Getting Started with the Parametric Universal Programming and Process Interface (PUPPI)

# What Is PUPPI?

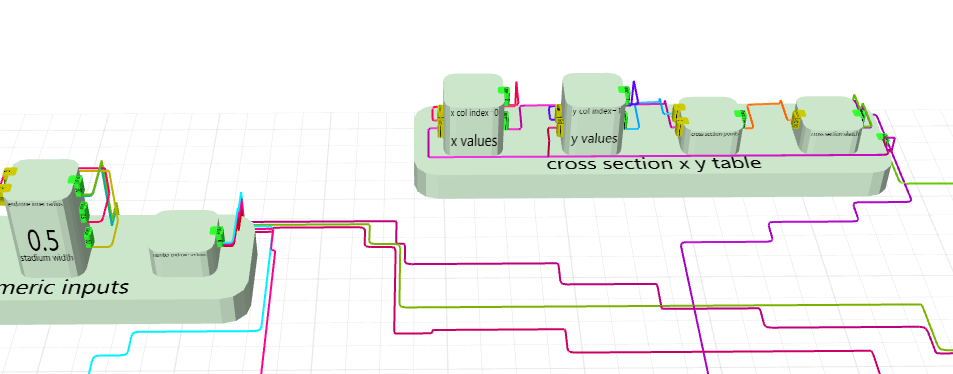
The Parametric Universal Programming & Process Interface (PUPPI) is a library for developing visual programming interfaces. PUPPI features advanced automation routines that allow developers to interface with existing APIs and libraries with minimal coding overhead. New visual modules with custom functions can be added within minimal constraints. PUPPI programs present end users with a revolutionary graphical user interface for creating visual scripts that remain easy to understand even at high levels of complexity. The look and feel of the interface can be altered to provide a customized visual programming experience.

# What can PUPPI do for you?

Parametric visual programming fulfills a need for deeper interaction with software in the design and engineering fields. Imagine having this capability easily added to any software, with minimal coding overhead! I believe that PUPPI, due to its versatility, will gain widespread adoption not only for CAE applications, but in many other fields where there is a need for a visual scripting capability to mount on existing software or to run standalone.

PUPPI delivers an appealing, intuitive and scalable interface. The analogy is that of the program as a city. Logical organization can arise naturally from the act of actually *building* aprogram node by node. Automation features keep the layout neat and easily readable. The 3D interface (2D mode also available) is future-proof as the third dimension becomes widely adopted through products such virtual reality glasses. However, a regular computer will work just as well!

A more intangible benefit of a 3D programming environment is its very tangibility. The structured and solid presence PUPPI code creates a sense of immediacy and of enhanced perception for the user.



For developers, PUPPI presents and easy to use library. Automation features make creating a visually appealing work environment and adding new modules doable with little coding. Many aspects of the environment can be customized. PUPPI developers can easily create PUPPI modules from existing functions and classes.

# Installation

Run the installer and enter the license provided, then click Activate License. When you upgrade from a Trial License to a Full License run the PUPPI License Manager, Relinquish your license, then re-enter it and press Activate License in order to get the license upgraded on your machine.

# Usage

The PUPPI library can be referenced from your Visual Studio .NET project. Browse for the file PUPPI.DLL and load it into the project. The easiest way to create a Visual Programming interface is to create a Windows Form and add the PUPPI elements programmatically as seen below. Refer to the Developer Reference help file available from the Program Files PUPPI menu. In the Tutorials folder, C# and VB Visual Studio example solutions are provided. The Hello PUPPI tutorials are introductory examples showing a basic default canvas with minimal functionality.

# PUPPI Library Structure

The PUPPI library comes packaged in a single DLL. The PUPPI interface uses the Helix 3D Toolkit library and the Windows Presentation Foundation (WPF). The PUPPI library has three main components: The visual programming canvas, the programming model and the menus.

## The PUPPI Programming Canvas

The programming canvas holds the visual program, which is composed of a number of connected nodes. The developer initializes a programming canvas by adding a component to a Windows form or to a WPF control. Knowledge of WPF I not required to develop with PUPPI. The canvas appearance can be customized before initialization, from node appearance to adding custom 3D or 2D non-interacting visuals.

The PUPPIGUIController class contains functions for interacting with the canvas, such as programmatic placement of nodes and other GUI interaction commands.

