SIMIT - Unity Coupling V3.0.0

SIMIT Coupling to Unity

https://support.industry.siemens.com/cs/ww/en/view/109769816

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with Industrial Security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of contents

1 Introduction

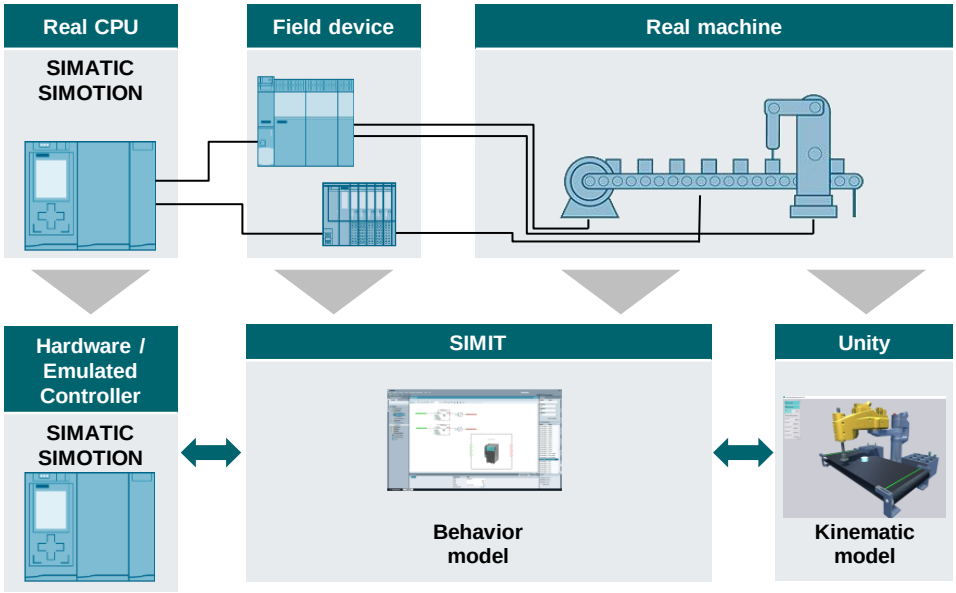
1.1 Overview

To perform a virtual commissioning, a model of the real machine is needed. This model is called the digital twin.

With the help of a digital twin the interaction of the individual components in the virtual world can be simulated and optimized - without having a real prototype. To reduce the risks and effort for the real commissioning, the virtual commissioning of a machine offers an efficient alternative. This enables shorter time-to-market and greater flexibility, efficiency and quality.

STEP 7 and the Totally Integrated Automation Portal (TIA Portal) allow you to create a Hardware- and Software-in-the-Loop scenario in order to simulate and to validate your PLC program. With Unity, machine builders can simulate and test the mechanical components of their machine in a virtual environment. The behavior of active components, such as drives or valves, is emulated with the SIMIT simulation software. This combination helps with the preparation and for a problem-free commissioning. Furthermore, these tools make it possible to validate the mechanical concept of the machine, and the interaction of the mechanical system, the electrical system, the software as well as the user program, at an early development phase of a plant.

Figure 1-1 Overview of the systems



This application example consists of the SIMIT coupling to Unity and a guidance to implement Unity scripts for the communication with SIMIT.

The SIMIT - Unity Coupling can be easily implemented in an already installed SIMIT v10.3 and newer software installation.

This external coupling is prepared to communicate isochronously with a Unity Build or Unity Project.

1.2 Components used

This application example has been created with the following software components:

Table 1-1 Software components

Component	Article number
STEP 7 Professional V17 + WinCC Professional	6ES7822-1..07-..
PLCSIM Advanced 4.0 SP1 HF1	6ES7823-1FA03-0YA5
SIMIT S V11.0	6DL8913-…01-….
Unity 2021.3 LTS (Unity Pro)	

This application example consists of the following components:

Table 1-2 Components

Component	File name	Note
Documentation	Manual_SIMIT-Unity_Coupling_V3_0_0_EN.pdf	
SIMIT Coupling	UnityCouplingConfiguration.exe	Subfolder: SIMIT_Coupling
SIMIT V11.0 Project	SIMIT-Unity_Coupling_V3_0_0.simarc	SIMIT archive
Unity Build	UnityGettingStartedExampleV3_0_0.exe	Subfolder: Unity_Build
Unity Package	Unity_Package\package.json	Subfolder: Unity_Package
TIA Portal V17 Project	GettingStarted.zap17	TIA Portal archive

Limitations

This application example does not contain any descriptions of the following topics:

- Basics of TIA Portal configuration
  https://support.industry.siemens.com/cs/ww/en/view/109798671
- Basics of SIMIT
  https://support.industry.siemens.com/cs/ww/en/view/109801804
- Basics of Virtual Commissioning
  https://support.industry.siemens.com/cs/ww/en/view/109758943
- Unity Community
  https://Unity3d.com/commUnity
- Unity Learning
  https://Unity3d.com/de/learn

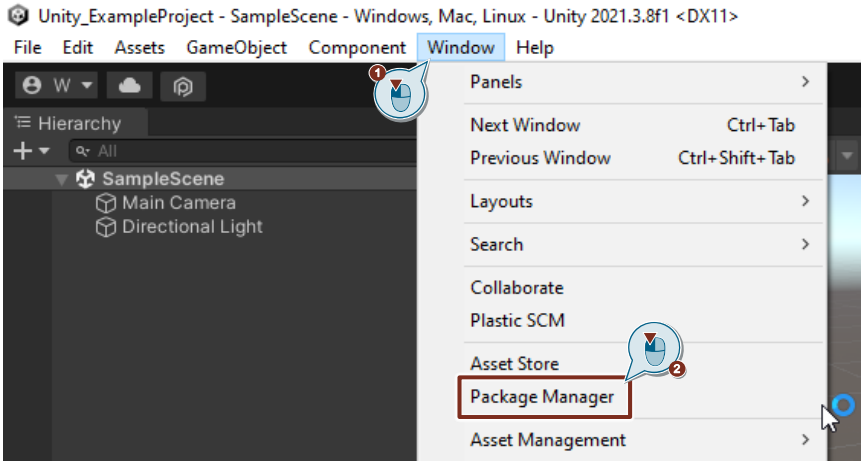
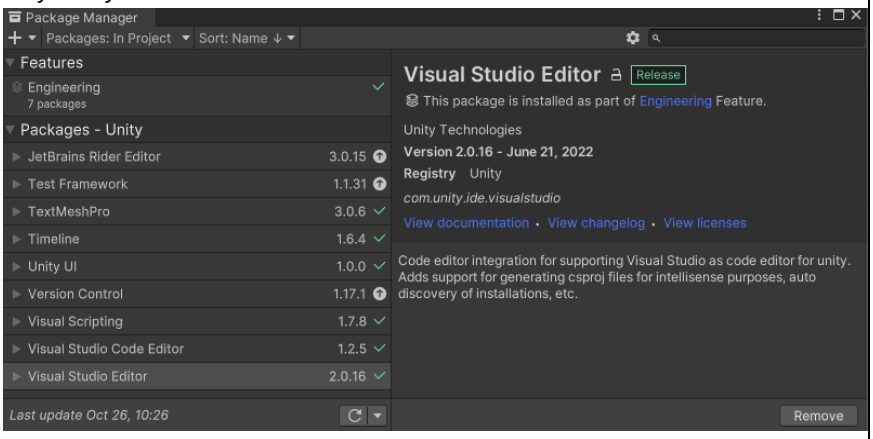
To understand this application example, it is assumed that readers have adequate knowledge of these topics.

