Technical Disclosure Commons

Defensive Publications Series

October 02, 2017

# Distributed Physics Simulation with Stateless Server

Robert Jagnow

Daniel Citron

Chun Yat Franki Li



# Distributed Physics Simulation with Stateless Server

**Abstract:**

Physics-based simulations of objects displayed in a scene are distributed amongst client systems supporting multiple users interacting with the scene. For each client system, ownership of one or more objects displayed in the scene is established. For the objects owned by a client system, and based on interactions within the scene, the client system performs physics-based simulations. Resultant changes in position and kinematics of the owned objects are communicated from the client system to other client systems via a stateless server.

**Keywords:**

Virtual reality, physical object, kinematic object, stateless server, physics simulation, physical body, rigid body

**Background:**

Virtual reality (VR) environments rely on display, tracking, and VR-content systems. Through these systems, realistic images, sounds, and sometimes other sensations simulate a user's physical presence in an artificial environment. Each of these three systems are illustrated below in Fig. 1.

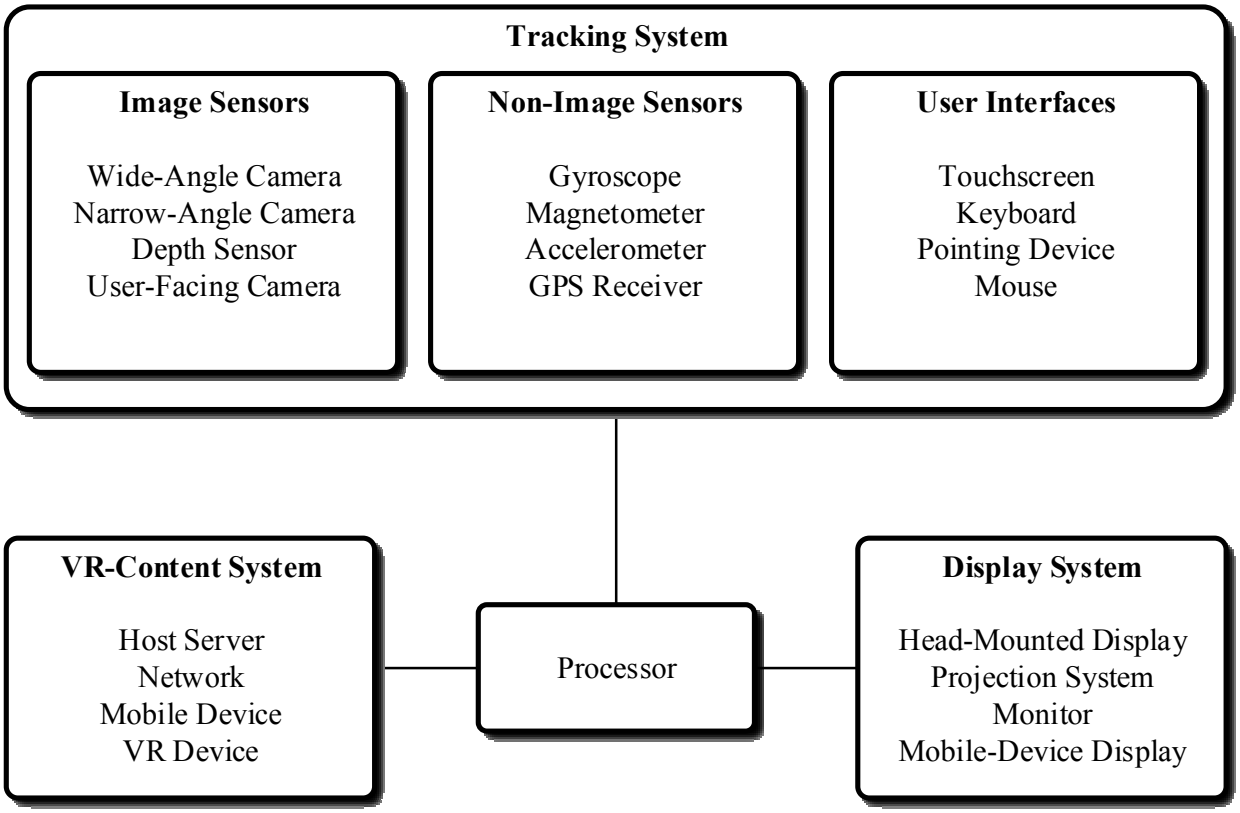


**Fig. 1**

The systems described in Fig. 1 may be implemented in one or more of various computing devices that can support VR applications, such as servers, desktop computers, VR goggles, computing spectacles, laptops, or mobile devices. These devices include a processor that can manage, control, and coordinate operations of the display, tracking, and VR-content systems. The devices also include memory and interfaces. These interfaces connect the memory with the systems using various buses and other connection methods as appropriate.

The display system enables a user to "look around" within the virtual world. The display system can include a head-mounted display, a projection system within a virtual-reality room, a monitor, or a mobile device's display, either held by a user or placed in a head-mounted device.

The VR-content system provides content that defines the VR environment, such as images and sounds. The VR-content system provides the content using a host server, a network-based device, a mobile device, or a dedicated virtual reality device, to name a few.

The tracking system enables the user to interact with and navigate through the VR environment, using sensors and user interfaces. The sensors may include image sensors such as a wide-angle camera, a narrow-angle camera, a user-facing camera, and a depth sensor. Non-image sensors may also be used, including gyroscopes, magnetometers, accelerometers, GPS sensors, retina/pupil detectors, pressure sensors, biometric sensors, temperature sensors, humidity sensors, optical or radio-frequency sensors that track the user's location or movement (*e.g.*, user's fingers, arms, or body), and ambient light sensors. The sensors can be used to create and maintain virtual environments, integrate "real world" features into the virtual environment, properly orient virtual objects (including those that represent real objects, such as a mouse or pointing device) in the virtual environment, and account for the user's body position and motion.

