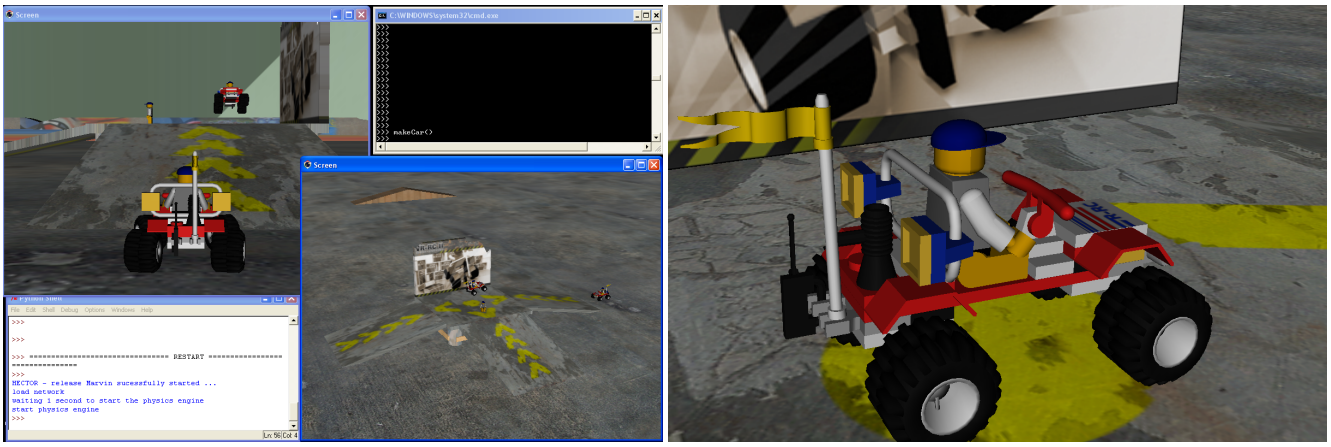
HECTOR – Scripting-Based VR System Design – Colorplate

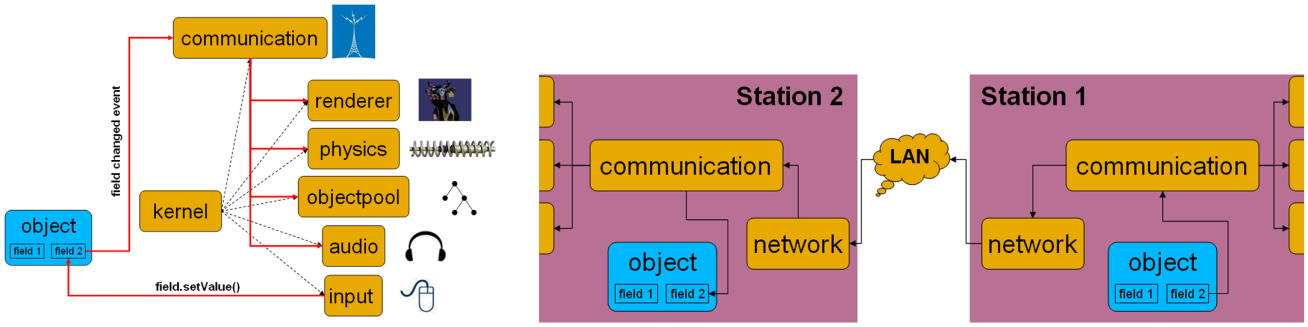
Gordon Wetzstein[1,3] Moritz Göllner[2,3] Stephan Beck[3] Felix Weiszig[3] Sebastian Derkau[3] Jan P. Springer[3] Bernd Fröhlich[3]

[1] University of British Columbia [2] Ceetron GmbH [3] Bauhaus-Universität Weimar



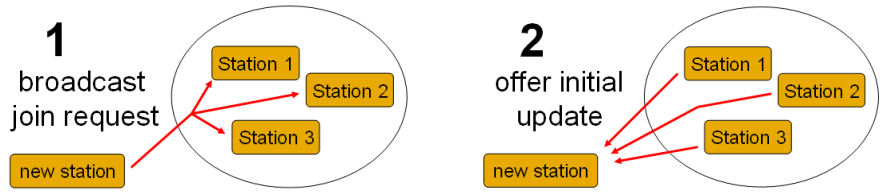
(a) (b)

Figure 1: (a) Two HECTOR stations running a consistent physically-based buggy racer. Although both stations run on the same computer, they are only connected via a Spread server, which allows the application to be executed over the network in the same way. (b) A close-up view of one of the buggy cars.

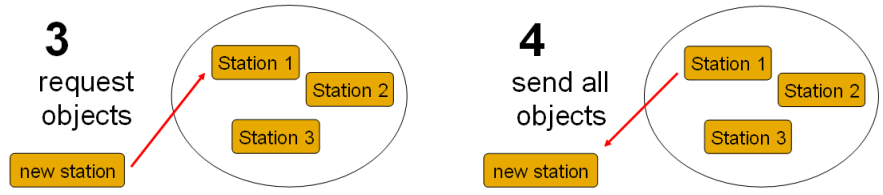


(a) (b)

Figure 2: (a) Brown rectangles represent HECTOR modules – threads that implement different functionality. They are loaded and unloaded by a micro kernel upon request by objects in a virtual environment. Attribute changes, for instance by input devices or physics simulation, are multicast as events. (b) In order to guarantee a consistent world events are serialized and reinitialized on a different station as if they were created locally on that station.



(a) Join. **(b)** Update.



(c) Request. **(d)** Send.

Figure 3: The bootstrap mechanism is a 4-way handshake ((a) to (d)) which updates a new station with all objects in an existing virtual environment and its hierarchical structure. All initialization events are distributed via reliable multicast. After the joining station received all objects in the scene graph its object pool recreates the hierarchy, thus updating other modules (e. g. renderer, physics engine, audio manager) with all necessary information.