2 Unity - Requirements for SIMIT coupling

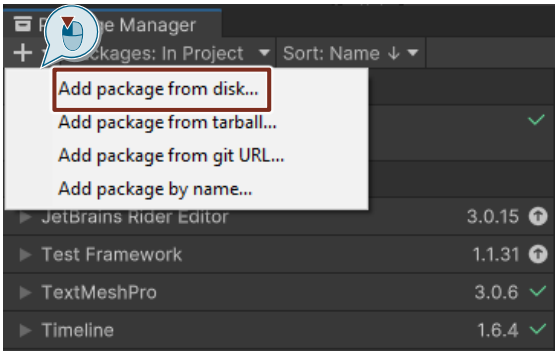
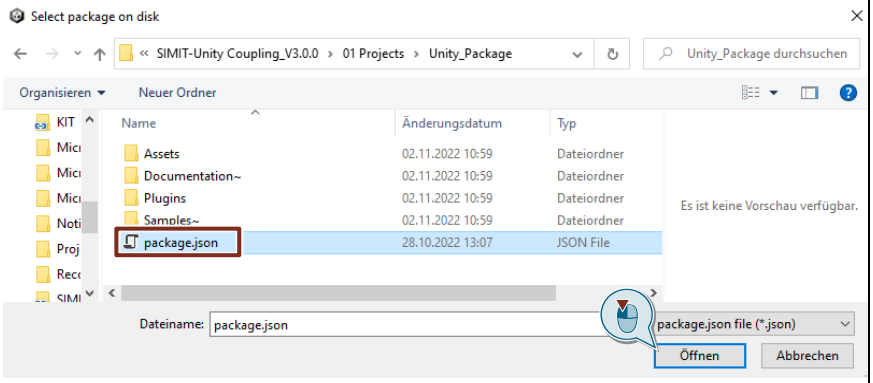
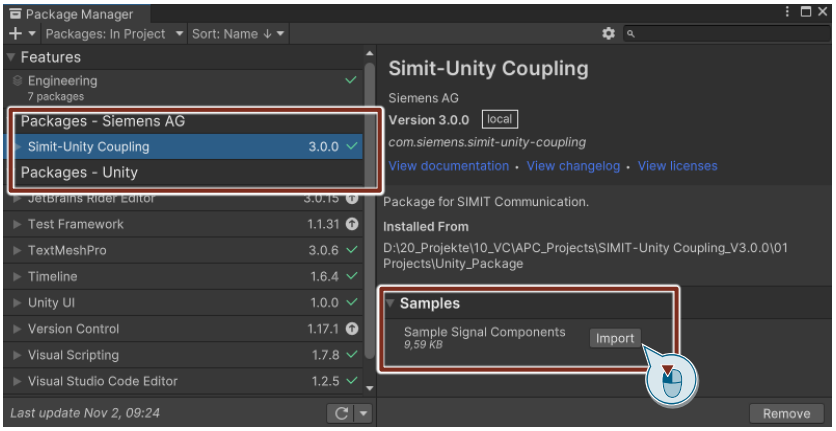
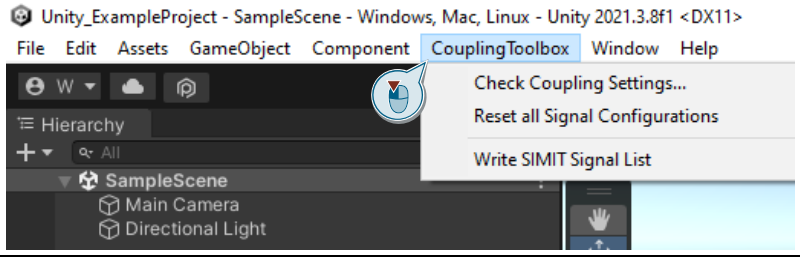
2.1 Installing a package from a local folder

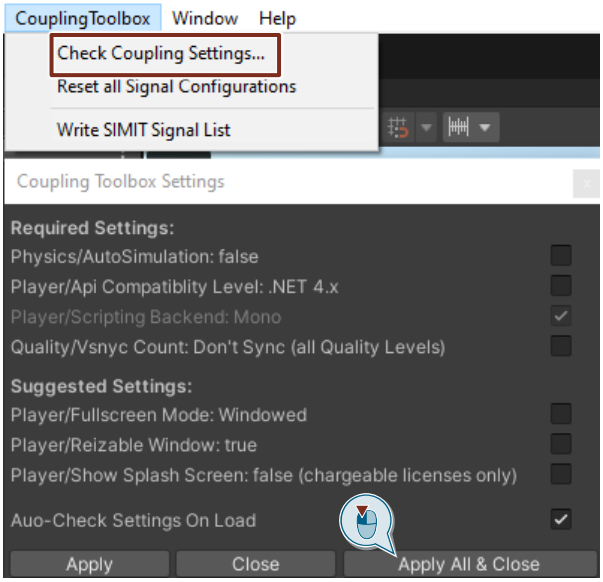
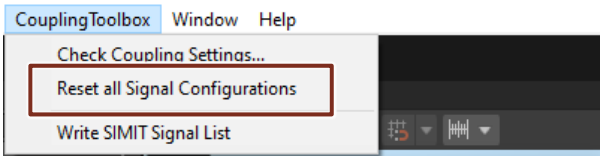
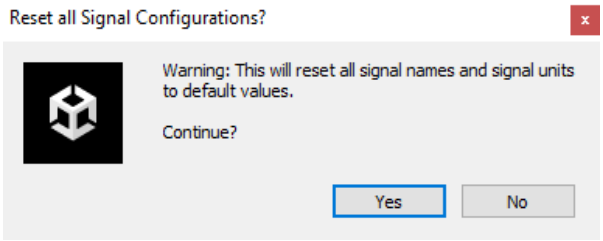
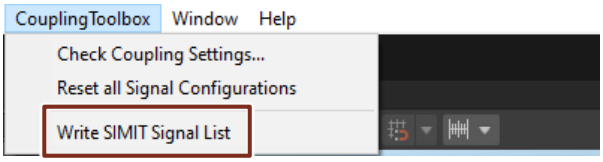
The process of installing a unity package from a local folder is described in the following. Additional information can be found in the Unity manual.[1]

Table 2-1 Installing Unity package

No.	Action
1.	Via the Unity „Package Manager" it is possible to install packages from a local folder. 
2.	The "Package Manager" shows an overview of alle used packages. The used packages can be updated directly. New packages can be implemented also very easily. 

[1] https://docs.unity3d.com/Manual/upm-ui-local.html

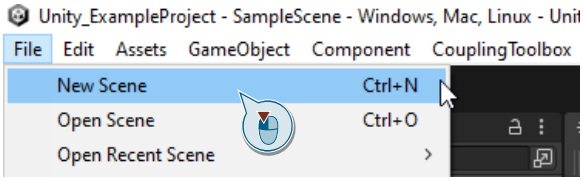
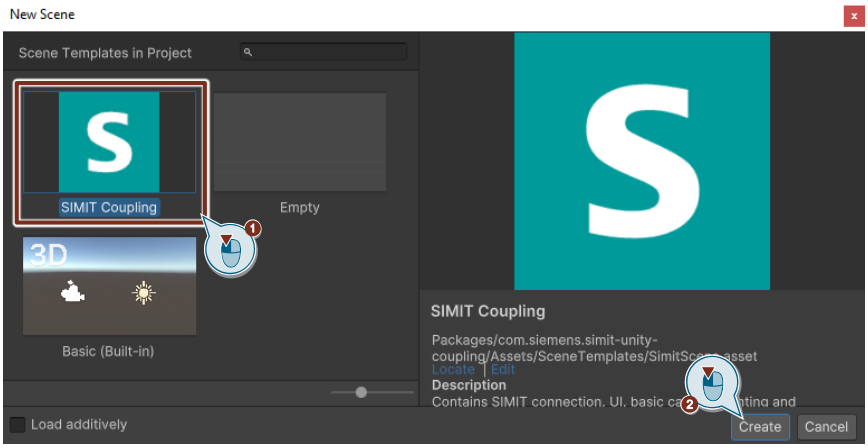
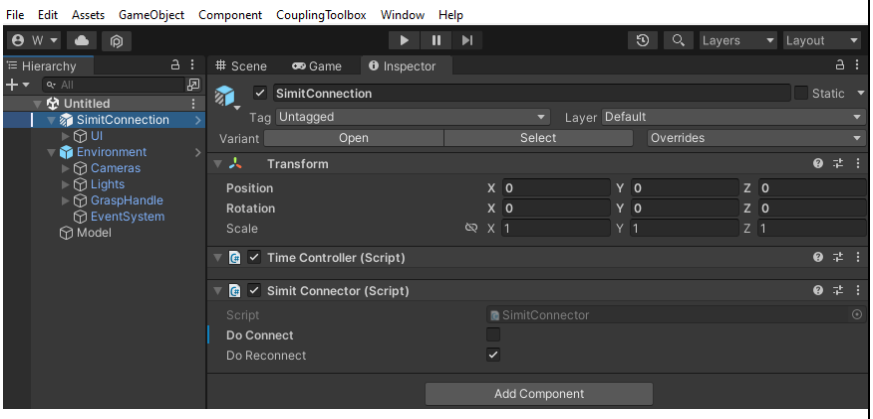
No.	Action
3.	Via “+”, new packages from different sources can be imported. In case of the simit-unity-coupling packages “Add package from disk” must be chosen.  The file “package.json” in the “Unity_Package” folder must be chosen to import.  If the import is done, the SIMIT-Unity Coupling is displayed and can be used in the project. To get an overview about the programming Styleguide, sample scripts can be imported to the project manually.  Additionally, the “Coupling Toolbox” is installed in the tollbar. The Coupling Toolbox implements different features to ensure a correct signal exchange with SIMIT. 

No.	Action
4.	**Check Coupling Settings** It is recommended to apply all Coupling Toolbox Settings to ensure a correct behavior.  **Reset all Signal configurations** Resets all signal names and signal units to default values.   **Write SIMIT Signal List** Updates the SIMIT Signal List. The SIMIT Signal List is after coupling with SIMIT visible in the Unity coupling. 