The user interfaces may be integrated with or connected to the computing device and enable the user to interact with the VR environment. The user interfaces may include a touchscreen, a keyboard, a pointing device, a mouse or trackball device, a joystick or other game controller, a camera, a microphone, or an audio device with user controls. The user interfaces allow a user to interact with the virtual environment by performing an action, which causes a corresponding action in the VR environment (*e.g.,* raising an arm, walking, or speaking).

The tracking system may also include output devices that provide visual, audio, or tactile feedback to the user (*e.g.*, vibration motors or coils, piezoelectric devices, electrostatic devices, LEDs, strobes, and speakers). For example, output devices may provide feedback in the form of blinking and/or flashing lights or strobes, audible alarms or other sounds, songs or other audio

files, increased or decreased resistance of a control on a user interface device, or vibration of a physical component, such as a head-mounted display, a pointing device, or another user interface device.

Fig. 1 illustrates the display, tracking, and VR-content systems as disparate entities in part to show the communications between them, though they may be integrated, *e.g.*, a smartphone mounted in VR goggles, or operate separately in communication with other systems. These communications can be internal, wireless, or wired. Through these illustrated systems, a user can be immersed in a VR environment. While these illustrated systems are described in the VR context, they can be used, in whole or in part, to augment the physical world. This augmentation, called "augmented reality" or AR, includes audio, video, or images that overlay or are presented in combination with the real world or images of the real world. Examples include visual or audio overlays to computing spectacles (*e.g.,* some real world-VR world video games or information overlays to a real-time image on a mobile device) or an automobile's windshield (*e.g.,* a heads-up display) to name just a few possibilities.

In a scene that is displayed as part of a VR environment of Fig. 1, it may be desirable for objects within the scene to behave realistically from a physics-based perspective. This is often accomplished by having a single dedicated host server (VR-content system) compute physics-based simulations of the objects within the scene based on either a user of a client system interacting with the objects or based on the objects within the scene interacting amongst themselves.

