applied sciences

MDPI

Article

Visual Programming as Modern and Effective Structural Design Technology—Analysis of Opportunities, Challenges, and Future Developments Based on the Use of Dynamo

Paweł Grzegorz Kossakowski

Faculty of Civil Engineering and Architecture, Kielce University of Technology, 25-314 Kielce, Poland; kossak@tu.kielce.pl

Featured Application: The structural design is the main application of the visual programming discussed in the paper.

Abstract: This study is dedicated to technology based on visual programming as a modern tool to support the design of building structures. It provides a brief background describing the computerization of design and discusses design-aided systems and the concept and tools of visual programming. The principles of the Visual Programming Language (VPL) are presented in detail. The programming and design environments used in the architecture and construction industry are presented. Dynamo, one of the VPL-based programs applicable to structural design, was used in the study. In order to best demonstrate and explore the capabilities of this environment, examples of the practical application of visual programming in structural design are presented. Both simple and more complex structures have been designed and discussed in detail. The integration of Dynamo with computational systems is also presented. A broad discussion was held on the possibilities of using visual programming in structural design, the problems and challenges, and the directions of its development in the architecture and construction industry.

Keywords: visual programming; VPL; BIM; structural design

Citation: Kossakowski, P.G. Visual Programming as Modern and Effective Structural Design Technology—Analysis of Opportunities, Challenges, and Future Developments Based on the Use of Dynamo. *Appl. Sci.* **2023**, *13*, 9298. https://doi.org/10.3390/app13169298

Academic Editor: Jürgen Reichardt

Received: 7 June 2023
Revised: 4 August 2023
Accepted: 10 August 2023
Published: 16 August 2023

1. Introduction

The dynamic development of technology since the end of the 18th century has led to the creation and implementation of many different and increasingly sophisticated devices and technologies. In the broad field of engineering, the development, construction, and, above all, the general use of computers have been of fundamental importance [1].

Computerization has, of course, also entered the broad field of engineering design. Furthermore, in addition to its fundamental application in architectural and construction design, there is a notable focus on utilizing diverse methods and models to manage the design process, as well as the implementation of various construction projects, even under challenging conditions [2–5].

The basic issue here was the possibility of programming, which made it possible to create advanced procedures for carrying out simulations of the processes being analyzed. Another extremely important factor was the ability to perform complex and time-consuming calculations.

Engineers directly involved in production are particularly interested in the Computer-Aided Design (CAD) system because it provides tools for automating a range of processes and activities. In short, the CAD idea is based on the concept of digital representation of the project being created [6]. The fundamental is digital geometric modeling based on elementary objects such as points, lines, curves, etc. Each of the elements representing the designed product is made up of the aforementioned geometric components and is defined graphically, specific to a given program. The designed structural element actually contains

three basic pieces of information regarding its geometric, dynamic, and material or broader technological characteristics.

However, for some time now, the CAD concept based on 2D and 3D graphics has been somewhat burnt out, as the programs have reached their maximum level of functionality and it is difficult to expect any innovation in this area.

The new idea was to find a way of defining an element that would be simple, fast, and accurate on the one hand and would achieve broad universality in terms of the types of elements modeled on the other [7]. It was then devised that the individual components of the building object would be modeled directly in the form of three-dimensional objects corresponding to the geometry of the given element, thus having the appropriate cross-section, length, and characteristic geometric attributes (e.g., holes, cuts, etc.) and being defined in a simplified manner. This simplification involves defining characteristic points of the modeled object in the model's workspace, such as the start and end, and loading a finished object from a database that is a model of the element being designed. This model also contains a set of information defining parameters of the element other than geometric ones, which is the first innovative functionality of the new idea described above [8]. This new modeling concept took the form of Building Information Modelling (BIM) technology, which is referred to as modeling information about the building, although it actually includes modeling the entire construction investment [9–11].