2.2 SIMIT - Unity Coupling scene template

The imported Unity package implements a scene template for new projects. The scene template implements all necessary objects to establish a SIMIT connection and additional features.

Table 2-2 SIMIT Coupling scene template

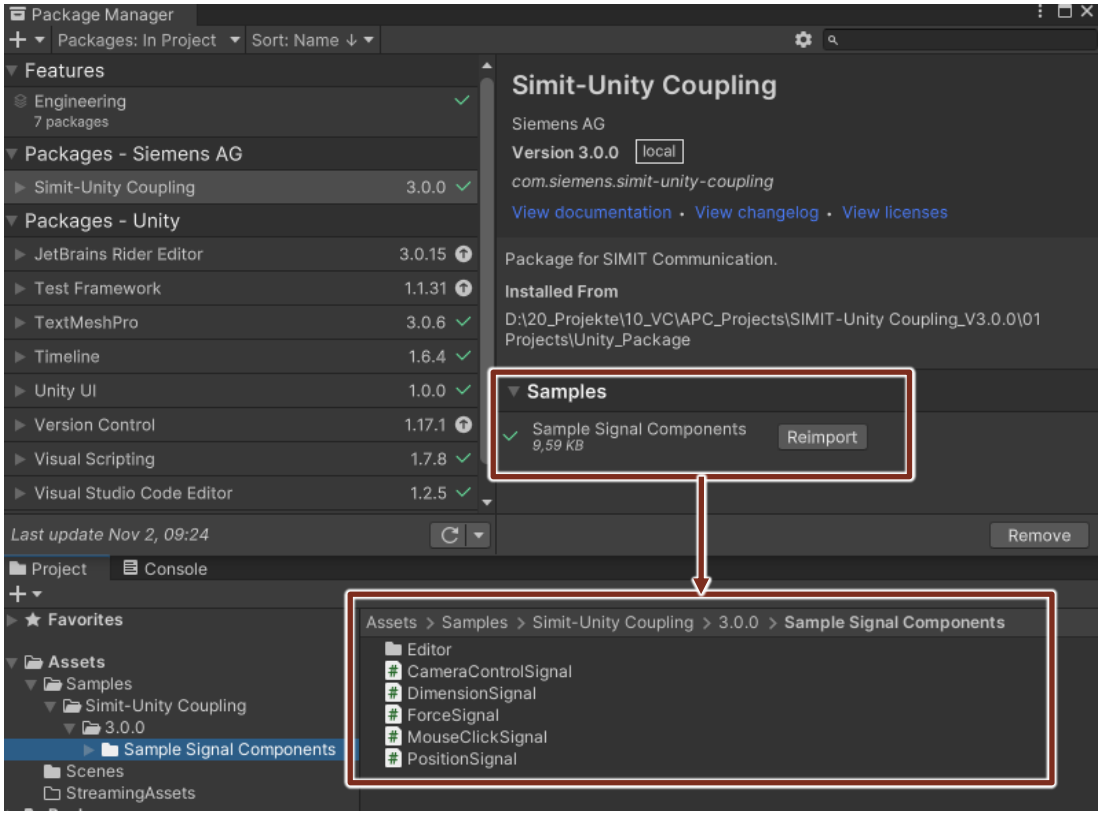
No.	Action
1.	
2.	Select the SIMIT Coupling scene template The scene template implements the SIMIT connection, UI, basic camera, lightning and interaction functionalities in the project. 
3.	A new Unity scene is created. It automatically implements the SIMIT connection and additional features. 

2.3 Sample scripts for Coupling Toolbox

The Unity package implements example scripts to get an overview about the programming styleguide for scripts interacting with the Coupling Toolbox.

NOTE The imported sample scripts should give just a simple guideline to implement further signals to the SIMIT-Unity Coupling Toolbox.

Figure 2-1 Sample scripts



CameraControlSignal

The camera control signal script implements an example for switching between several cameras while simulation. The input signal “CameraControl” can be controlled via an integer value (Signal from SIMIT).

Figure 2-2 Sample script for camera control signal

CameraControlSignal.cs

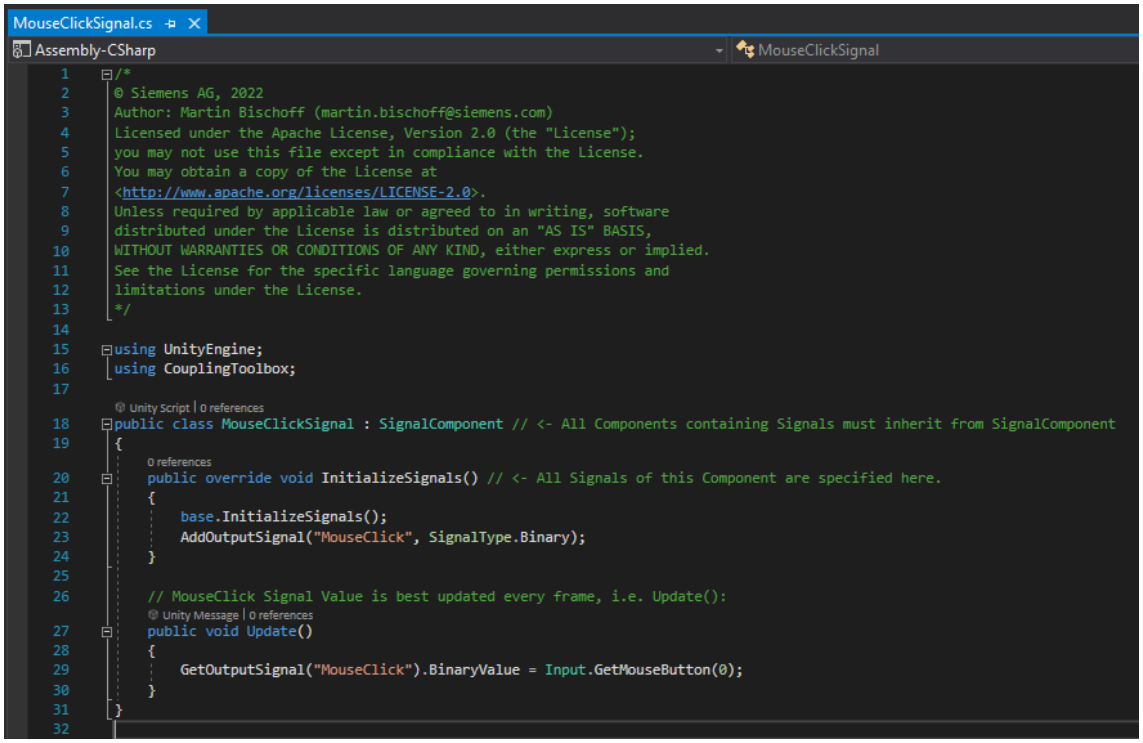
Assembly-CSharp — CameraControlSignal

```
/*
© Siemens AG, 2022
Author: Martin Bischoff (martin.bischoff@siemens.com)
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
<http://www.apache.org/licenses/LICENSE-2.0>.
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/

using UnityEngine;
using CouplingToolbox;

Unity Script | 0 references
public class CameraControlSignal : SignalComponent // <- All Components containing Signals must inherit from SignalComponent
{
    private Camera[] cameras;
    public int CameraNumber;

    Unity Message | 0 references
    protected void Awake()
    {
        cameras = FindObjectsOfType<Camera>();
        UpdateCameras();
    }

    0 references
    public override void InitializeSignals() // <- All Signals of this Component are specified here.
    {
        base.InitializeSignals();
        AddInputSignal("CameraControl", SignalType.Integer);
    }

    // CameraControl Signal Value is best updated every frame, i.e. Update():
    Unity Message | 0 references
    public void Update()
    {
        int cameraNumber = (int) GetInputSignal("CameraControl").IntegerValue % cameras.Length;
        if (cameraNumber != CameraNumber)
        {
            CameraNumber = cameraNumber;
            UpdateCameras();
        }
    }

    2 references
    private void UpdateCameras()
    {
        for (int i = 0; i < cameras.Length; i++)
            cameras[i].enabled = i == CameraNumber;
    }
}
```

MouseClickSignal

The mouse click signal script implements an output signal (binary) to read out each mouse click while simulation (Signal to SIMIT).