Using the dedicated host server to compute physics-based simulations for scenes of a VR environment presents several challenges. For example, expenses associated with the dedicated host server can manifest in terms of hardware, power, computational load, or infrastructure

necessary to support communication between the multiple client systems and the dedicated host server. If the dedicated host server is not part of a local area network (LAN) shared by the multiple client systems, communication latencies can impact virtual reality environment authenticity. And, if the dedicated host server has performance or reliability issues, all client systems and the entire VR environment could be impacted by a failure experienced by the dedicated host server.

**Description:**

To alleviate these challenges techniques are described that utilize multiple client devices participating in a scene of a VR environment to compute physics-based simulation of objects in conjunction with a stateless server. In a displayed scene, multiple objects may be present and may interact with one another. If order for the scene to be as authentic a replica of the real world as possible, physics-based simulations must be performed for the multiple objects and the interactions with, or amongst, the multiple objects in the scene. For example, if a scene contains two balls colliding, the mass, velocity, and coefficient of restitution for each ball would be needed to accurately represent how each ball would respond to the collision. Another example could be a scene where a virtual arm of a person reaches into the scene to pet a dog. Normally petting the dog would require the virtual arm apply a virtual force of a few pounds to the dog, and the dog's head would move accordingly. Physics-based simulations applied to the interactions of the multiple objects in the scene will, if computed accurately, result in the multiple objects reacting with "real" kinematic responses to applied forces or changes in energy. A virtual reality application, in response to accurately computed physical simulations of the applied forces or changes in energy associated to the multiple objects, replicates the physical objects in the scene to

appear real not only in terms of object position, but also in terms of kinematic variables such as velocity, acceleration, or rotation.

In a scene of a VR environment, the computing power necessary to quickly compute a physics-based simulation using variables necessary for accurate representations of all kinematic responses of all objects interacting in the scene can be inordinate. Consider a scene as viewed through a display of a client system (*e.g.,* a desktop computer type of VR-content system of Fig. 1) as displayed in Figure 2:

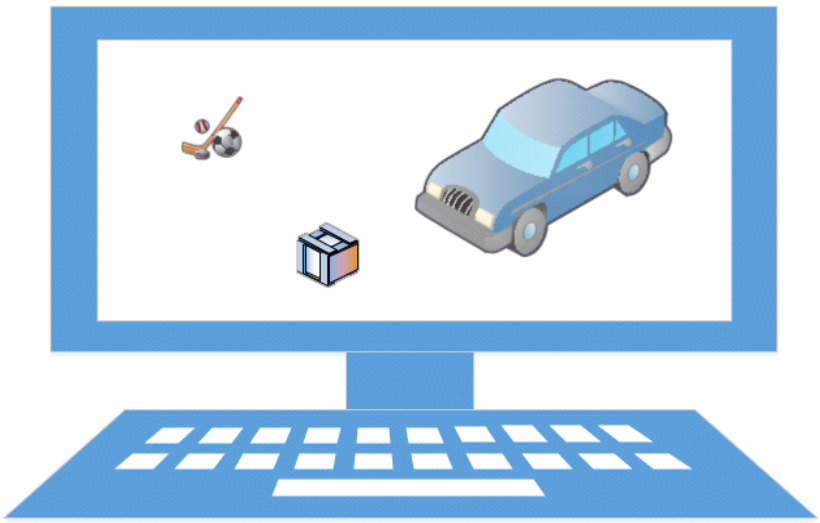

**Fig. 2**

In Fig. 2, multiple objects of a scene are presented through a display system. Each object in the environment has its own physical properties defining it to be a full physics object, including, as a simplistic example, an object mass and an object coefficient of restitution that is useful to compute kinetic energy losses during a collision with another object. At a particular point in time, additional variables, such as a velocity or acceleration in one direction or a rotation about a certain

axis, indicate each physical object's position and kinematic state. When the scene changes, computations must be performed taking into account the aforementioned properties and variables, as well as changes to the aforementioned properties and variables, in order to replicate each object in an authentic real world fashion. As the number of VR-content systems participating in the scene increases or the number of physical objects in the displayed scene increases, the number of computations required to maintain an accurate representation of the scene increases substantially.

