13.21. XR_EXT_overlay

Name String

`XR_EXT_overlay`

Extension Type

Instance extension

Registered Extension Number

34

Revision

1

Extension and Version Dependencies

- Requires OpenXR 1.0

Last Modified Date

2018-11-05

IP Status

No known IP claims.

Contributors

Mark Young, LunarG
Jules Blok, Epic Jared Cheshier, Pluto VR Nick Whiting, Epic

Overview

Application developers may desire to implement an OpenXR application that renders content on top of another OpenXR application. These additional applications will execute in a separate process, create a separate session, generate separate content, but want the OpenXR runtime to composite their content on top of the main OpenXR application. Examples of these applications might include:

- A debug environment outputting additional content
- A Store application that hovers to one side of the user's view
- A interactive HUD designed to expose additional chat features

This extension introduces the concept of "Overlay Sessions" in order to expose this usage model.

This extension allows: * An application to identify when the current sessions composition layers will be applied during composition * The ability for an overlay session to get information about what is going on with the main application

In order to enable the functionality of this extension, you **must** pass the name of the extension into xrCreateInstancevia the XrInstanceCreateInfo `enabledExtensionNames` parameter as indicated in the [extension] section.

When you are ready to create an overlay session, you will need to create a XrSessionCreateInfoOverlayEXT structure and passing it into the xrCreateSession via the XrSessionCreateInfo structure's `next` parameter.

13.21.1. Overlay Session Layer Placement

Since one or more sessions may be active at the same time, this extension provides the ability for the application to identify when the frames of the current session will be composited into the final frame.

The `XrSessionCreateInfoOverlayEXT sessionLayersPlacement` parameter provides information on when the sessions composition layers should be applied to the final composition frame. The larger the value passed into `sessionLayersPlacement`, the closer to the front this session's composition layers will appear (relative to other overlay session's composition layers). The smaller the value of `sessionLayersPlacement`, the further to the back this session's composition's layers will appear. The main session's composition layers will always be composited first, resulting in any overlay content being composited on top of the main application's content.

If `sessionLayersPlacement` is 0, then the runtime will always attempt to composite that session's composition layers first.
If `sessionLayersPlacement` is UINT32_MAX, then the runtime will always attempt to composite that session's composition layers last. If two or more overlay sessions are created with the same `sessionLayersPlacement` value, then the newer session's will be treated as if they had a slightly higher value of `sessionLayersPlacement` than the previous sessions with the same value. This should result in the newest overlay session being composited closer to the user than the older session.

The following image hopefully will provide any further clarification you need:

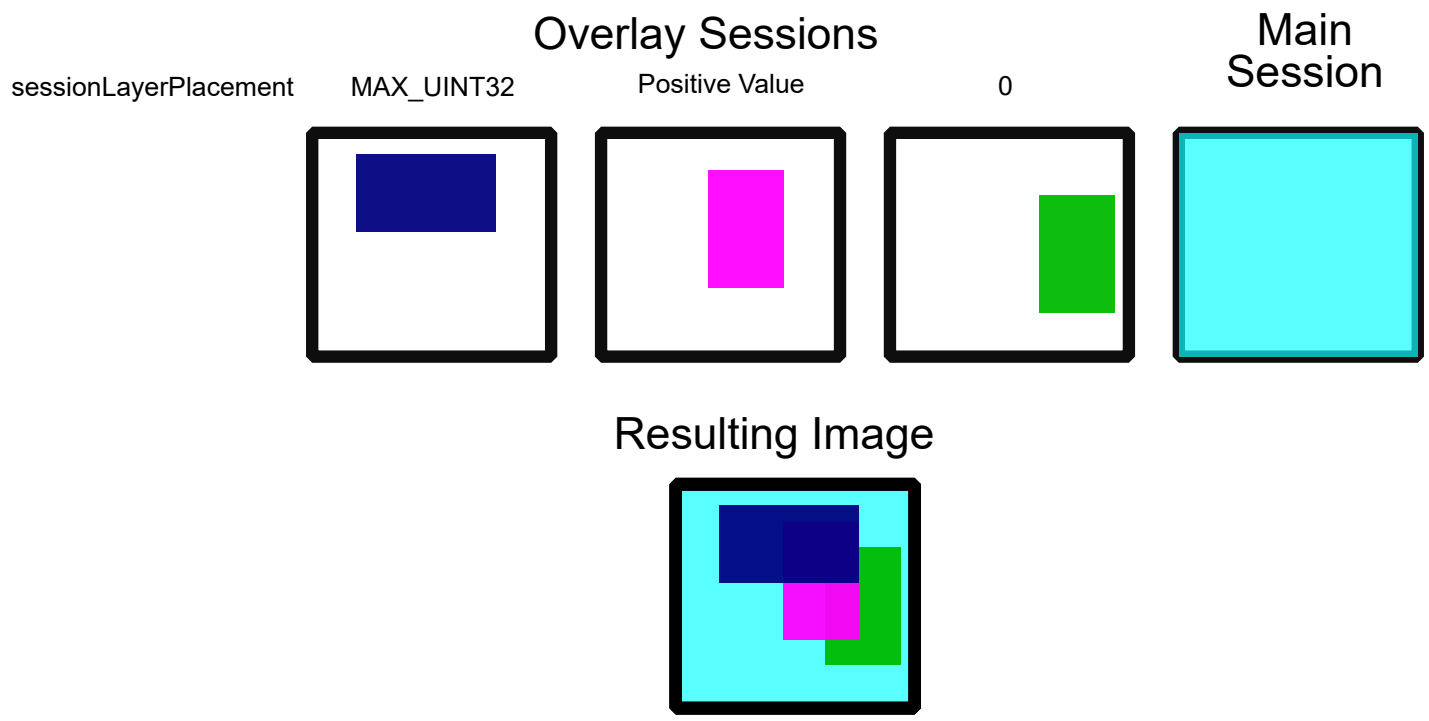

Figure 4. Overlay Composition Order

13.21.2. Main Session Behavior Event

Since an overlay session's intends to work in harmony with a main session, some information needs to be provided from that main session to the overlay session.

The XrEventDataMainSessionVisibilityChangedEXT event structure provides information on the visibility of the main session as well as some additional flags which can be used to adjust overlay behavior.

If XR_KHR_extra_layer_info_depth is enabled in the main session, then XrEventDataMainSessionVisibilityChangedEXT `flags` **should** contain the value: `XR_MAIN_SESSION_BIT_ENABLED_EXTRA_LAYER_INFO_DEPTH_EXT` . If the overlay session also enables XR_KHR_extra_layer_info_depth, then when the both sessions are visible, the runtime can integrate their projection layer content together using depth information as described in the extension. However, if either the main session or the overlay do not enable the extension, then composition behavior will continue as if neither one enabled the extension.

13.21.3. Modifications to the OpenXR Specification

When this extension is enabled, certain core behaviors defined in the OpenXR specification must change as defined below:

Modifications to Composition

The Compositing section description of the composition process will be changed if this extension is enabled. If this extension is enabled, and there is only one active session, then there is no change. However, if this extension is enabled, and there are multiple active sessions, then the composition will occur in order based on the overlay session's XrSessionCreateInfoOverlayEXT:: `sessionLayersPlacement` value as described in the table below:

Table 3. Overlay Session Composition Order

Session Type	XrSessionCreateInfoOverlayEXT:: `sessionLayersPlacement`	Composited
Overlay Session	UINT32_MAX	*Composited last, appears in front of all other XrSessions*
Overlay Session	<Positive value>	
Overlay Session	0	
Non-overlay Session	N/A	*Composited first, appears behind all other XrSessions*

The above change only applies to when a session's composition layers are applied to the resulting image. The order in which composition layers are handled internal to a session does not change. However, once the sessions have been properly ordered, the runtime should behave as if all the composition layers have been placed into a single list (maintaining the separation of viewport images) and treat them as if they were from one

original session. From this point forward, the composition behavior of the resulting composition layers is the same whether or not this extension is enabled.