Figure 2-3 Sample script for mouse click signal

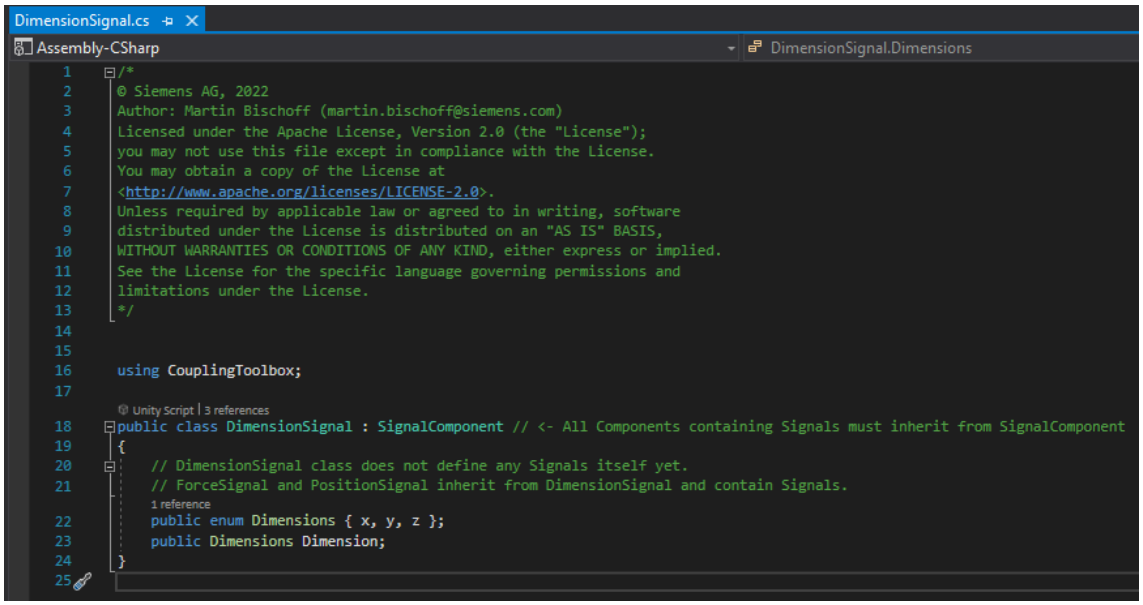


```
MouseClickSignal.cs
Assembly-CSharp                                   MouseClickSignal
 1  /*
 2  © Siemens AG, 2022
 3  Author: Martin Bischoff (martin.bischoff@siemens.com)
 4  Licensed under the Apache License, Version 2.0 (the "License");
 5  you may not use this file except in compliance with the License.
 6  You may obtain a copy of the License at
 7  <http://www.apache.org/licenses/LICENSE-2.0>.
 8  Unless required by applicable law or agreed to in writing, software
 9  distributed under the License is distributed on an "AS IS" BASIS,
10  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
11  See the License for the specific language governing permissions and
12  limitations under the License.
13  */
14
15  using UnityEngine;
16  using CouplingToolbox;
17
    Unity Script | 0 references
18  public class MouseClickSignal : SignalComponent // <- All Components containing Signals must inherit from SignalComponent
19  {
        0 references
20      public override void InitializeSignals() // <- All Signals of this Component are specified here.
21      {
22          base.InitializeSignals();
23          AddOutputSignal("MouseClick", SignalType.Binary);
24      }
25
26      // MouseClick Signal Value is best updated every frame, i.e. Update():
        Unity Message | 0 references
27      public void Update()
28      {
29          GetOutputSignal("MouseClick").BinaryValue = Input.GetMouseButton(0);
30      }
31  }
32
```

DimensionSignal

The dimension signal script implements an example for dimensioning objects in x, y and z. It implements the base for reading and writing values on the specific axes (x, y, z).

Figure 2-4 Sample script for dimension signal



ForceSignal

The force signal script implements an input signal to write a force value on a rigid body (Signal from SIMIT). The force value can be added to the x, y or z direction.

The calculation is done in the PreStep().

Figure 2-5 Sample script for force signal

ForceSignal.cs

Assembly-CSharp

ForceSignal

```
/*
© Siemens AG, 2022
Author: Martin Bischoff (martin.bischoff@siemens.com)
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
<http://www.apache.org/licenses/LICENSE-2.0>.
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/

using UnityEngine;
using CouplingToolbox;

[RequireComponent(typeof(Rigidbody))]
Unity Script | 0 references
public class ForceSignal : DimensionSignal, IPreSimulator // <- Please note inheritance and interface implementation.
{
    public Vector3 forceVector;
    private Rigidbody _rigidbody;

    Unity Message | 0 references
    public void Awake()
    {
        forceVector = new Vector3();
        _rigidbody = GetComponent<Rigidbody>();
    }

    0 references
    public override void InitializeSignals() // <- All Signals of this Component are specified here.
    {
        base.InitializeSignals();
        AddInputSignal("AddForce", SignalType.Analog, Quantity.Force);
    }

    // PreStep is called directly before every physics step after receiving new signal values.
    // Here is the best place for applying updated setpoint signal values which are consumed every physics step.
    0 references
    public void PreStep(float timeStep)
    {
        forceVector[(int)Dimension] = (float)GetInputSignal("AddForce").GetDoubleInSI();
        _rigidbody.AddForce(forceVector);
    }

}
```

PositionSignal

The position signal script implements an output signal to read out a position value of a game object (Signal to SIMIT). The position value can read out the x, y and z direction.

The calculation is done in the PostStep().

Figure 2-6 Sample script for position signal

PositionSignal.cs

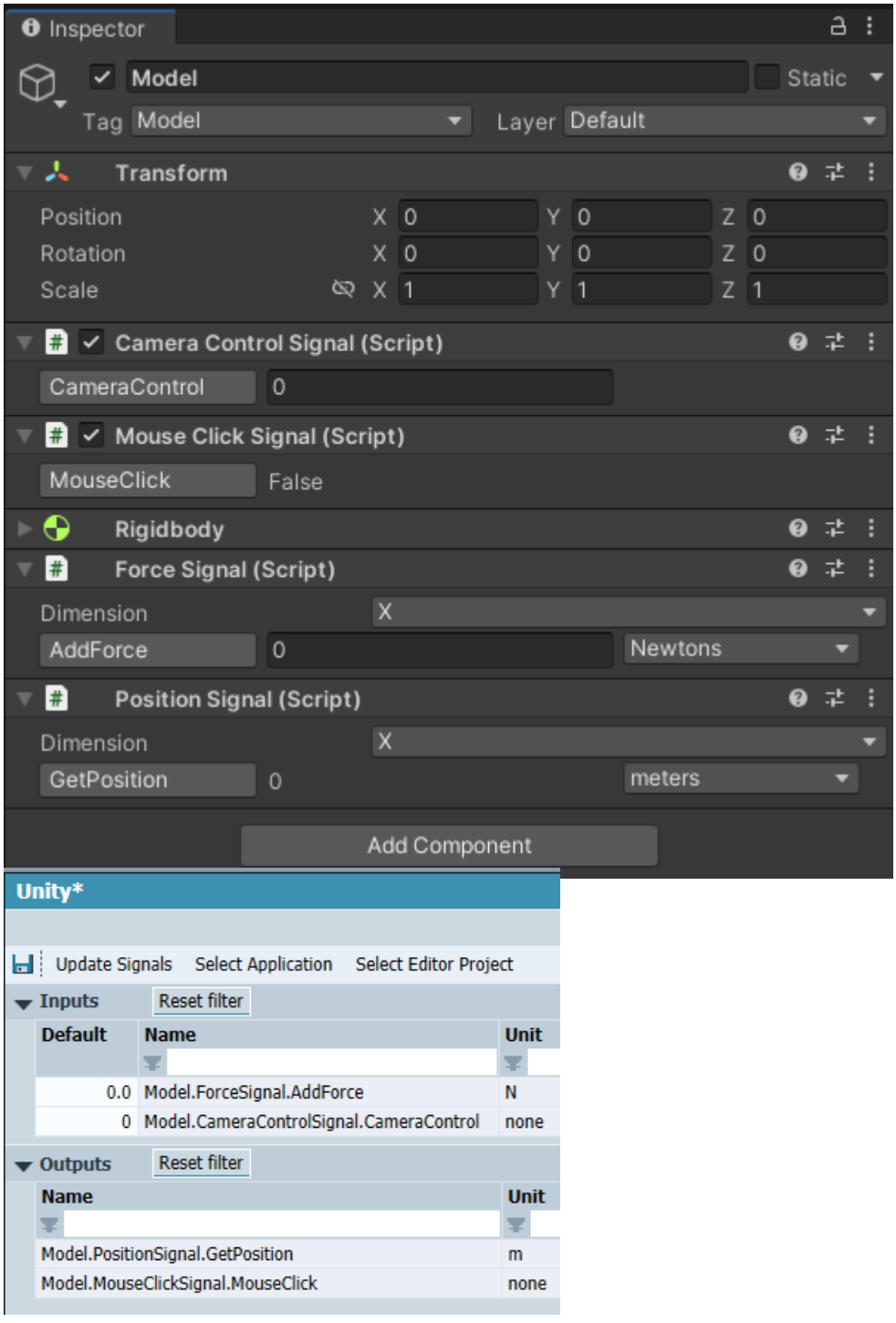
Assembly-CSharp — PositionSignal

```
/*
© Siemens AG, 2022
Author: Martin Bischoff (martin.bischoff@siemens.com)
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
<http://www.apache.org/licenses/LICENSE-2.0>.
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/

using CouplingToolbox;


Unity Script | 0 references
public class PositionSignal : DimensionSignal, IPostSimulator  // <- Please note inheritance and interface implementation.
{
    0 references
    public override void InitializeSignals() // <- All Signals of this Component are specified here.
    {
        base.InitializeSignals();
        AddOutputSignal("GetPosition", SignalType.Analog, Quantity.Length);
    }

    // PostStep is called directly after every physics step before sending new signal values.
    // Here is the best place for applying updated state signal values which are produced every physics step.
    0 references
    public void PostStep(float timeStep)
    {
        GetOutputSignal("GetPosition").SetDoubleInUnit(transform.position[(int)Dimension]);
    }
}
```

