Augmented Reality Controls (AR Controls)

1. Introduction

This section describes CGI Studio controls to enable augmented reality applications with CGIStudio, which were designed for generic navigation overlay use case in the automotive area. Typically in **video see-through** AR, a real world video feed is enhanced with graphical elements.

---

2. Use Case: Navigation Overlay

Major elements for this use case are:

- **Video Streaming:** Rendering from a video source into a texture.
- **Camera Tracking:** Updating the moving AR camera via data binding.
- **Navigation Path:** Visualization of the proposed path a navigation system.
- **Points of Interests (POIs):** Loading POIs (e.g. street names, buildings, ...) from an external dictionary and place them in the scene at runtime. Metadata is received as JSON formated string.

Detailed descriptions can be found in Predefined Controls and Predefined Behaviors.



---

3. Prerequisites

3.1. Cmake flags

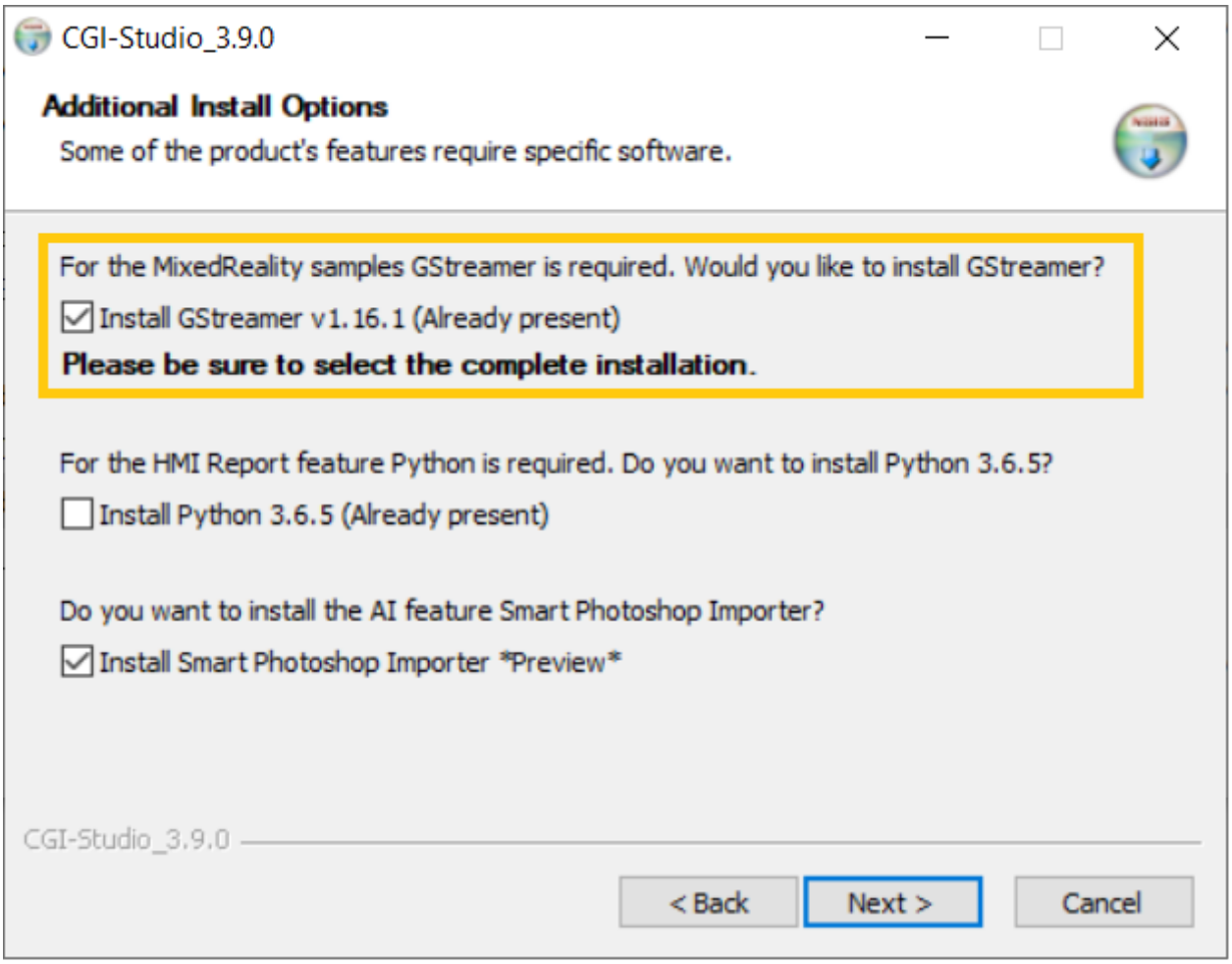
Enable and disabling mixed reality and streaming features:

- `CGIAPP_SAMPLE_STREAMING_ENABLED` is used for the streaming feature. It includes GStreamer libraries from the 3rd party directory.
- `CGIAPP_SAMPLE_MIXEDREALITY_ENABLED` is used for all augmented reality related behaviors and controls.