The widespread use of computer technology to support the design process has not only led to the creation of the systems presented above. An important and very interesting path is the development of many specific design methods based on the use of computerization. Furthermore, it is necessary to distinguish architectural ideas and concepts that are related and based on the use of these systems. Several types of computer-aided design can be distinguished, based on the use of computers and numerical methods. The main concepts and types relevant to this study are as follows:

- Computational Design, which covers the design process in its broadest sense, including both the use of the computing power of equipment and methods and the design of products and simulation processes based on computer programs.
- Algorithmic Design, where the design process is carried out using specially prepared computer algorithms. The essence here is the ability to perform operations on input datasets of a wide and diverse nature in order to achieve a specific effect, e.g., in the form of the geometry of the designed object.
- Parametric Design, where the design process or the designed product is described by parameterized rules. The designer has the ability to flexibly manipulate the parameters to achieve a specific goal.
- Generative Design, which is a kind of derivative and the next stage in the evolution of the methods described above. By using intelligent numerical algorithms, it is possible to achieve multi-variability, leading to the optimization of the designed product. This process is based on elements such as initial conditions and parameters defining the task, boundary conditions, a control mechanism (algorithm), and the selection of the optimal solution obtained on the range of prepared (generated) variants.

At this point, it should be noted that there are many ways to search for sources of shapes of designed objects, and consequently, to find algorithms describing them.

Computational approaches to building design are highly relevant and have been extensively researched in the field of architecture. The most common applications include parametric design [12–14] and algorithmic-parametric design [15,16]. Some approaches explicitly incorporate a mathematical framework to design architecture [17], which can be easily programmed [18]. In addition, the digitization of fabrication processes in architecture, engineering, and construction [19] is an interesting avenue for exploration.

The construction industry has been an early adopter of computerization, which for a long time mainly covered the design phase but now encompasses the entire process of construction investment implementation. The fundamental issue in the types of design described above is the use of a dynamic description to flexibly control the parameters of

the designed object. The best way to achieve this is through a mathematical description. This works particularly well for describing the geometry of the designed structure. To achieve this, the geometry should be represented using mathematical functions. In addition to standard and already-established design support systems, an interesting tool in this area is the use of programming environments. Unfortunately, their use by designers is still marginal, mainly due to the barrier that programming itself represents for a civil engineer. From an engineering point of view, the fundamental problem is to use a suitable programming environment in which such a procedure can be created. The use of classical programming languages such as C++ is rather ineffective in this case. A different approach based on visual programming is currently being used.

Visual code-based programming addresses this issue, as it does not require the designer to have the specialist knowledge necessary to create traditional text scripts. In addition, visual programming environments used in the architecture and construction industry are equipped with a number of options and tools that significantly automate, simplify, and speed up the design process and even investment management.

Today, visual programming is gaining significant traction in various sectors of the construction industry, particularly in design. Notable examples include its application to structural design [20], infrastructure projects [21], and the assessment of the structural integrity of historic buildings [22]. However, particularly intriguing work involves the use of VPL for the optimization of building structures based on genetic algorithms [23] and the planning of construction site layouts, particularly through the use of generative design techniques [24].

To date, there has been considerable documentation and discourse surrounding VPL technology and computer-aided design within the broader design and architecture literature. In contrast, concise and comprehensive publications that focus specifically on the application of visual programming in structural design, similar to those available for BIM, for example, are still rare. Few comprehensive textbooks deal exclusively with the design of structures using VPL tools. It is clear that such publications are needed in this particular area.

In this paper, the concept of Visual Programming Language (VPL) is thoroughly examined, with a comprehensive exploration of the programming and design environments employed in the field of architecture and construction. This study focuses specifically on Dynamo, a VPL-based design platform. The topic discussed in this paper is closely related to algorithmic-parametric design, specifically the use of existing programming environments in engineering design. To effectively showcase the capabilities of this environment, practical instances of visual programming's application in the design of both simple and intricate structures, as well as its integration with computational systems, have been meticulously prepared and extensively discussed. In addition, an extensive discourse has taken place regarding the potential applications of visual programming in structural design, along with the associated challenges and obstacles, as well as the future directions of its advancement within the architecture and construction industry.

Based on the Dynamo application, this study is devoted to exploring the utilization of visual programming as a contemporary tool to enhance the design process of building structures. It attempts to provide a reasonably broad overview of the current possibilities for the application of VPL in structural design, along with a critical analysis of related problems. It serves to complement existing works in this field, providing insight into the current practical applications of VPL technology in structural design. Moreover, it delves into various theoretical aspects pertaining to its utilization.