SIMIT – Unity coupling example signals

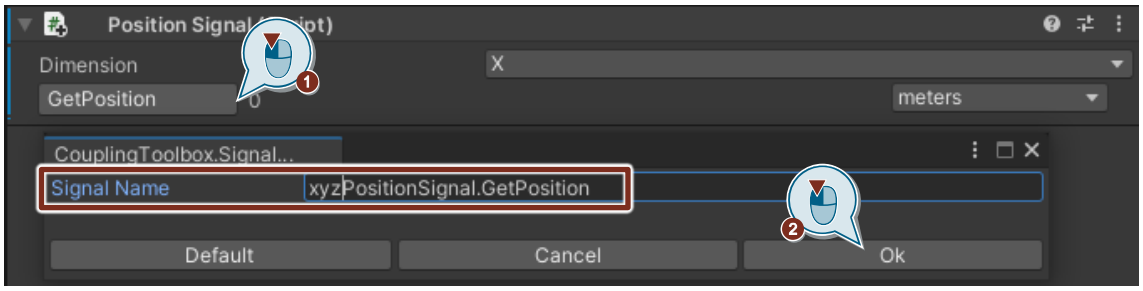
The following figure shows the example signals imported in SIMIT.

Figure 2-7



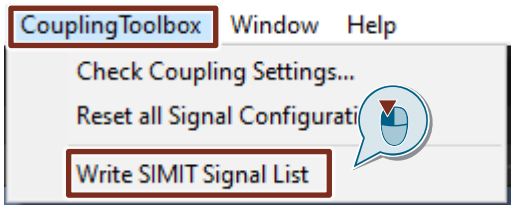
The signal names can be changed within the Unity project via click on the signals.

Figure 2-8 Change signal names



After changing signal names, the SIMIT signals list must be refreshed. To update the signal list, click “Write SIMIT Signal List” in the “Coupling Toolbox”.

Figure 2-9 Write SIMIT Signal List



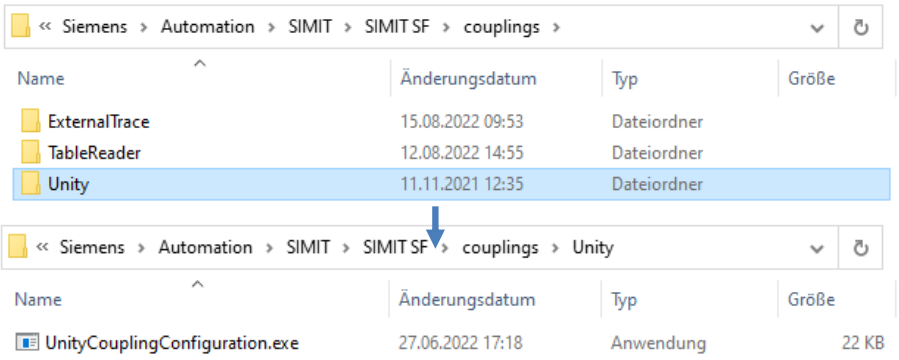
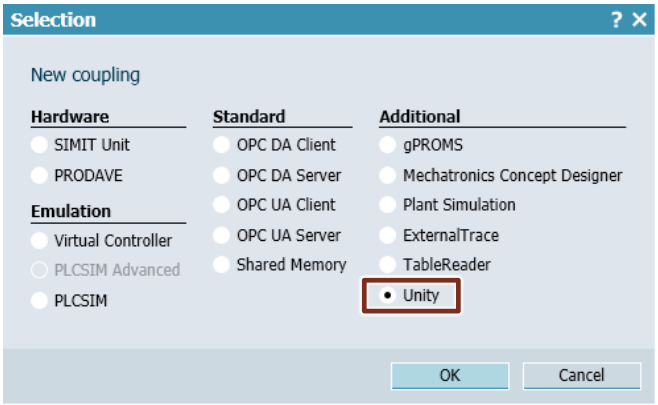
3 Installation of Unity Coupling in SIMIT

The Unity coupling is based on an external coupling for SIMIT and must be added manually. The required steps are described in the following table.

3.1 Implementing the Unity Coupling in SIMIT

NOTE The name of the Unity coupling folder in …\SIMIT SF\couplings must be “Unity”.

Table 3-1 Implementation of Unity Coupling in SIMIT

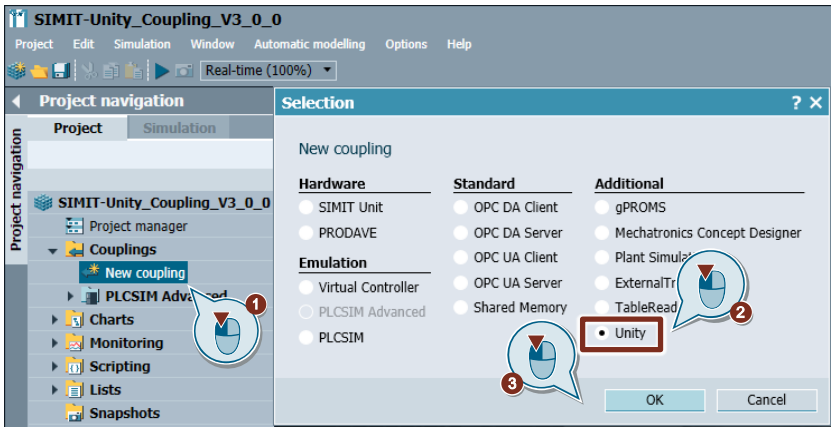
No.	Action
1.	Copy the “UnityCouplingConfiguration.exe” from folder “SIMIT_Coupling” into a new folder "Unity" in the following directory: …\Siemens\Automation\SIMIT\SIMIT SF\couplings If the folder "couplings" does not exist yet, please create it. 
2.	Restart SIMIT to update the couplings dialogue. The new coupling "Unity" is visible in the coupling selection dialog. 

3.2 Creating a Unity Coupling in SIMIT

Via a “Unity” Coupling, a connection to either the Unity Editor Project or a Unity Application (Build) can be established. Both options are explained in the following.

When coupling with the Unity Editor Project, the Unity project can be modified and tested quickly and simultaneously to modifications on the SIMIT project. Couplings with Unity applications (Build) have a better performance and require no Unity installation. The usage of the coupling in detail is explained in chapter 5.

Figure 3-1 Creating a new Unity Coupling in SIMIT



Via “Unity Coupling Configuration” the properties of the coupling can be checked and adjusted.

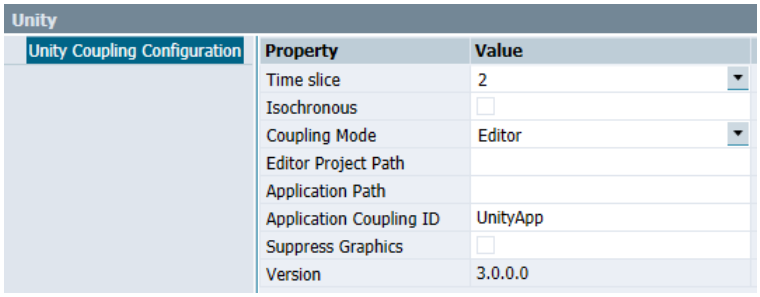
The checkbox “Isochronous” enables the synchronization to a time slice, if the operation mode of the project is isochronous, too.

Via “Coupling Mode” the coupling can switch between Unity Editor and Unity Application connection.

The “Application Coupling ID” can be adjusted to show a specific name in the runtime information.

Via “Suppress Graphics”, the visualization of graphics can be switched of completely to reach a much better simulation performance.