3.2. GStreamer

GStreamer is the default implementation of the VideoStreamer interface. It is used to decode the video source and provides the data to the VideoStream Behavior. If you want to use GStreamer, a working installation of GStreamer is required. The GStreamer version provided in the CGI Studio 3PSW is GStreamer version 1.16.1. Successfull installation of GStreamer modules can be verified with the console commmand `gst-inspect-1.0`.
Note that, under Windows you might need to add GStreamer to the environment variables. Instructions can be found here: https://gstreamer.freedesktop.org/documentation/installing/on-windows.html

Please also note that, the Android Evaluation Package comes with an installer and by default, the CGI Player is built with streaming enabled. The recommendation is to check the "Install GStreamer" option in the provided installer, and do a "Repair" installation regardless of whether GStreamer is installed or not, because of some dependencies.



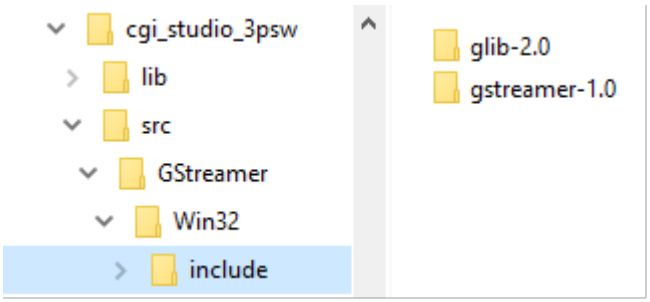
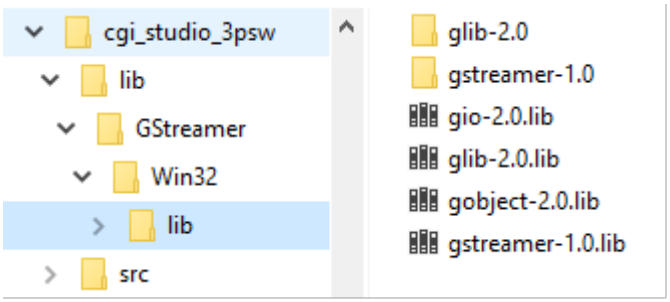
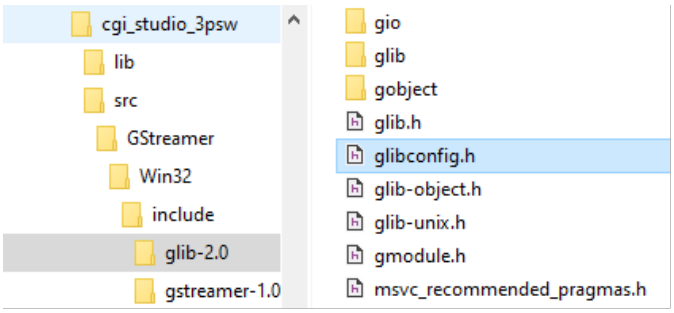
If in any case the CGI Player still looks for a missing dll, check that the GStreamer is set in the PATH environment variable, or copy the GStreamer dlls to the same location as the CGI Player executable.

For more general information, please visit the official GStreamer website: https://gstreamer.freedesktop.org/.

3.3. How to build the CGI Player using GStreamer and Microsoft Visual Studio

- Make sure you have a working Microsoft Visual Studio installation.
- Close Microsoft Visual Studio.
- Install GStreamer for Windows developer and runtime:
  - https://gstreamer.freedesktop.org/data/pkg/windows/1.22.12/mingw/gstreamer-1.0-devel-mingw-x86_64-1.22.12.msi
  - https://gstreamer.freedesktop.org/data/pkg/windows/1.22.12/mingw/gstreamer-1.0-mingw-x86_64-1.22.12.msi
- If you do not have GStreamer directories in the folder `cgi_studio_3psw/lib` and `cgi_studio_3psw/src` you need to copy the following files and folders from your GStreamer

installation directory:

Copy the following folders from your GStreamer installation directory (`C:/gstreamer/1.0/msvc_x86/include`) to folder `cgi_studio_3psw/src/GStreamer/Win32/include` • glib-2.0 • gstreamer-1.0	
Copy the following folders from your GStreamer installation folder (`C:/gstreamer/1.0/msvc_x86/lib`) to folder `cgi_studio_3psw/lib/GStreamer/Win32/lib` • glib-2.0 • gstreamer-1.0 Copy the following library files from your GStreamer installation folder (`C:/gstreamer/1.0/msvc_x86/lib`) to folder `cgi_studio_3psw/lib/GStreamer/Win32/lib` • gio-2.0.lib • glib-2.0.lib • gobjext-2.0.lib • gstreamer-1.0.lib	
Copy the following file from your GStreamer installation folder (`C:/gstreamer/1.0/msvc_x86/lib/glib-2.0/include/glibconfig.h`) into folder `cgi_studio_3psw/src/GStreamer/Win32/include/glib-2.0` • glibconfig.h	