2. Methodology

Generally, the study is based on the Dynamo visual programming environment, which is used in the design industry to date. The Dynamo system uses the DesignScript VPL-class language. Due to the development of this program, it has been implemented and integrated with other environments for quite some time. Dynamo has been available in three versions, as a standalone Dynamo Sandbox environment, as an Autodesk product called Dynamo

Studio (currently not developed), and as a plug-in to the Revit design environment. This makes it possible to use it in different areas and stages of design. Dynamo is used for calculations, creating shapes and geometry of building objects, managing BIM information, and creating technical documentation, including detailing structural elements for design-build projects. Dynamo's visual coding system is based on two anatomical elements, nodes and wires, such as other VPL-class environments and languages.

The methodology of the research includes the presentation and analysis of examples of the practical application of the VPL system using scripts prepared in Dynamo, as well as a critical analysis of various aspects of the implementation of visual programming in structural design.

In particular, the general concept of creating visual scripts is presented, as well as examples of their practical application. These will allow a detailed analysis of the VPL-based design process, to evaluate its evolution in the creation of increasingly complex structural forms, and learn about the possibilities of integrating the modeling environment with computational programs. In the following section, research on current possibilities for the application of visual programming in structural design in its broadest sense is given, together with an analysis of the problems, challenges, and likely directions of development of this technology.

3. Concept of Visual Programming

The idea of visual programming is based on solutions that rely on non-textual programming but use intelligent graphical elements with certain capabilities. This is expressed in the ability of graphical objects to perform certain operations on datasets—the procedures and commands for processing the data are encoded in graphical blocks. These blocks must form a larger whole, a population of blocks when given a larger task to perform. This is the general idea behind programming based on the so-called Visual Programming Languages (VPL). This is simply a large group of programming languages based on intelligent blocks capable of processing data that can be linked together to transfer the data for further processing. As a result of this connection, the user creates an algorithm that forms a graphical code that is a fully functional program. The only difference is that in classical programming, the programmer writes a script, whereas in visual programming, a structure of interconnected graphical objects is created that reflects the syntax elements of a given programming language. The graphical code that works in VPL languages is also known as a visual script.

As mentioned above, the general idea of VPL languages is to replace textual syntax with graphical objects in which individual commands or contained data are encoded and create graphical connections between them. Therefore, the basic elements of a VPL language and programming environment are command blocks called nodes and connectors called wires. They take a suitable form, such as nodes and wires, which create the structure of the script and are responsible for data processing and flow. Figure 1 shows an example of a visual script prepared in the Dynamo Sandbox environment.

As we can see, creating the above program in a visual way is very simple, as it is enough to connect a few nodes to process and flow data and obtain the desired result.

Of course, VPL languages reflect textual syntax, which could be prepared in the traditional way by simply writing a script. However, this is often a tedious process as the visual script is designed to be simple, clear, and fast, whereas the textual syntax of a given VPL language can be very complicated. An example of this is the miniscript in the Dynamo environment, where the task is to calculate the square root of the number 4. Figure 2a shows the visual script, while Figure 2b shows a fragment of the script written in the DesignScript language, which is the programming engine of the Dynamo environment.

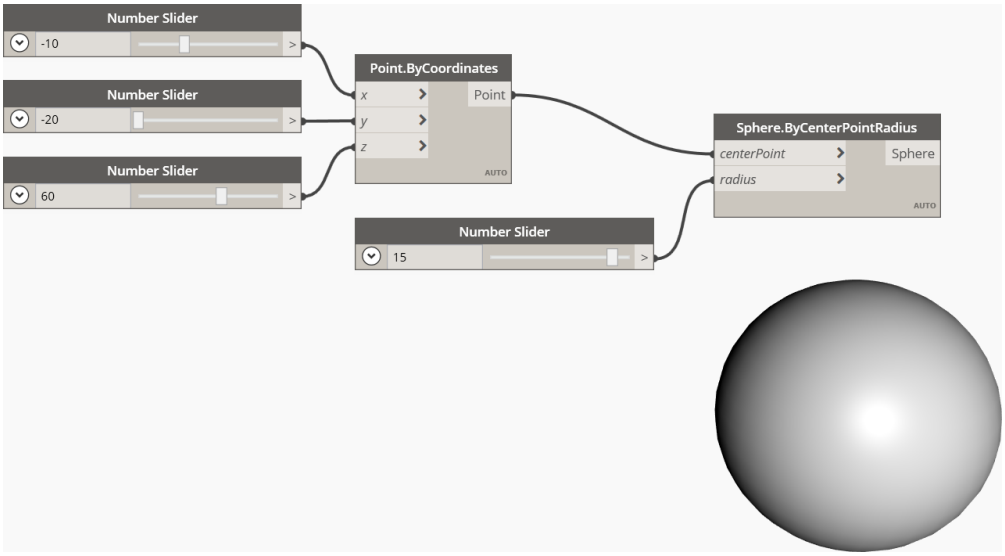
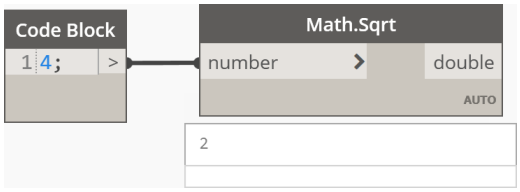


Figure 1. Structure of a VPL code in Dynamo environment.



