OpenXR

Und die Monado OpenXR runtime

Christoph Haag

Software Engineer
bei Collabora

COLLABORA

- Wozu VR Runtimes?
- OpenXR Funktionsweise
- Monado

COLLABORA

Wozu VR Runtimes?

Teaser

Presentation: Moving KDE to another Reality

AR and VR did not only introduce a new class of output devices, but with tracked controllers and hands also the requirement for a new set of user interactions. This talk investigates solutions in existing implementations and points out how the classical UX model with keyboard and mouse translates to these new devices. The technical aspect of these requirements will also be highlighted. The audience will get an overview of the status of Open Source in XR and the opportunities for KDE.
We will propose an implementation that integrates VR in the KDE desktop, featuring 3D window management and desktop input synthesis.

Info

Day: 2019-09-08
Start time: 10:45
Duration: 00:30
Room: U4-01

Links:

Konventionelle Ein-Ausgabe

Vom Betriebssystem schon unterstützt

- Input
  - Maus: analoge x,y Koordinaten, 3+ Buttons, Mausrad
  - Touchpad: evtl analoges Scrollen
  - Tastatur: ~104 digitale Tasten
- Output
  - Monitor: x,y Auflösung, Refresh Rate

Konventionelle Ein-Ausgabe

Vom Betriebssystem schon unterstützt

- Input
  - Maus: analoge x,y Koordinaten, 3+ Buttons, Mausrad
  - Touchpad: evtl analoges Scrollen
  - Tastatur: ~104 digitale Tasten
- Output
  - Monitor: x,y Auflösung, Refresh Rate

VR Runtimes

- Input
  - Pose von HMD und Controllern (rotation & xyz position)
  - Controller Buttons, Touchpads, Thumbsticks, …
  - Finger Tracking, Eye Tracking, Full Body Tracking?
- Output
  - Abstand der Linsen voneinander
  - Größe des Bildschirms
  - Distortion Correction

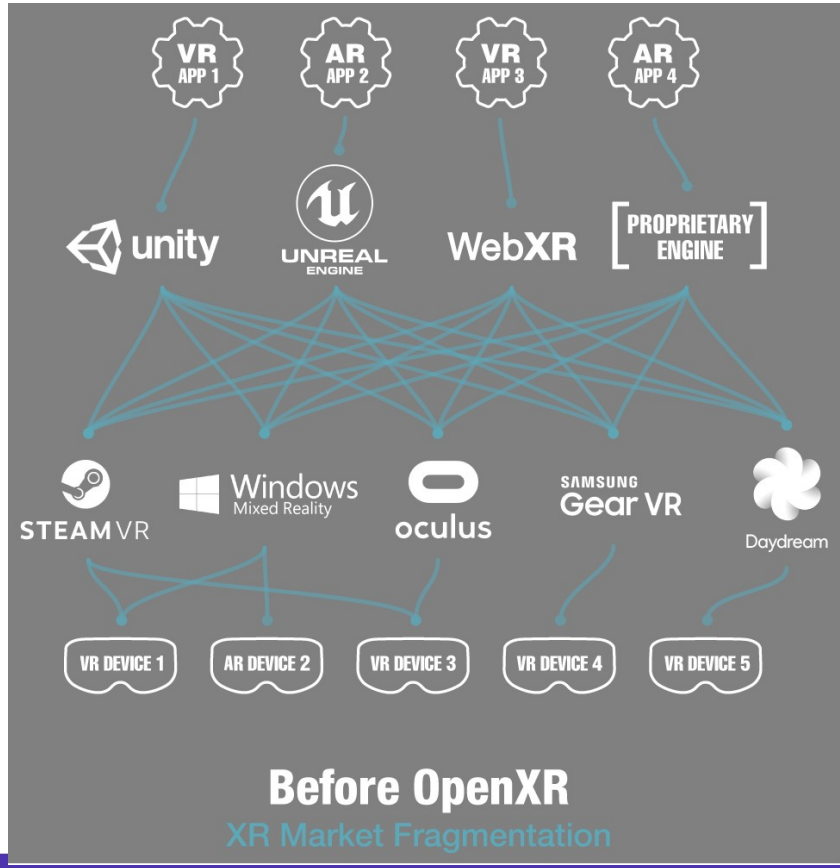
VR Runtimes bisher

- Keine Betriebssystem-Integration
  - Windows MR macht den Anfang, aber ohne Standard
- Jede Runtime hat eigene API
- Wahl der API bestimmt, auf welchen HMDs App läuft
- Hersteller kann API jederzeit ändern

VR Entwicklung bisher

Choose Wisely

- OpenVR: openvr.h, openvr_api.dll
  - Anwender benötigen SteamVR
- Oculus: OVR_CAPI.h, libovr.dll
  - Anwender benötigen Oculus Home
- Windows MR
- OSVR
- OpenHMD
- ...