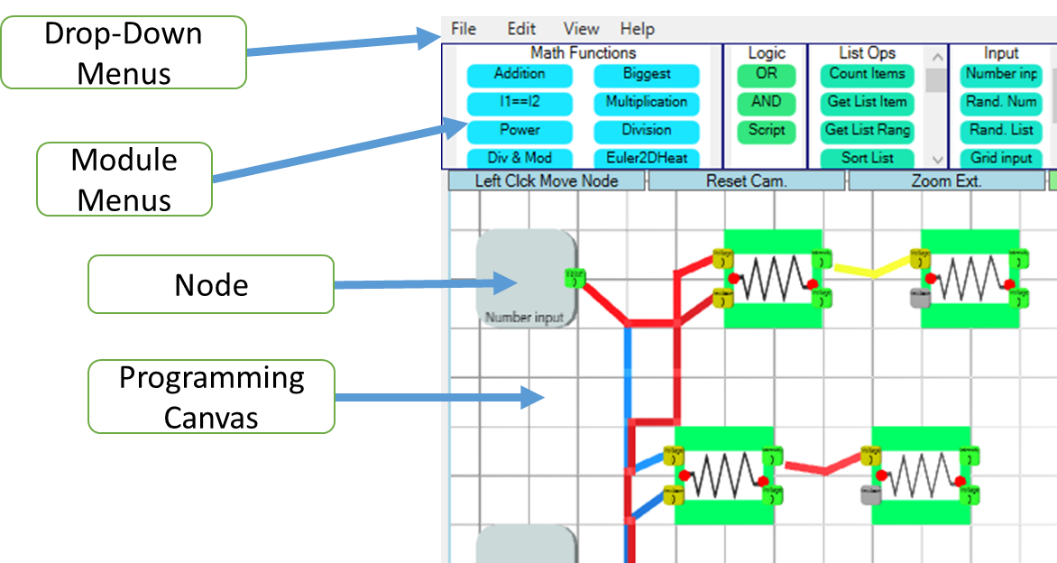


Figure 1: PUPPI GUI Example

### PUPPI GUI Settings

The PUPPI GUI Parameters are set from a static class called PUPPIGUISettings. Settings are defaulted to generate a 3D programming canvas. Colors and node size and many other parameters can be set from here. Some menu parameters can also be set from this class.

### Custom Renderers

Custom renderers can be applied to change the visual appearance of nodes, inputs and outputs and add visuals to the PUPPI Canvas. Custom renderers are applied to modules and the canvas when the canvas and menus are initialized, but can be changed at runtime or by the user through gestures or double clicks for individual nodes. The PUPPINodeCustomRenderer class stores rendering information for nodes with custom visuals. Custom renderers are added to the canvas with unique names and also applied to PUPPIModule objects.

## The PUPPI Programming Model

The logic that operates inside a node is defined in a PUPPI Module. The developer can use existing modules, create custom modules, or convert exiting methods into modules automatically.

The developer can create custom modules by inheriting from the PUPPIModule base class. A class constructor needs to be provide. In the constructor, the user assigns a name to the module, as well as any number of inputs and outputs.

When a visual program is run, an overridable method of a PUPPI Module object, called process\_usercode, can modify outputs based on input values. The sequence of these functions operating on connected modules constitutes the logical arm of the visual program. From the PUPPI Module class, the developer can also define how user’s double clicks and gestures over the node are processed. Please see the included sample projects for more examples of custom modules.