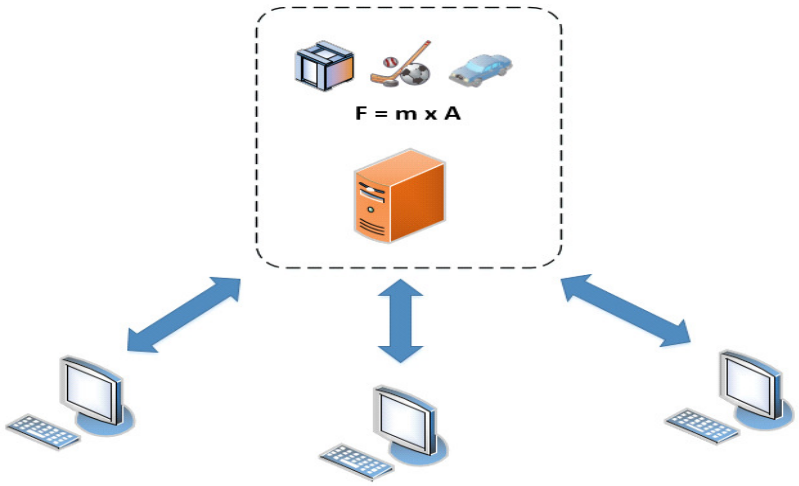


**Fig. 3**

Fig. 3 represents the conventional model of a dedicated host server interacting with multiple client systems participating in a scene (each, in whole or in part, a VR-content system as illustrated in Fig. 1). In Fig. 3, the dedicated host server has authority over all physical objects and performs all physics-based computations (as represented by the simplified equation F = m x A) necessary to simulate and maintain replication of all of the physical objects in the scene. At regular intervals, the dedicated host server performs computations in order to simulate position

and kinematic states for all of the objects in the scene and then sends corrections to each client system. As discussed above, the expenses for the dedicated host server can manifest in terms of hardware, power, computational load, or infrastructure necessary to support communication between the multiple client systems and the dedicated host server. Performance issues may also develop in terms of latency or dedicated host server reliability.

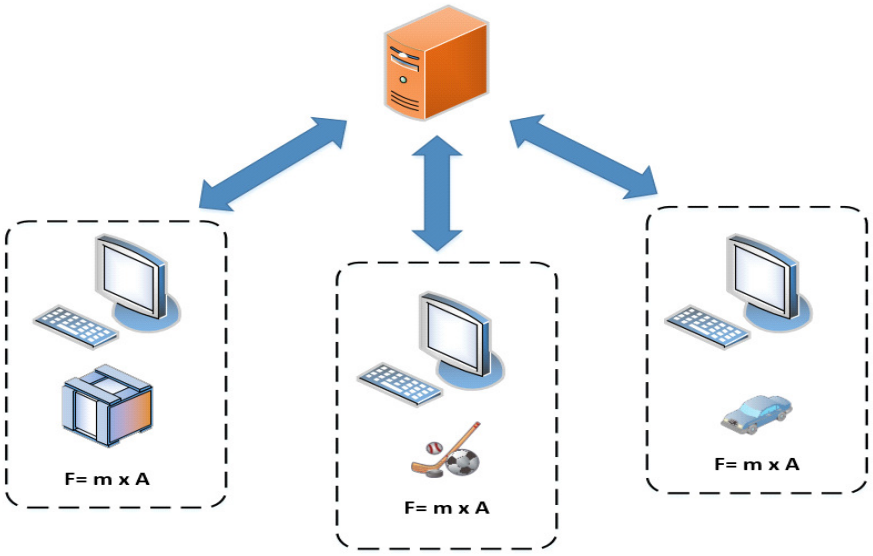


**Fig. 4**

Fig. 4 represents of a model of a stateless server interacting with multiple client systems participating in a scene of a VR environment, thus each acting as part of the VR-content system of Fig. 1. In Fig. 4, computations supporting physics-based simulations are distributed across the multiple client systems participating in the scene. Each of the multiple client systems may assume authority over one or more physical objects displayed in a scene in order to perform computations for physics-based simulations related to the assumed physical objects. As an output of respective physics-based simulations, each client system sends, to other client systems participating in the