- Open CMake
- Input the location of your source code into the first input field "Where is the source code:" (*<cgi-studio-root>/cmake/Candera/Player*).
- Input the location where to build the binaries to into the input field "Where to build the binaries:"
- To be sure you have a clean build, go to the menu "File - Delete Cache".
- Click the "Configure" button in the middle of the CMake window on the left side.
- A dialog will be opened, choose "Optional platform for generator": Win32
- Wait for CMake to finish configuration.
- In the CMake defines, adjust the following settings:
  - change COURIER_PLATFORM to "imx6platform"
  - enable CGIAPP_SAMPLE_STREAMING_ENABLED
  - enable CGIAPP_SAMPLE_MIXEDREALITY_ENABLED
- Click the "Configure" button again.
- Wait for CMake to finish configuration.

- Press the "Generate" button (just right to the "Configure" button).
- Wait for the Microsoft Visual Studio Solution to be generated.
- Navigate to the Microsoft Visual Studio Solution file in the folder "Where to build the binaries".
- Open the Solution file "Player.sln" with your working Microsoft Visual Studio.
- Wait for the project to load.
- In the Microsoft Visual Studio menu press "Build - Build Solution".
- Wait for the solution to be built.

Finally, you can start the Player application.

---

4. Interfaces

4.1. Events

- **VideoStateEvent:** Event emitted from Video Stream Behaviors to registered behaviors when video state changes. In our sample, the video stream control uses this event to let the PoseTracking behavior know that the video has started and therefore the camera tracking should be started.
- **ArTargetEvent:** Event emitted from target nodes to tracker components to start or stop tracking. In our sample, the PoseTracking Behavior is sending an AddTarget event as message to the CameraTrackerComponent to notify it, that the tracking should start.
- **TrackingEvent:** Event emitted from PoseTracking Behaviors to registered behaviors when the tracking state changes, the position or rotation are updated by the tracker. Possible tracking states are: *Registered, Detected, Tracked,* and *Lost*. In our sample, the PoseTracking Behavior on the node that holds the camera emits this event to the PoiManager Control and the NavigationPath Control, such that they are also informed about the current camera position and rotation, and the current tracking state.
- **PoiUpdateEvent:** Event sent from PoiManager to one of its POIs when a new metadata update arrives. A PoiBehavior will handle the event accordingly. It has a PoiUpdateEventType that states which kind of update this event represents, and a Variant with the value. In our sample, the PoiManager is creating these events based on JSON metadata messages and sending them the respective PoiBehaviors.

4.2. Messages

- **ArTargetEventMsg:** Used to send the ArTargetEvent as message from the view to the model. The only member of the message is the event.
- **PoiReqMsg:** Used by the PoiManager to request a new list of points of interest from the model. Members are the start and end position, which the model uses to build a bounding

rectangle. It will then search for POIs that lie within the bounds of this rectangle and put them into the PoiResMsg.
- **PoiResMsg:**Reply to the PoiReqMsg. Holds a vector of PoiData.

---

5. Known Issues and Limitations

5.1. Streaming

5.1.1 Video Stream

- The property "FlipVertically” in the VideoStream does only work in 2D VideoStream control. In 3D one has to change the UV texture mapping to flip the image. This is why the "no signal" texture is displayed upside down in the 3D sample scenes.

5.1.2 GStreamer

- GStreamer only supports Windows and Linux platforms.
- The default GStreamer modules only support the "Baseline" profile of the H.264 codec.

5.2. Mixed Reality

5.2.1. PoseTracking

- Camera position is a FeatStd::Vector3, which consists of three `FeatStd::Float` values. Therefore, there is a limit on how far the camera can move. Could be an issue.
- Static offsets to position and rotation of the camera that might result from mounting the tracker to the camera must be applied to the camera directly. Offsets cannot be set via data binding so far.

5.2.2. Points of Interest

- Position is not part of metadata message, but independent. It would probably make sense to just move it to metadata and let the respective PoiBehavior position itself in space. The PoiManager does not have to be the one to do it. In fact, the PoiManager already forwards the camera position to the PoiBehavior, but so far, the PoiBehavior only uses it to handle the level of detail fading.
- Alignment is handled by the PoiManger, as in early design, PoiManager was the only one who had access to the camera position. However, this might as well be forwarded to the PoiBehavior as part of the metadata.

Revision #14
Created 2023-02-21 09:07:18 UTC by Kanai Tomoaki
Updated 2026-02-18 03:06:11 UTC by Kanai Tomoaki