Mit OpenXR

openxr.h, libopenxr_loader.dll

- Anwendung läuft auf jeder OpenXR runtime
- Aktuelle OpenXR runtimes:
  - Windows MR (Windows MR headsets)
    - Nur WMR headsets
  - Monado
    - Viele mit OpenHMD unterstützte HMDs leider noch of unvollständig (Pos Tracking)
  - (Oculus, nicht offiziell)

Device Plugins erst nach 1.0

COLLABORA

OpenXR Funktionsweise

Ansätze

- SteamVR API für HMD Treiber
  - Kein Standard (außerdem nicht gut dokumentiert)
  - Offizielle Wrapper: Oculus, WMR
- OpenOVR/OpenComposite
  - Unabhängig, SteamVR Apps auf Oculus runtime
- ReVive
  - Unabhängig, Oculus Apps auf SteamVR

Über OpenXR

- Angelehnt an Vulkan Spezifikation
  - Aktuell: Version 0.90 Provisional
  - 1.0 kommt bald mit einigen kleinen Änderungen
- Erweiterbar mit Extensions
  - Funktionalität wurde zugunsten früherer Veröffentlichung weggelassen (z.b. kaum AR)
  - Jeder kann Extensions spezifizieren, aber:

    Nur nützlich, wenn Runtimes sie auch implementieren

OpenXR setup

- xrCreateInstance()
- xrGetSystem()
  - XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY
  - XR_FORM_FACTOR_HANDHELD_DISPLAY
- xrEnumerateViewConfigurations()
  - XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO
  - XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STERE O

OpenXR rendering setup

- xrEnumerateViewConfigurationViews()
  - Mono: 1 view. Stereo: 2 views (links, rechts)
- xrCreateSession(), xrBeginSession()
  - D3D10, 11, 12, OpenGL, OpenGL ES, Vulkan, Headless
  - Bisher: Keine gleichzeitigen Sessions (z.B. “Overlays”)
- xrCreateSwapchain()
  - Texturen, die an die VR Runtime geschickt werden

OpenXR rendering

- xrWaitFrame, xrBeginFrame, xrEndFrame
  - XrCompositionLayerProjection
  - XrCompositionLayerQuad (UI, Video Stream, “Overlays”)
    - Optional: Cube, Cylinder, Depth, Equirect
  - XR_ENVIRONMENT_BLEND_MODE_OPAQUE
  - XR_ENVIRONMENT_BLEND_MODE_ADDITIVE
  - XR_ENVIRONMENT_BLEND_MODE_ALPHA_BLEND
- xrLocateViews()

OpenXR Actions

- Angelehnt an OpenVR Actions
- xrCreateActionSet()
- XrCreateAction()
  - Name, Typ, vorgeschlagene Inputpfad Belegung
  - BOOLEAN, VECTOR1F, VECTOR2F, POSE, VIBRATION
- Inputpfad Beispiel
  - /user/hand/left/input/select/click

OpenXR Actions

- Interaction Profiles
  - Soll vorgeschlagene Inputpfad Bindings vereinfachen
  - khr/simple_controller, google/daydream_controller, htc/vive_controller, htc/vive_pro, microsoft/motion_controller, microsoft/xbox_controller, oculus/go_controller, oculus/touch_controller, valve/knuckles_controller

OpenXR Interaction Profile

6.4.1. Khronos Simple Controller Profile

Path: */interaction_profiles/khr/simple_controller*

Valid for user paths:

- */user/hand/left*
- */user/hand/right*

This interaction profile provides basic pose, button, and haptic support for applications with simple input needs. There is no hardware associated with the profile, and runtimes which support this profile **should** map the input paths provided to whatever the appropriate paths are on the actual hardware.

