

University of Applied Sciences Dresden
Faculty Information Technology / Mathematics
Prof. W. Oertel

VRIE

Working Environment, Static and Dynamic Objects

Practice 01

1. Working Environment

Start one of the VRML scripts in the directory Examples/VR-02 within the VRML viewer. Make you acquainted with the working environment of the VRML viewer. Navigate through the scene by use of the mouse and the soft buttons on the window border. Try to find and use all six spatial degrees of freedom. Use also the menu bound at the right mouse button for changing parameters.

2. Static Objects

Start single VRML scripts in the directories Examples/VR-03 – VR-05 within the VRML viewer. Navigate through the represented scenes. Open the VRML files and try to understand the essential parts of the stored scripts. Change single parameters in the script, save the files, and start it again within the VRML viewer. For example, you can change colours, sizes, positions, or types of objects. You can also add or remove objects.

3. Dynamic Objects

Start single VRML scripts in the directories Examples/VR-06 – VR-08 within the VRML viewer. Navigate through the represented scene. Watch the temporal behaviour of the objects. Interact with selected objects of the scene. Open the VRML files and try to understand the essential parts of the stored scripts. Change single parameters in the script, save the files, and start it again within the VRML viewer. For example, you can change, add, or remove times, sensors, links, routes, and java scripts.

University of Applied Sciences Dresden
Faculty Information Technology / Mathematics
Prof. W. Oertel

VRML

Prototypes and Intelligent Models

Practice 02

1. Prototype Use

Start the example VRML scripts in the directory Examples/VR-09 within the VRML viewer. Navigate through and interact with the scene. Use the editor to change the scene. Have a look at the source code of the scene.

2. Spatiotemporal Models

Create your own simple spatiotemporal model of a house by adding objects using predefined prototypes.

3. Intelligent Behaviours

Create a simple intelligent behaviour of your house by adding relations between parameters of the predefined prototypes.

4. Prototype Definition

Define a new prototype with respective parameters and integrate it in the already defined scene.

University of Applied Sciences Dresden
Faculty Information Technology / Mathematics
Prof. W. Oertel

LISP Programming Practice 03

1. Working Environment

Start the Lisp system and try to work with the user interface. Type in some variable definitions, initialise them, and access after this the values again. Define in the same way a simple function and call it with the required parameters.

2. External Editor and Functions

Open an external text editor. Define there a program that defines the faculty function. Load this program to the Lisp editor and start it. Call the respective function with different parameters. Try to find the maximum number for the faculty function that can still be handled by the system.

3. Lists and Operations

Define in the external editor a program that operates on list structures. Define, therefore, in one variable a list containing 5 components of a building with their parameters. Define, for instance, a wall with the attributes name, size, position, material, and colour. Or define a door with the same attributes. Write a function that adds a new component to the existing list and a function that removes an element from this list. Define a last function that provides the different values of a certain attribute of all components of the building.

University of Applied Sciences Dresden
Faculty Information Technology / Mathematics
Prof. W. Oertel

LISP

Knowledge Representation and Problem Solving

Practice 04

1. Use of Predefined Knowledge Bases and Problem Solvers

Start the Lisp system and load one of the 5 example programs in the example directory. Start the program by calling the respective main function. Use, if possible, different parameters in the function call.

2. Change of the Knowledge Base

Change the knowledge base of the loaded example program by adding new elements or updating existing elements. Test the changed behaviour. Try to understand the operation of the problem solving component.

2. Continuation

Proceed in the same way with the other 4 example programs.