scene via the stateless server, updates indicating changes to the positions and the kinematic states of objects over which it has authority. For objects over which the client system does not have authority, objects are represented as rigid kinematic bodies. In this model, where the stateless server interacts with multiple client systems performing physics-based simulations in a distributed fashion, the stateless server maintains a low computations load, acting instead as a conduit enabling the multiple client systems to exchange physical object location and kinematic information.

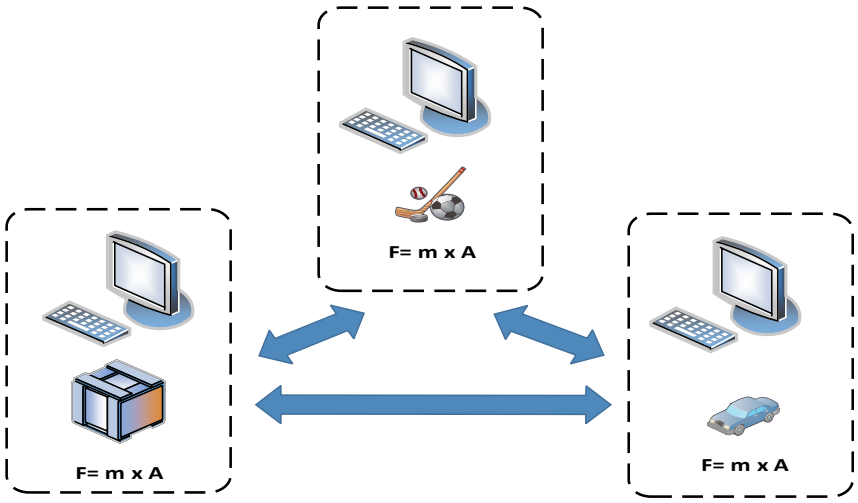


**Fig. 5**

Fig. 5 represents another model afforded by the use of distributed physics-based simulations. In Fig. 5, computations supporting physics-based simulations are distributed across the multiple client systems participating a scene while communications between the client systems are performed in a peer-to-peer fashion. Also, as illustrated in Fig. 5, a dedicated host server

performing physics-based simulations or even a host server acting in a stateless fashion and routing communications simply is not present.

An authority transfer system embedded in a virtual reality application of the aforementioned models implements rules to govern authority of the multiple client systems over the physical objects. Rules for assuming authority over a physical object, as put into practice by the authority transfer system, are based on any number of factors, including, for example, an order in which the multiple client systems interact with the physical objects. In this instance, a first client system to interact with the physical object or spawn the physical object in the scene assumes authority over the physical object. The authority transfer system also has the option of automating transfer of authority from the client system to another client system if a user of the system discontinues participation in the scene. Additional rules may be request/confirmation-based rules, security-based rules, or rules invoked in response to performance needs resulting from either the stateless server environment or the peer-to-peer environment that has no host server. Performance needs may include a client system failing to execute computations supporting a physics-based simulation of an object, a latency within a network supporting communication amongst the client systems, or computing power necessary for a particular client system to compute physics-based simulations of a particular object.

Although a distribution of physics-based simulations directed to basic Newtonian physics (e.g., the relationship between a physical body and the forces acting upon the physical body) has been described, other physics-based simulations of physical objects presented as part of a VR environment may be distributed as well. For example, sounds that might be made by an object in the VR environment (consider the object being a ball bouncing and the sound of the bounce off a floor) may be distributed across multiple client systems. Another example of distributable physics-

based simulations is the reflection/illumination of objects displayed as part of the VR environment (*e.g.*, the reflection of the objects in a scene such as a pond, mirror, or other object).