Virtual Reality
Winter Term 2020/2021

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.

You are required to submit this assignment by **10 December 2020, 11:59pm** on Moodle. Furthermore, you will be asked to present and discuss your results on **11 November 2020** to one of the teaching assistants. Please register for an individual time slot with the teaching assistants on Moodle (one slot per group). This assignment contains tasks worth a total of **13** points. Solutions from the asset store are not permitted and will result in zero points for the corresponding exercise.

Getting Started

Download the source code package from the assignment page on Moodle and extract it to your local hard drive and import it as a new Unity project.

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 (Figure 1).

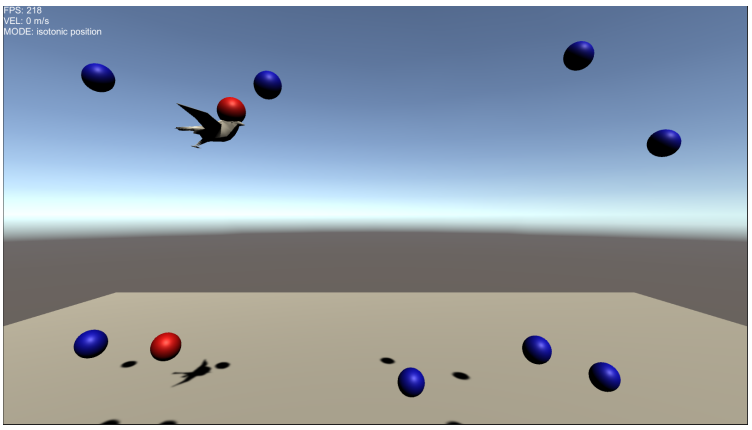

Figure 1: This assignment features a virtual bird model that should be moved across the screen to collect all of the blue spheres with different input devices and transfer functions for its manipulation.

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] (Figure 2) or a gamepad joystick. In case you want to use the space navigator you have to install the `3DxWare 10`[2] driver. You can pick up a spacemouse device at our department. If you want to use a gamepad joystick instead, please set the `spacemouse_flag` to `false` and comment out the `SpaceNavigatorDriver` import at the top of the `Mover` script.

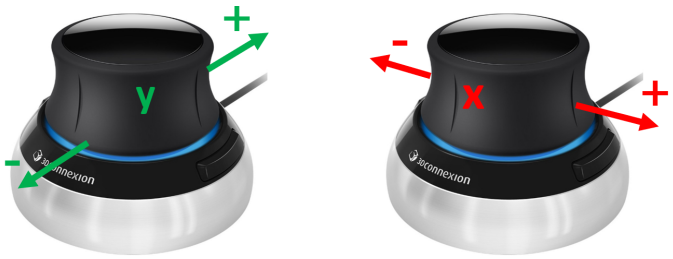

Figure 2: Axes of the space navigator to be used in this assignment.

Regarding transfer function types you have to implement position-, rate-, and acceleration-control for both input types, which results in six combinations in total (Table 1). You can toggle between the combinations via the number keys (1-6). In the HUD in the upper left corner the actual mode is displayed. The actual cursor/bird velocity is also displayed here, which might be helpful to debug your solutions.

Key	Input Type / Transfer Function
1	Isotonic - Position Control
2	Isotonic - Rate Control
3	Isotonic - Acceleration Control
4	Elastic - Position Control
5	Elastic - Rate Control
6	Elastic - Acceleration Control

Table 1: Six combinations of input types and transfer functions.

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.

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

Exercise 3.1 (no grading)

The script `Mover` attached to the Game Object `Cursor` is responsible for retrieving the inputs of all involved input devices and for using them to manipulate the position of the cursor aka the bird model. You will complete all the coding exercises of this assignment within this script. Only the x and y components of the bird's position 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 in the function `IsotonicPosition(float X, float Y)`.

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 3.2 (2 points)

Implement the function `IsotonicRate(float X, float Y)` 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 start position. You can add a further variable to the class to store the bird's velocity over frames. Make sure to apply a suitable scaling factor to the input values.

Compensate for varying frame rates, such that the bird moves with the same velocity independent of the actual application frame rate.

Exercise 3.3 (2 points)

Implement the function `IsotonicAcceleration(float X, float Y)` 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 introduce additional variables to store the bird's acceleration and velocity over frames. Make sure to apply a suitable scaling factor to the input values.

Again, compensate for varying frame rates, such that the bird moves with the same velocity independent of the actual application frame rate.

Exercise 3.4 (2 points)

Implement the function `ElasticPosition(float X, float Y)` 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 (self-centering), 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 to rach all targets.

Exercise 3.5 (2 points)

Implement the function `ElasticRate(float X, float Y)` 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 and ensure frame-rate independent movement velocity.

Exercise 3.6 (2 points)

Implement the function `ElasticAcceleration(float X, float Y)` 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 and ensure frame-rate independent behavior.

Exercise 3.7 (3 point)

Which combinations were suitable for this task, which combinations were less suitable? Why? Come up with potential use cases for the six different combinations. Think in the context of object manipulation and viewpoint navigation. You can enter your ideas at the very end of the `Mover` script.