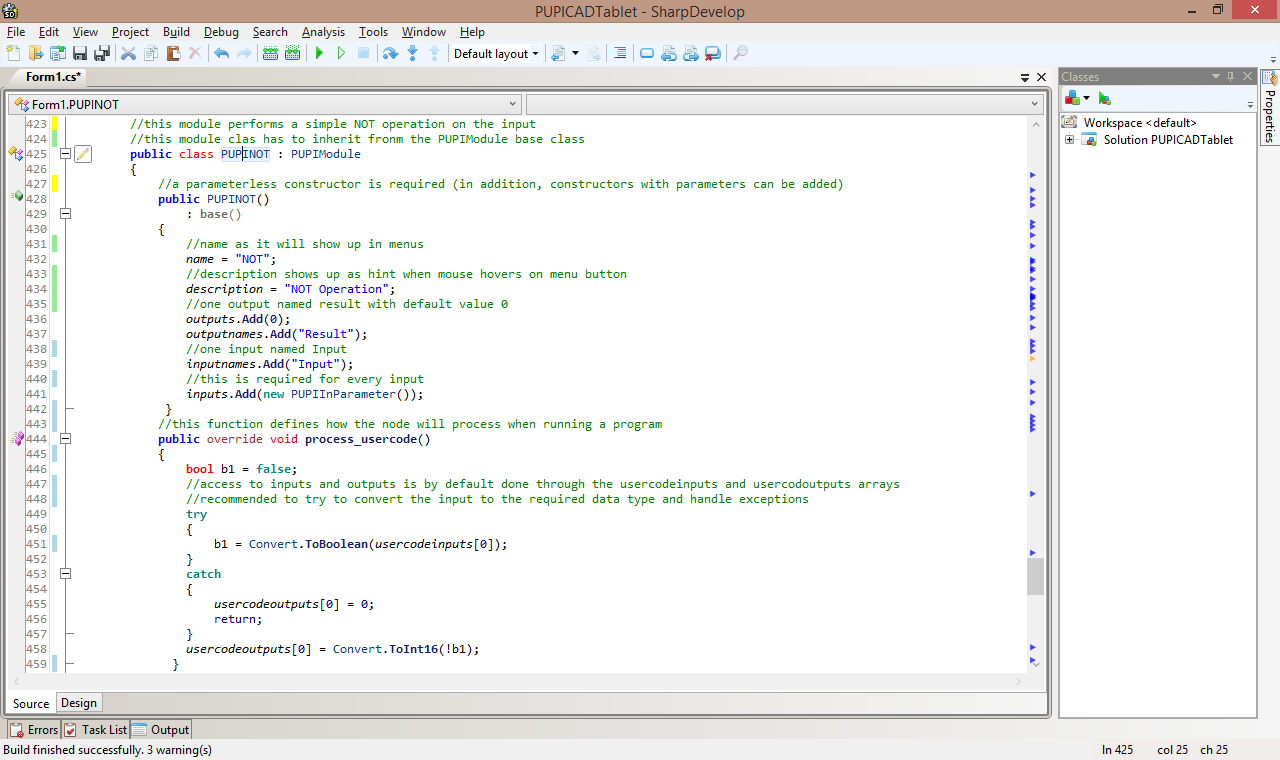


Figure 2: Creating a custom PUPPIModule

## Menus

### Module Keep Menus

Modules are displayed in containers called Module Keeps which can be added anywhere on the form. The developer creates a menu and then adds PUPPI modules to it. The user will drag modules from menus onto the canvas to create nodes. The PUPPIModuleKeepMenu class handles module menu operations.

### Other Menus

File, Edit, View and Help menus can also be added to interact with the programming canvas. The PUPPIDropDownMenu class can be used to create these standard menus and custom dropdown menus.

## Client-Server Architecture

PUPPI applications can run in server mode. A TCP socket server and / or a HTTPListener server can easily be started from the PUPPIServer namespace. Servers accept text commands and can return text or binary data. See below for more information on text commands. Sample web and standalone clients are distributed with the PUPPI toolkit.

## Text Commands

The canvas supports commands sent as scripts. These can be passed from several sources:

-Inside the process\_usercode function of a module, the addCanvasCommand can send a textual command. Commands sent from here are queued to be executed by the canvas during the next available slot.

-From the PUPPIProgramCanvas class, the sendImmediateCommandAsText function sends a command to the canvas to be executed immediately.

-Clients can send commands to the canvas through the server. Commands changing the canvas are queued to be executed by the canvas during the next available slot. Commands requesting information are executed immediately. Commands sent to a TCP Socket Server need to end with password + "<EOF>", as seen in the console client example project.

Below is a list of text commands that can be sent from any of the sources mentioned above:

setinput\_|\_nodeID\_|\_inputIndex

addnodetocanvascommand\_|\_x\_|\_y\_|\_PUPPIModuleName

addnodeonnodecommand\_|\_baseNodeID\_|\_PUPPIModuleName

connectnodescommand\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_destNodeID\_|\_destInputIndex

disconnectnodeinputcommand\_|\_destNodeID\_|\_destInputIndex

disconnectnodeoutputcommand\_|\_sourceNodeID\_|\_sourceOutputIndex

getnodeinformationxml\_|\_nodeID

getcanvaschangedtime

performnodegesture\_|\_nodeID\_|\_startXratio\_|\_startYratio\_|\_startZratio\_|\_endXratio\_|\_endYratio\_|\_endZratio

performnodedoubleclick\_|\_nodeID\_|\_xRatio\_|\_yRatio\_|\_zRatio

runcanvasprogramcommand

forcenodeprocess\_|\_nodeID

exportnoderendererimage\_|\_nodeID

getnoderendererstatexml\_|\_nodeID

getindexedmodulenames

deletecanvasnodebyid\_|\_nodeID

canvasreadablereport

setoutputinteger\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value

setoutputdouble\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value

setoutputstring\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value

setoutputliststring\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value1,value2,etc…

setoutputlistdouble\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value1,value2,etc…

setoutputlistinteger\_|\_sourceNodeID\_|\_sourceOutputIndex\_|\_value1,value2,etc…

The commands below can be executed from clients and receive immediate response:

canvasgetstatus

canvasgetlayout

getnodeinformationxml \_|\_nodeID

getnoderendererimage\_|\_nodeID

getindexedmodulenames

getnodeoutputvalue\_|\_nodeID\_|\_outputIndex

# Packaged PUPPI Utilities

The PUPPI Namespace Explorer can be used to select classes, methods and constructors from a DLL assembly to convert to visual programming modules. Once all the desired objects are selected, they are saved into an MTPS file. PUPPI has functions to load an MTPS file at runtime and generate visual programming modules automatically.

