, use the key `4` to activate and test your implementation.

Since elastic devices snap back to their default position when released, applying a position-control transfer function results in the same effect for the bird. Make sure to apply a suitable scaling factor to the input values.

Exercise 3.4 (2 points)

Implement the function `apply_elastic_rate_control_mapping(x_input, y_input)` such that the input values of the space navigator are applied to the bird using a rate-control transfer function. When running the application, use the key `5` to activate and test your implementation. Make sure to apply a suitable scaling factor to the input values.

Exercise 3.5 (2 points)

Implement the function `apply_elastic_acceleration_control_mapping(x_input, y_input)` such that the input values of the space navigator are applied

to the bird using an acceleration-control transfer function. When running the application, use the key `6` to activate and test your implementation. Make sure to apply a suitable scaling factor to the input values.

Exercise 3.6 (4 points)

Instead of mapping the input values of a device directly to the position, velocity, or acceleration of a virtual object, target-based approaches use the input device for the selection of a target to which the virtual object is then automatically teleported or animated. When pressing the key `7`, you can activate a red cursor that can be moved using the isotonic mouse and a position-control transfer function (see Figure 3).

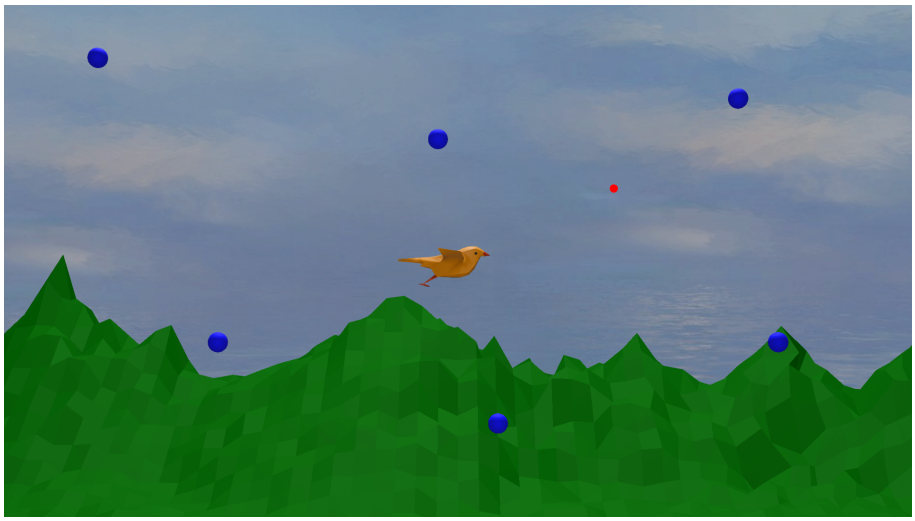

Figure 3: Using an target-based approach, a red virtual cursor defines the target position of the bird. An automatic animation of the bird to the target is triggered when clicking the mouse.

Implement a simple linear animation of the bird towards the specified target when clicking the left mouse button. We prepared the function `sf_mouse_click_changed()`, which is called on every mouse click and sets the member variables `self.animation_start_pos`, `self.animation_target_pos`, and `self.animation_start_time` to the correct values. It is now your task to implement the function `animate_bird()`, which is called every frame and performs the actual animation sequence. In particular, you should ...:

- ... determine the straight-line path to be traversed by the bird.
- ... measure the time since the beginning of the animation.
- ... determine a fraction between 0.0 (beginning of path) and 1.0 (end of path) to specify where the bird should be positioned within this frame. Please ensure that the bird always moves at constant speed, independent of the distance to be traversed.
- ... set the bird model to the computed position.
- ... reset the member variables `self.animation_start_pos`, `self.animation_target_pos`, and `self.animation_start_time` to `None` when the animation is completed.

Exercise 3.7 (1 point)

Implement a slow-in-slow-out animation of the bird, meaning that the bird accelerates and decelerates slowly in the beginning and the end of the animation path, respectively. For this purpose, apply an appropriate mathematical mapping on your computed animation fraction. One example of such a mapping is visualized in Figure 4.

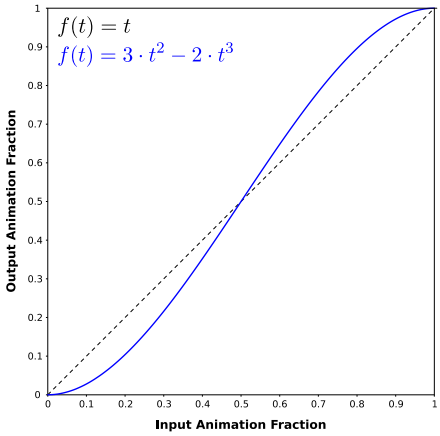

Figure 4: Mathematical illustration of a slow-in-slow-out animation function

Exercise 3.8 (1 point)

Complete the sphere collection task with all available input devices and transfer functions and report on your task completion times. Which combinations were suitable for this task, which combinations were less suitable? Why?