August 21, 2019

XR Extensions

Anonymous Anonymous

XR Extensions

Abstract

A system for exposing a webXR API (application program interface) to a web developer for addition of new features is disclosed. The webXR API acts as an interface between an XR (extended reality) application and a webXR runtime. The webXR runtime is a software, which implements the webXR API. The webXR API is a layered API having a number of API layers. The webXR API is exposed to the web developer for adding a set of new webXR functions or modifying a behavior of a set of existing webXR functions. The web developer inserts new API layers for adding the set of new webXR functions. The web developer intercepts the set of existing functions from the API layers for modifying their behavior. This way, the new features are introduced to the webXR API. The webXR API exposes the new features in the form of XR extensions. Once the XR extensions are successfully added, these are enabled by creating an XR instance. The XR instance is an object that allows an XR application to communicate with the webXR runtime.

Problem statement

In absence of upgrades in functionalities of the XR application, a user might sometimes miss out on a user-friendly, engaging and a dynamic experience. In such a scenario, there is no scope for the user to experience the new features, that can only be made available in a reprogrammable platform.

The present disclosure proposes a novel solution to solve the above problem.

System and working

The present disclosure describes a system that exposes a webXR API (application program interface) to a web developer for addition of new features. The webXR API is an interface between an XR (extended reality) application and a webXR runtime. XR (extended reality) refers to a continuum of real-and-virtual combined environment that is inclusive of the technologies associated with virtual reality (VR), augmented reality (AR) and mixed reality (MR). In the XR, the webXR runtime is a software that implements the webXR API. A plurality of webXR runtimes may be installed on the system but only one of the plurality of webXR runtimes is active at any given time. The webXR API is a layered API having a number of API layers. It is an extensible API such that its functionalities can be extended through addition of the new features. The webXR API is exposed to the web developer for adding a set of new webXR functions or modifying a behavior of a set of existing webXR functions. The web developer may also insert new API

layers between the XR application and the webXR runtime. The web developer creates the new API layers using the webXR API for adding the set of new webXR functions. The web developer adds additional functionalities by intercepting the set of existing webXR functions from an existing layer in the webXR API. This is done by the web developer for modifying the behavior of the set of existing webXR functions in the API layers. This way, by adding the set of new webXR functions or the additional functionalities to the set of existing webXR functions, the new features are introduced to the webXR API. The webXR API exposes the new features in the form of XR extensions. Once the XR extensions are successfully added, these must be enabled by the XR application before the new features are made available.

The XR extensions are enabled by means of an XR instance explained in subsequent sections.

The XR instance, as defined below, is an object that allows the XR application to communicate with the webXR runtime.

XR_DEFINE_HANDLE(XrInstance)

The XR application accomplishes this communication by calling an xrCreateInstance function. The xrCreateInstance function is defined as:

XrResult xrCreateInstance (Const XrInstanceCreateInfo info_Creation, XrInstance resulting_Instance);

The Info_Creation is an instance of XrInstanceCreateInfo structure, which controls creation of the XR instance. The resulting_Instance points to an Xrinstance handle in which a resulting instance (i.e. the XR instance) is returned. The info_Creation calls the XrInstanceCreateInfo structure, which specifies the API layers and the XR extensions that are to be enabled. The XrInstanceCreateInfo structure is defined as:

Typedef struct XrInstanceCreateInfo {
XrStructureType type; //type is the XrStructureType of this structure.
const void next; //next is NULL*
XrInstanceCreateFlags create_Flags; //create_Flags is a bitmask of XrInstanceCreateFlags.
XrApplicationInfo application_Info; //application_Info is an instance of XrApplicationInfo.
uint32_t Number_enabled_API_Layer; //Number_enabled_API_Layer is number of the API layers to be enabled.

const char const* Names_enabled_API_Layer; //Names_enabled_API_Layer is a pointer to an array of Number_enabled_API_Layer strings containing names of the API layers to be enabled for the XR instance.*
uint32_t Number_enabled_Extension; //Number_enabled_Extension is number of XR extensions to be enabled.
const char const* Names_enabled_Extension; //Names_enabled_Extension is a pointer to an array of Number_enabled_Extension strings containing names of the XR extensions to be enabled.*
} XrInstanceCreateInfo;

This way, the xrCreateInstance function creates the XR instance, then enables and initializes the API layers and the XR extensions by utilizing the XrInstanceCreateInfo structure.

Additional embodiments

In an additional embodiment, the system also enables the web developer to control the API layers exposed to him/her. For instance, there is a default frame rate set for each device type (Quest, Go, etc.). But the web developer may set a different frame rate. Similarly, there might be a default level of foveated rendering set for content items for each device type (Quest, Go, etc.). But the web developer may set a different amount of the foveated rendering for the content items.

Conclusion

Hardware that enables XR (extended reality) applications are now broadly available to consumers, offering an immersive computing platform with both new opportunities and challenges. The ability to interact directly with immersive hardware is critical for ensuring that the web is well equipped to operate as a first-class citizen in this environment. Immersive computing introduces strict requirements for high-precision, low-latency communication in order to deliver an acceptable experience. Also, for the immersive computing platform to be compatible with the evolving hardware, it needs to be dynamic in a manner that changes, or upgrades are possible in the functionalities of the XR application. With the system in the present disclosure, there is a scope for the web developer to extend the functionalities of the XR application. The WebXR API provides interfaces necessary for enabling the web developers to build user friendly, comfortable, and safe immersive applications on the web for the immersive environment.