Figure 3-2 Unity Coupling Configuration



4 Simulation with Unity

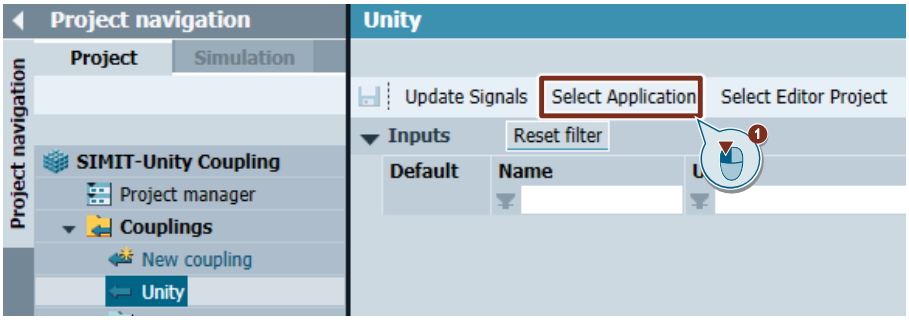
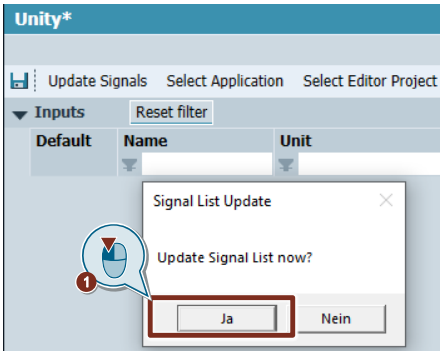
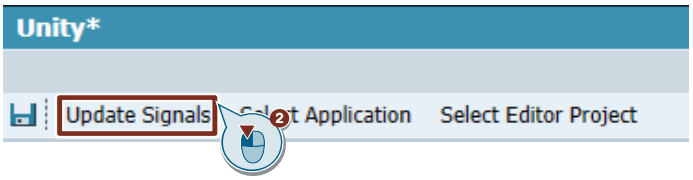
The following chapter describes different scenarios of simulating with Unity.

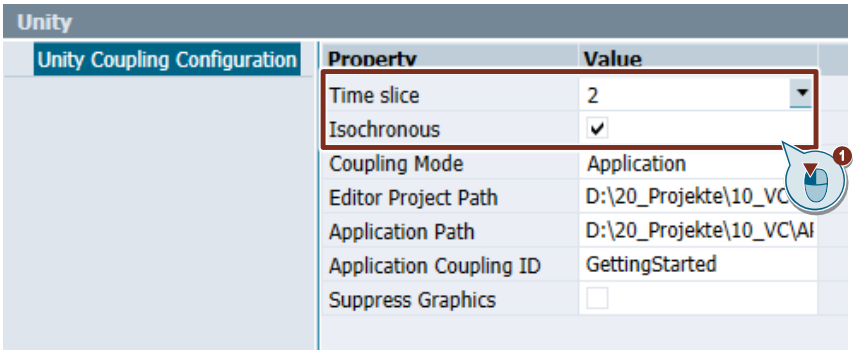
4.1 Coupling SIMIT with Unity Build

This chapter describes the coupling of SIMIT to a Unity Build.

If SIMIT should be connected to a Unity Build, the “Unity_Package\package.json” must be part of the Unity project.

Table 4-1 Coupling with Unity Build

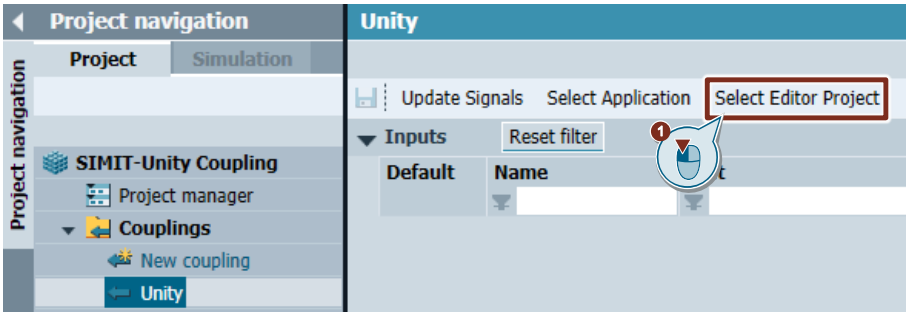
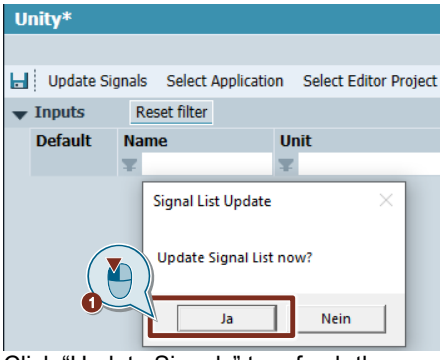
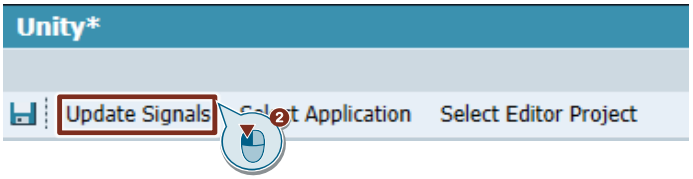
No.	Action
1.	Click “Select Application” to choose a Unity Application (Build).  Choose the Unity Build of the Unity project. Within this application example a Unity Build “UnityGettingStartedExampleV3_0_0.exe” including the “GettingStartedExample” scene is provided.
2.	After choosing a Unity Application, the Signal List Update Popup must be acknowledged to read out all input and output signal information from the Unity Application.  Click “Update Signals” to refresh the coupling e.g. after changing the Unity Application.  On success, all signal information is listed in the coupling. Save the coupling.

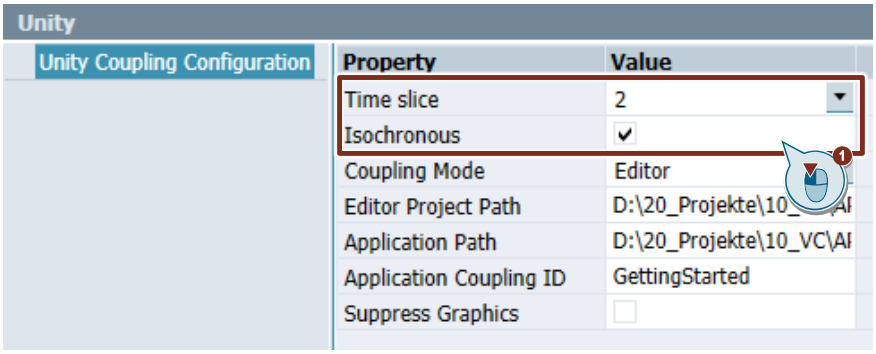
No.	Action
3.	To ensure the isochronous connection type of the Unity coupling, select the checkbox and correct Time slice. 

4.2 Coupling SIMIT with Unity Editor

The coupling to the Unity Editor is explained in the following table.
The Unity Project executed in the Unity Editor in Step 2 must include the "Unity_Package\package.json".

Table 4-2 Coupling with Unity Editor Project

No.	Action
1.	Click "Select Editor Project" to choose the directory of the Unity Project. 
2.	After choosing "Select Editor Project", the Signal List Update Popup must be acknowledged to read out all input and output signal information from the Unity Project.  Click "Update Signals" to refresh the coupling e.g. after changing the Unity Editor Project.  On success, all signal information is listed in the coupling. Save the coupling.

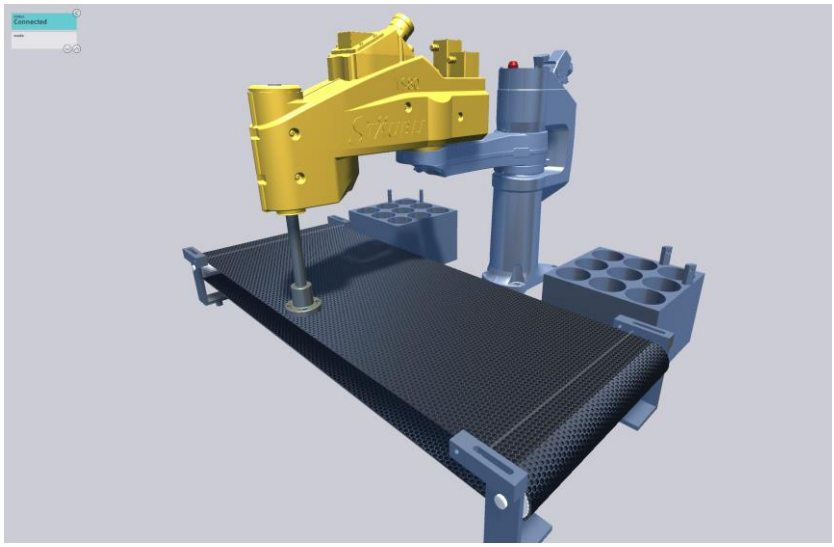
No.	Action
3.	To ensure the isochronous connection type of the Unity coupling, select the checkbox and correct Time slice. 