(**a**)


```
{
  "Uuid": "26c4250e-94ed-437e-b9a5-5312212c0350",
  "IsCustomNode": false,
  "Description": null,
  "Name": "Fig_02",
  "ElementResolver": {
    "ResolutionMap": {}
  },
  "Inputs": [],
  "Outputs": [],
  "Nodes": [
    {
      "ConcreteType": "Dynamo.Graph.Nodes.ZeroTouch.DSFunction, DynamoCore",
      "NodeType": "FunctionNode",
      "FunctionSignature": "DSCore.Math.Sqrt@double",
      "Id": "018061aee05c44229967826c27fd74d5",
      "Inputs": [
        {
          "Id": "adf3df5783f049cc961ad786d6293831",
          "Name": "number",
          "Description": "Liczba w zakresie [0, ∞).\n\ndouble",
          "UsingDefaultValue": false,
          "Level": 2,
          "UseLevels": false,
          "KeepListStructure": false
        }
      ],
      "Outputs": [
        {
          "Id": "707cf847993e44b09c3ca82af61e7c66",
          "Name": "double",
          "Description": "Dodatni pierwiastek kwadratowy z liczby.",
          "UsingDefaultValue": false,
          "Level": 2,
          "UseLevels": false,
          "KeepListStructure": false
        }
      ],
```


(**b**)

Figure 2. The square root of the calculation of number 4: (**a**) Dynamo visual script; (**b**) fragment of the DesignScript text script.

At this point, it would be necessary to define what VPL class languages are. According to the definition given in the FolDoc online dictionary, "a Visual Programming Language (VPL) is any programming language that allows the user to specify a program in a two- (or more) dimensional way" [25]. Therefore, a VPL is any programming language that allows the user to define a program in two or more dimensions. In the case of conventional programming languages, there is a collision with one-dimensional processing since compilers or interpreters process one-dimensional character streams.

An important feature in the classification of VPLs is their visual expression, including graphical elements that constitute their 'visual syntax', such as icons, forms, diagrams, etc.

It should also be noted that some languages that use graphical environments are not VPLs because they only use a graphical GUI interface to facilitate programming.

Visual programming languages have been developed and implemented for quite some time, as their history goes back several decades. Their development, or perhaps more accurately their popularization, is to some extent related to the development of graphical environments, which were closely related to the development of computer systems, including the development of Windows systems. Visual programming languages have been implemented in many different areas. They can be divided into groups that include the following applications:

- Education.
- Multimedia.
- Video games.
- Systems/simulations.
- Automation.
- Data warehousing/business intelligence.
- Legacy.
- Miscellaneous.

Currently, there are several visual programming environments whose main area of activity is embedded in the broadly understood computer modeling of building objects. Their scope is very broad, covering basic design-related industries such as architecture, construction, and systems, but due to the openness of the philosophy adopted, there are no restrictions on applying them to other more specialized industries or specialties. This is gradually being implemented as visual programming is applied in other areas. It also applies to other stages of construction investment not related to design, such as the implementation or operating phases. The most important VPL design environments are Rhinoceros-Grasshopper, Dynamo, Vectorworks-Marionette, and GenerativeComponents.

4. Structural Design Based on VPL

At present, there are several visual programming environments whose basic area of operation is embedded in the broadly understood computer modeling of construction objects. Their scope is very wide, as it includes the basic industries related to design, namely architecture, construction, and building systems. The subject of this paper is related to structural design, and information is presented on the visual programming-based system Dynamo that is currently in use in this area.

The possibilities to design different types of building structures in the VPL Dynamo Sandbox environment are presented below. They focus on form generation, primarily. Other possibilities, such as VPL-based structural analyses, are presented last.