Supported input sources:

- *.../input/select/click*
- *.../input/menu/click*
- *.../output/haptic*
- *.../input/pointer/pose*
- *.../input/palm/pose*

OpenXR Interaction Profile - Code

```
XrActionCreateInfo actionInfo{XR_TYPE_ACTION_CREATE_INFO};
actionInfo.actionType = XR_INPUT_ACTION_TYPE_VECTOR1F;
xrCreateAction(actionSet, &actionInfo, &gripAction)
    ----
std::array<XrPath, 2> selectPath;
xrStringToPath(instance, "/user/hand/left/input/select/click", &selectPath[0]);
xrStringToPath(instance, "/user/hand/right/input/select/click", &selectPath[1]);
    ----
std::array<XrActionSuggestedBinding, 6> bindings{{ // Fall back to a click input to emulate the grip action.
  {gripAction, selectPath[0]},
  {gripAction, selectPath[1]},
    ----
xrStringToPath(instance, "/interaction_profiles/khr/simple_controller", &khrSimpleInteractionProfilePath));
XrInteractionProfileSuggestedBinding suggestedBindings{XR_TYPE_INTERACTION_PROFILE_SUGGESTED_BINDING};
suggestedBindings.interactionProfile = khrSimpleInteractionProfilePath;
suggestedBindings.suggestedBindings = &bindings[0];
xrSetInteractionProfileSuggestedBindings(session, &suggestedBindings);
```

OpenXR Interaction Profile - Code

```
XrActionCreateInfo actionInfo{XR_TYPE_ACTION_CREATE_INFO};
actionInfo.actionType = XR_INPUT_ACTION_TYPE_VECTOR1F;
xrCreateAction(actionSet, &actionInfo, &gripAction)
    ----
std::array<XrPath, 2> selectPath;
xrStringToPath(instance, "/user/hand/left/input/select/click", &selectPath[0]);
xrStringToPath(instance, "/user/hand/right/input/select/click", &selectPath[1]);
    ----
std::array<XrActionSuggestedBinding, 6> bindings{{ // Fall back to a click input to emulate the grip action.
  {gripAction, selectPath[0]},
  {gripAction, selectPath[1]},
    ----
xrStringToPath(instance, "/interaction_profiles/khr/simple_controller", &khrSimpleInteractionProfilePath));
XrInteractionProfileSuggestedBinding suggestedBindings{XR_TYPE_INTERACTION_PROFILE_SUGGESTED_BINDING};
suggestedBindings.interactionProfile = khrSimpleInteractionProfilePath;
suggestedBindings.suggestedBindings = &bindings[0];
xrSetInteractionProfileSuggestedBindings(session, &suggestedBindings);
```

OpenXR Support

- Praktisch keine echten Programme
- Unreal Engine 4 – nur Windows
- Godot Engine – nur Linux, ohne Controller
- LÖVR (VR support für LÖVE Engine) – WIP
- Alle warten auf OpenXR 1.0?

COLLABORA

Monado

Open Source runtime

- https://gitlab.freedesktop.org/monado/monado
- Aktuell für Linux entwickelt
  - Offen für Windows Port, aber bisher keine Priorität
- Soll auch als Basis für kommerzielle Produkte dienen
- Direkte Vorteile:
  - OpenXR Extension entwickeln und testen für Alle!
  - Portierung auf mehr Plattformen (Standalone AR)

Open Source, auch die Treiber!

- Viele existierende Projekte
  - OpenHMD, Lighthouse Redox, libsurvive, PSMoveService, OpenPSVR, OSVR, (OpenTrack)
  - Relativ, Northstar, Atmos
- Direkte Vorteile von Open Source Treibern:
  - Tracking Code kann modifiziert werden, z.B.
    - Motion Simulator benötigt Kompensation
    - Positions Tracking ersetzen/erweitern

Monado Aktuell

- Compositor auf Vulkan basis mit Direct Mode
  - Distortion Correction Modelle für Vive & panotools
  - OpenXR Apps mit Vulkan und OpenGL
- Positions Tracking in Entwicklung
  - Gute Referenz: OSVR tracking code
  - Guter Fortschritt mit PSMove Controller, PSVR HMD
  - Geplant: Oculus Rift CV1 und Vive Lighthouse
  - Experimenteller libsurvive Treiber für Vive

Further reading

- https://www.khronos.org/registry/OpenXR/specs/0.90/html/xrspec.html
- Einführende Texte in OpenHMD und Monado (Englisch)
  - https://github.com/OpenHMD/OpenHMD/wiki/Getting-Started
  - https://gitlab.freedesktop.org/monado/monado/wikis

Thank you!