5 SIMATIC Machine Simulator - Getting Started Example

In this application example a ready-to-run simulation, including Unity Build, SIMIT and TIA Portal project.

Detailed information about operating, testing and failure handling of this application example (SIMIT + TIA Portal project), can be found in the documentation of the SIMATIC Machine Simulator Getting Started.[2]

Figure 5-1 Unity Getting Started Example



5.1 SIMIT Project - Getting Started Example

A physical axis simulation in Unity combined with a synchronized behavior model in SIMIT results in a deterministic position control loop.

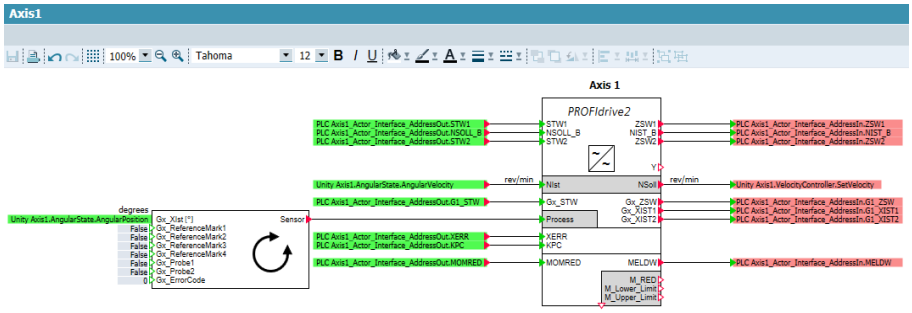
NOTE Since SIMIT v10.3 the External Coupling supports a synchronized communication.

[2] https://support.industry.siemens.com/cs/ww/en/view/109758943

SIMIT chart “Axis1” and “Axis2”

TIA technology objects for Axis1 and Axis2 are not in simulation mode. Due to this, the PROFIdrive telegram is used for the communication.

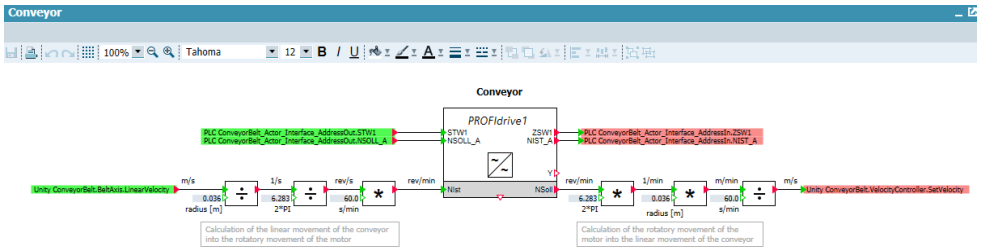
Figure 5-2 Axis1 simulation SIMIT chart



SIMIT chart “Conveyor”

The SIMIT chart “Conveyor” shows the drive simulation of the Conveyor system.

Figure 5-3 Conveyor simulation SIMIT chart



SIMIT chart “Bitlogic”

The SIMIT chart “Bitlogic” shows special I/O signals of the machine.

There are also buttons to interact with the simulation, directly.

- Trigger new Product in Unity

  Produces a new workpiece in Unity.
- Clear all Products in Unity

  Deletes all created workpieces in Unity.
- Deactivate Sensors-Failure Test

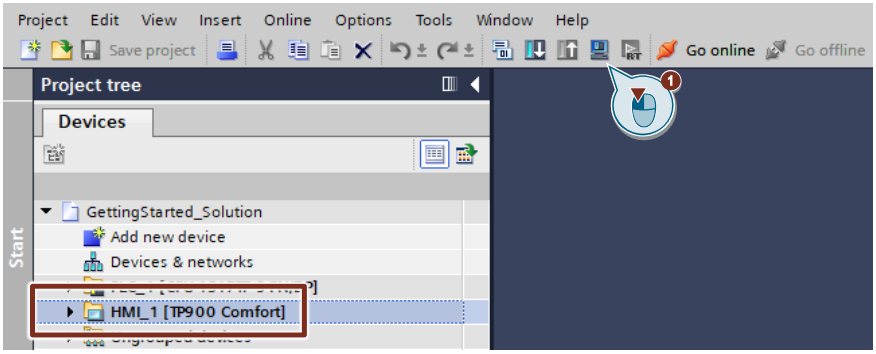
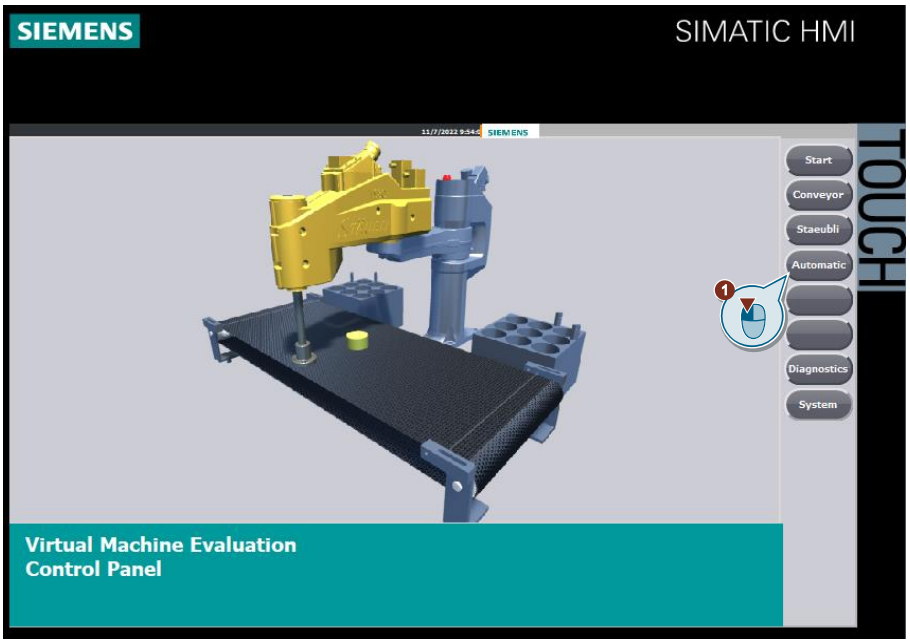
  Deactivates the chosen light barrier to test failure scenarios.

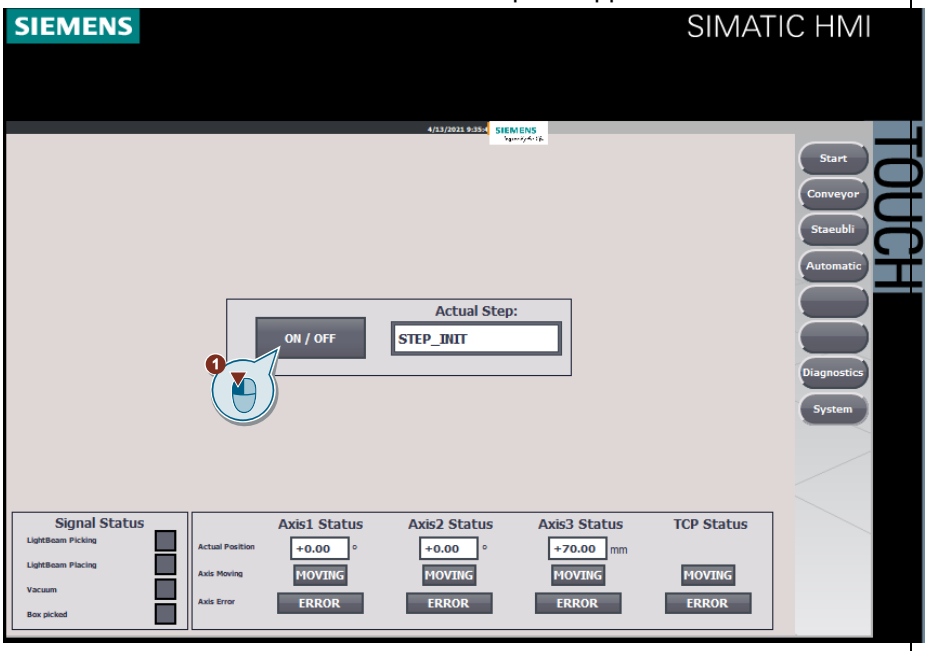
Figure 5-4 Bitlogic chart in SIMIT

5.2 Starting the Getting Started Example via HMI Runtime

When the Unity application is operational in SIMIT simulation mode as described in chapter 5.3. the PLC application of the Getting Started example can be executed as described in the following table.

Table 5-1 Starting the application via WinCC Runtime

No.	Action
1.	Start the WinCC Runtime 
2.	Switch to Automatic mode in the WinCC Runtime. 

No.	Action
3.	Click the "ON / OFF" button to start and stop the application. 

5.3 Operation

This chapter describes, how to start the simulation of the example project.