If the overlay session is created as part of an XrInstance which has enabled the XR_KHR_extra_layer_info_depthextension, and a XrExtraLayerInfoDepthKHR structure has been provided to one or more composition layers, then it intends for those layers to be composited into the final image using that depth information. This composition occurs as defined in the XR_KHR_extra_layer_info_depth extension. However, this is only possible if the main session has provided depth buffer information as part of its swapchain. In the event that a main session does not provide depth buffer information as part of its swapchain, then overlay application's composition layers containing depth information will be composited as if they did not contain that information.

Modifications to xrEndFrame Behavior

Compositor Layer Behavior currently states that if xrEndFrame is called with 0 layers, then the runtime should clear the VR display.

If this extension is enabled, the above statement is now only true if the session is not an overlay session. If the session is an overlay session, and it provides 0 layers in the call to `xrEndFrame`, then the runtime will just ignore the overlay session for the current frame.

Modifications to Input Synchronization

If a runtime supports this extension, it **must** separate input tracking on a per-session basis. This means that reading the input from one active session does not disturb the input information that can be read by another active session. This may require duplicating events to more than one session.

New Object Types

None

New Flag Types

```
XrOverlayMainSessionFlagsEXT
```

New Enum Constants

XrStructureType enumeration is extended with:

```
XR_TYPE_SESSION_CREATE_INFO_OVERLAY_EXT
XR_TYPE_EVENT_DATA_MAIN_SESSION_VISIBILITY_CHANGED_EXT
```

New Enums

```
XR_MAIN_SESSION_BIT_ENABLED_EXTRA_LAYER_INFO_DEPTH_EXT
```

New Structures

```
typedef struct XrSessionCreateInfoOverlayEXT {
    XrStructureType             type;
    const void* XR_MAY_ALIAS    next;
    XrBool32                    overlaySession;
    uint32_t                    sessionLayersPlacement;
} XrSessionCreateInfoOverlayEXT;
```

Member Descriptions

- `type` is the type of this structure.
- `next` is `NULL` or a pointer to an extension-specific structure.
- `overlaySession` is an XrBool32 value that indicates if the session is an overlay (XR_TRUE) or not (XR_FALSE).
- `sessionLayersPlacement` is a value indicating the desired placement of the session's composition layers in terms of other sessions.

Valid Usage (Implicit)

- `type` **must** be `XR_TYPE_SESSION_CREATE_INFO_OVERLAY_EXT`
- `next` **must** be `NULL`

Receiving the XrEventDataMainSessionVisibilityChangedEXT event structure indicates that the main session has gained or lost visibility. This can occur in many cases, one typical example is when a user switches from one OpenXR application to another.
See XrEventDataSessionVisibilityChanged for more information on the standard behavior. This structure contains additional information on the main session including `flags` which indicate additional state information of the main session. Currently, the only flag value supplied is `XR_MAIN_SESSION_BIT_ENABLED_EXTRA_LAYER_INFO_DEPTH_EXT` which indicates if the main session has enabled the XR_KHR_extra_layer_info_depth extension.

```
typedef struct XrEventDataMainSessionVisibilityChangedEXT {
    XrStructureType                 type;
    const void* XR_MAY_ALIAS        next;
    XrBool32                        visible;
    XrOverlayMainSessionFlagsEXT    flags;
} XrEventDataMainSessionVisibilityChangedEXT;
```

Member Descriptions

- `type` is `XR_TYPE_EVENT_DATA_SESSION_VISIBILITY_CHANGED`.
- `next` is `NULL` or a pointer to an extension-specific structure.
- `visible` is an `XrBool32` which indicates if `session` is now visibility or does not.
- `flags` is 0 or one or more `XrOverlayMainSessionFlagBitsEXT` which indicates various state information for the main session.

Valid Usage (Implicit)

- `type` **must** be `XR_TYPE_EVENT_DATA_MAIN_SESSION_VISIBILITY_CHANGED_EXT`

- `next` **must** be `NULL`
- `flags` **must** be a valid combination of XrOverlayMainSessionFlagBitsEXT values
- `flags` **must** not be `0`

New Functions

None

New Function Pointers

None

Issues

None

Version History

- Revision 1, 2018-11-05 (Mark Young)
  - Initial draft