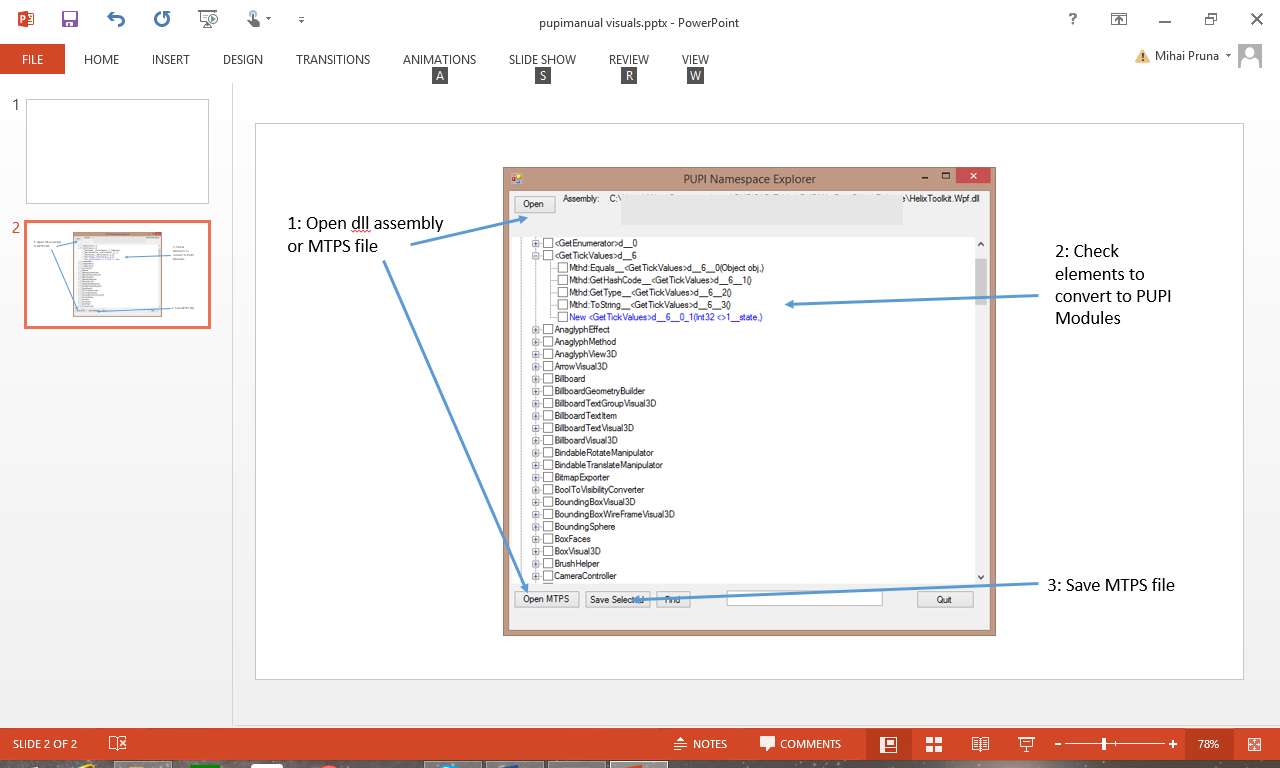


Figure 3: Using the PUPPINamespaceExplorer App

The PUPPI Assembly Creator makes DLL files holding PUPPI modules from MTPS files created with the PUPPI Namespace Explorer. After the list of modules has been finalized it is recommended to convert the MTPS file to PUPPIModule assemblies and load those in your application or plugin since it is much faster than generating modules from an MTPS file at runtime.

# Licenses

## PUPPI

MIT License

Copyright 2021 Mihai Pruna

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Maze Solver (shortest path finder)

http://www.codeproject.com/Articles/9040/Maze-Solver-shortest-path-finder

License:

http://www.codeproject.com/info/cpol10.aspx

## Helix 3D Toolkit

http://helixtoolkit.codeplex.com/

License:

http://helixtoolkit.codeplex.com/license

Windows Presentation Foundation (WPF) is a registered trademark of Microsoft Corporation.