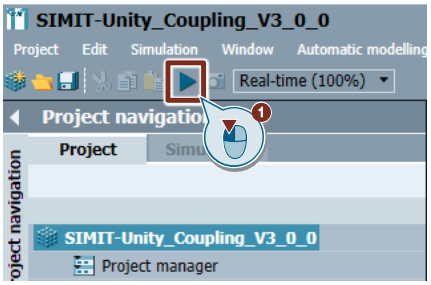
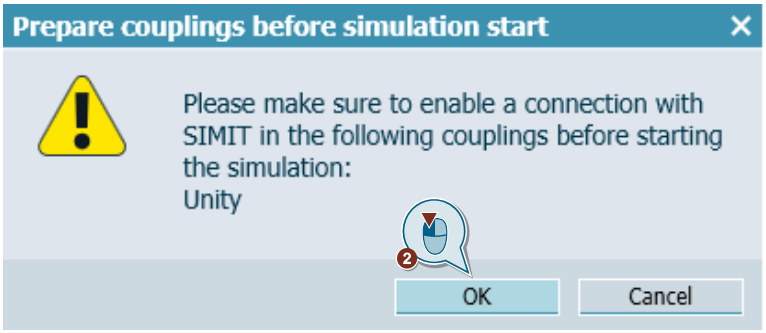
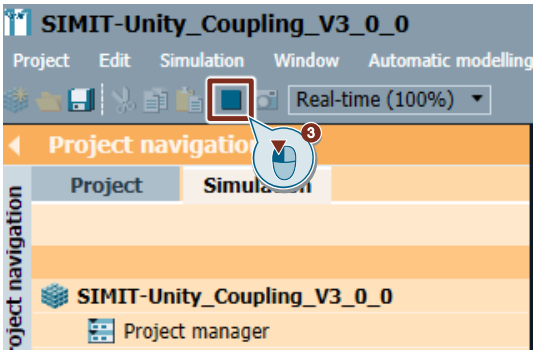
The digital twin is controlled via SIMIT and WinCC Runtime as shown in the following.

5.3.1 Starting the SIMIT simulation, connected to a Unity Application

If all couplings and charts in SIMIT are prepared, the simulation can be started in SIMIT.

An orange background indicates that the simulation is running.

Table 5-2 Starting the SIMIT simulation, connected to a Unity Build

No.	Action
1.	Click the “Start“ button to start the SIMIT simulation. → Unity Application will start up in run mode automatically. → PLCSIM Adv. Instance will start up automatically.  Click “Ok” to accept the connection to an external coupling. 
2.	Click the “Stop” button to stop the simulation. →Unity Application will shut down automatically. →PLCSIM Advanced instance will be shut down automatically. 

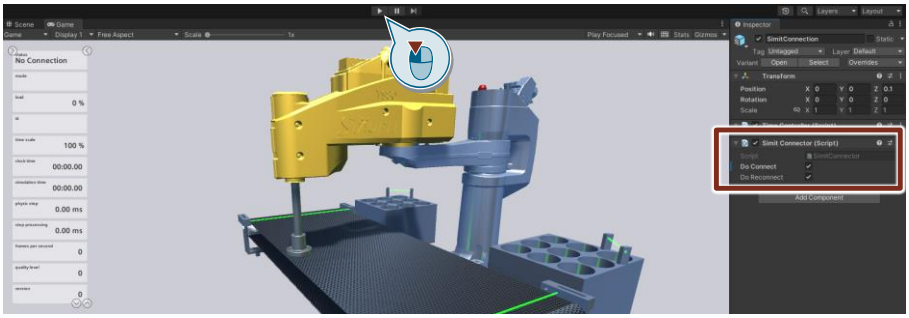
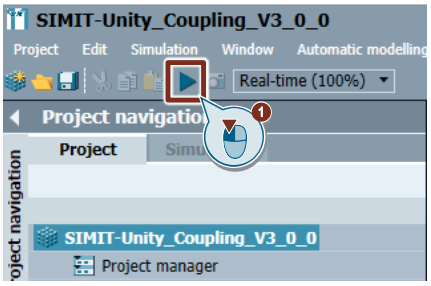
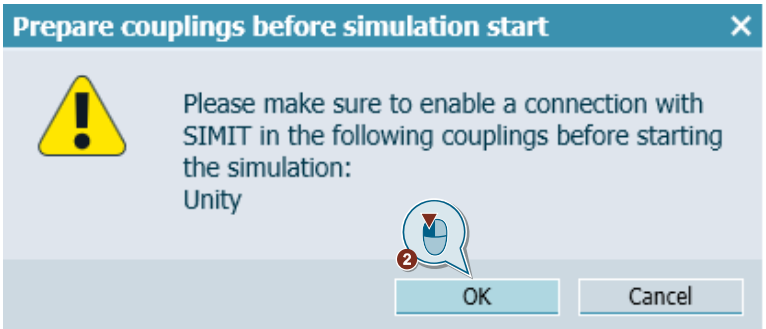
5.3.2 Starting the SIMIT simulation, connected to a Unity Editor Project

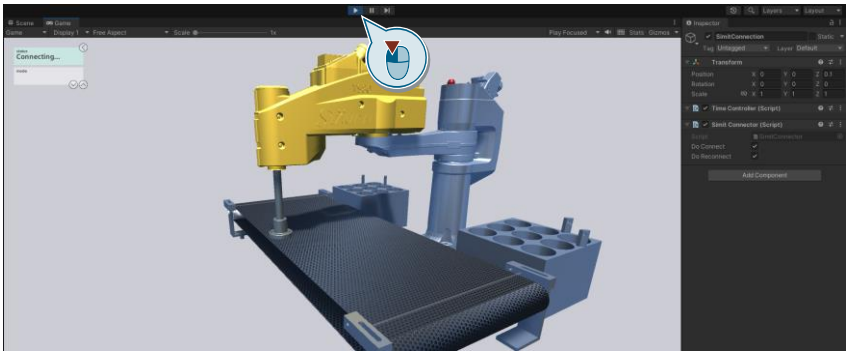
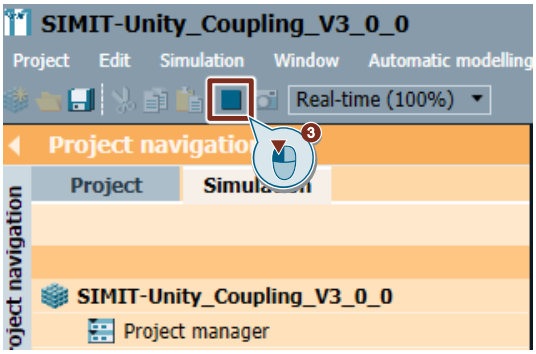
If all couplings and charts in SIMIT are prepared, the simulation can be started in SIMIT.

An orange background indicates that the simulation is running.

NOTE Unity Editor Project is not provided with the application example.

Table 5-3 Starting the SIMIT simulation, connected to a Unity Editor

No.	Action
1.	For the coupling with SIMIT, the “Simit Connector” script must be activated and the “Do Connect” selected. Click the "Start" button to start the simulation in Unity. → Unity is prepared and waiting for SIMIT connection. 
2.	Click the “Start“ button to start the SIMIT simulation. → Connection to Unity Editor Project will be established. → SIMATIC PLCSIM Advanced Instance will start up automatically.  Click “Ok” to accept the connection to an external coupling. 

No.	Action
3.	Click the “Start” symbol again to stop the simulation in Unity. →Unity Editor is disconnecting. 
4.	Click the “Stop” button to stop the simulation in SIMIT. →SIMIT-Unity coupling is disconnecting. →SIMATIC PLCSIM Advanced instance will be closed automatically. 

6 Appendix

6.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form:

support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that’s tailored to the customer’s specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

support.industry.siemens.com/cs/ww/en/sc/2067

6.2 Industry Mall

The Siemens Industry Mall is the platform on which the entire siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:
mall.industry.siemens.com

6.3 Application support

Siemens AG
Digital Industries
Factory Automation
Production Machines
DI FA PMA APC
Frauenauracher Str. 80
91056 Erlangen, Germany

mailto: tech.team.motioncontrol@siemens.com

6.4 Links and literature

Table 6-1 Links and Literature

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109769816
\4\	Link to Unity - Learning https://Unity3d.com/de/learn

6.5 Change documentation

Table 6-2 Change documentation

Version	Date	Modifications
V1.0.0	08/2019	First version
V2.0.0	04/2021	Update: new software versions, Unity project, Unity assets, Documentation
V2.0.1	12/2021	Update: Know-How Protection
V2.1.0	12/2021	Update: New software versions, New External Coupling with isochronous connection type, Unity Asset structure, Know-How Protection, Documentation
V3.0.0	11/2022	Only SIMIT-Unity coupling will be provided from now. Update software versions, projects and documentation