Code de création du maillage dans Tekla Structures 21.1

```
using System;
using System.Collections.Generic;

using Grasshopper.Kernel;
using Rhino.Geometry;

using Tekla.Structures;
using Tekla.Structures.Model;
using Tekla.Structures.Model.Operations;

namespace EIFFAGE_ADD_ON
{
    public class TeklaGrid : GH_Component
    {
        private Model myModel;

        /// <summary>
        /// Delete all the existing grids from a Tekla Model
        /// </summary>
        private void DeleteExistingGrids()
        {
            Grid objGrid = null;
                foreach (ModelObject obj in
myModel.GetModelObjectSelector().GetAllObjects())
                {
                    objGrid = obj as Grid;

                    if (!(objGrid == null))
                    {
                        objGrid.Delete();
                    }
                }
        }

        /// <summary>
        /// Convert a list of coordinates into a string of pitches
        /// </summary>
        /// <param name="coords"></param>
        /// <returns></returns>
        private string ConvertCoords(List<int> coords)
        {
            string strReturned = coords[0].ToString();
            int pitch = 0;

            for (int i = 1; i < coords.Count; i++)
            {
                pitch = coords[i] - coords[i - 1];
                strReturned = strReturned + " " + pitch.ToString();
            }

            return strReturned;
        }

        /// <summary>
        /// Convert a list of string into a string with space separator
        /// </summary>
        /// <param name="strList"></param>
        /// <returns></returns>
        private string ConvertListString(List<string> strList)
        {
            string strReturned = strList[0];
```

```
            for (int i = 1; i < strList.Count; i++)
            {
                strReturned = strReturned + " " + strList[i];
            }

            return strReturned;
        }



        /// <summary>
        /// Initializes a new instance of the TeklaConnector class.
        /// </summary>
        public TeklaGrid()
          : base("TeklaGrid", "TKLGrid",
              "Create new grids on Tekla",
              "EIFFAGE", "Tekla")
        {
        }

        /// <summary>
        /// Registers all the input parameters for this component.
        /// </summary>
        protected override void RegisterInputParams(GH_Component.GH_InputParamManager
pManager)
        {
            pManager.AddBooleanParameter("OnOff", "IO", "Boolean to send the
connections", GH_ParamAccess.item, false);
            pManager.AddBooleanParameter("New Grid", "N", "Delete the old grids if
true (recommended). Simply add a new grid above the old ones if false.",
GH_ParamAccess.item,true);
            pManager.AddIntegerParameter("X Coordinates", "X", "X positions of the
grid", GH_ParamAccess.list);
            pManager.AddIntegerParameter("Y Coordinates", "Y", "Y positions of the
grid", GH_ParamAccess.list);
            pManager.AddIntegerParameter("Z Coordinates", "Z", "Z position of the
grid", GH_ParamAccess.list);
            pManager.AddTextParameter("X Labels", "Lx", "X labels",
GH_ParamAccess.list);
            pManager.AddTextParameter("Y Labels", "Ly", "Y Labels",
GH_ParamAccess.list);
            pManager.AddTextParameter("Z Labels", "Lz", "Z Labels",
GH_ParamAccess.list);
        }

        /// <summary>
        /// Registers all the output parameters for this component.
        /// </summary>
        protected override void
RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
        {
        }

        /// <summary>
        /// This is the method that actually does the work.
        /// </summary>
        /// <param name="DA">The DA object is used to retrieve from inputs and store
in outputs.</param>
        protected override void SolveInstance(IGH_DataAccess DA)
        {
            //Variables
```

```
            bool onOff = false;
            bool newGrid = true;
            List<int> xCoords = new List<int>();
            List<int> yCoords = new List<int>();
            List<int> zCoords = new List<int>();
            List<string> xLabels = new List<string>();
            List<string> yLabels = new List<string>();
            List<string> zLabels = new List<string>();



            myModel = new Model();

            //Getters
            if (!DA.GetData(0, ref onOff)) { return; }
            if (!DA.GetData(1, ref newGrid)) { return; }
            if (!DA.GetDataList(2, xCoords)) { return; }
            if (!DA.GetDataList(3, yCoords)) { return; }
            if (!DA.GetDataList(4, zCoords)) { return; }
            if (!DA.GetDataList(5, xLabels)) { return; }
            if (!DA.GetDataList(6, yLabels)) { return; }
            if (!DA.GetDataList(7, zLabels)) { return; }



            //Errors handling
            if (!myModel.GetConnectionStatus())
            {
                AddRuntimeMessage(GH_RuntimeMessageLevel.Warning, "Unable to establish
a connection with Tekla. Please start Tekla before using this component!");
                return;
            }

            if (xCoords.Count != xLabels.Count | yCoords.Count != yLabels.Count |
zCoords.Count != zLabels.Count)
            {
                AddRuntimeMessage(GH_RuntimeMessageLevel.Warning, "One or more of your
labels list doesn't have the same length as the coordinate list");
                return;
            }


            //Work
            string strXCoords = ConvertCoords(xCoords);
            string strYCoords = ConvertCoords(yCoords);
            string strZCoords = ConvertCoords(zCoords);
            string strXLab = ConvertListString(xLabels);
            string strYLab = ConvertListString(yLabels);
            string strZLab = ConvertListString(zLabels);


            if (onOff & newGrid)
            {
                DeleteExistingGrids();
            }

            if (onOff)
            {
                Grid grid = new Grid();

                grid.CoordinateX = strXCoords;
                grid.LabelX = strXLab;
```

```
                grid.CoordinateY = strYCoords;
                grid.LabelY = strYLab;
                grid.CoordinateZ = strZCoords;
                grid.LabelZ = strZLab;

                grid.Insert();

                myModel.CommitChanges();
            }

        }

        /// <summary>
        /// Provides an Icon for the component.
        /// </summary>
        protected override System.Drawing.Bitmap Icon
        {
            get
            {
                return Properties.Resources.TKLGridIcon;
            }
        }

        /// <summary>
        /// Gets the unique ID for this component. Do not change this ID after
release.
        /// </summary>
        public override Guid ComponentGuid
        {
            get { return new Guid("{3e151b1f-64b2-4501-879b-ece3790f62b2}"); }
        }


        /// <summary>
        /// Gets the position of the component into the Subcategory
        /// </summary>
        public override GH_Exposure Exposure
        {
            get
            {
                return GH_Exposure.quarternary;
            }
        }
    }
}
```