4.1. Basic Structures—Flat Truss

The simplest type of building structure that can be used to illustrate the general concept of geometry modeling in the Dynamo environment is a flat frame. It is still one of the most basic and popular structural systems, used for example in industrial construction to form hall objects. The supporting structure here consists of transverse systems of flat frames composed of columns and beams.

A more complex type of structure than a frame, both geometrically and statically, is the flat truss. It is also one of the most basic and popular structural elements used in construction. Due to their more efficient statics compared to frames, trusses are used to form roofs of objects with larger spans. A truss system is also one of the basic types of bridge and viaduct structures. The standard truss consists of top and bottom chords, diagonals, and, optionally, verticals (Figure 3).

Figure 3. Construction of a hall with a truss system [26].

The geometry of a truss is more complex than that of a flat frame because of the greater number of components and their different and alternative placement and arrangement. The general idea of creating a visual script to generate a truss is that, as in the case of a flat frame, the component elements of the truss are spanned on a grid of characteristic points. To simplify the problem, the left half of the element is generated and then mirrored relative to the plane of symmetry. In order to achieve flexibility in modeling the geometry of the truss, including its types, a dynamic definition of the feature points and a number of fields has been applied using Number Slider nodes, as well as settings of truss diagonals based on logical True/False functions. Due to the specific nature of the truss geometry, the first step is to define the crucial points that define the lower and upper chords (Figure 4). This will enable the geometry of the entire truss to be changed dynamically, allowing it to be modified flexibly.

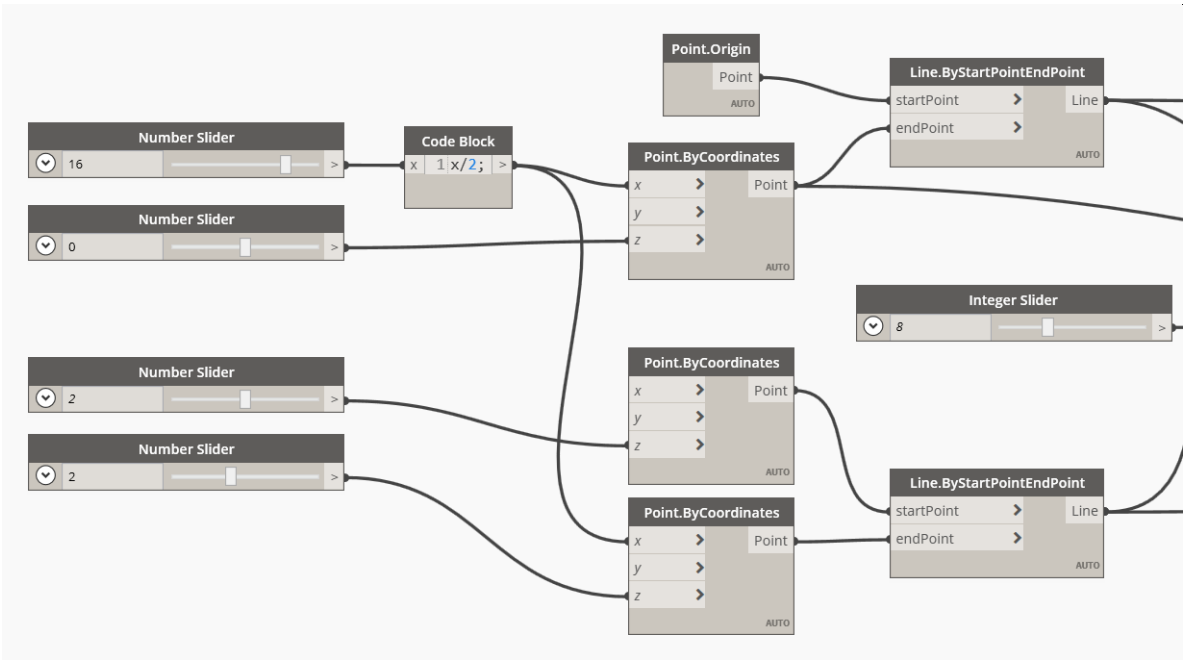


Figure 4. Definition of crucial points for a flat truss.

The points that define the arrangement of the diagonals and verticals were generated using the Curve.PointAtParameter node, which allows this operation to be performed on a curve according to specified rules. In this script, the syntax used to define the points on the upper and lower chords has been used, which also defines the division of the truss into a certain number of fields using the range definition in the form of 0..1..#(n + 2)/2 (Figure 5).

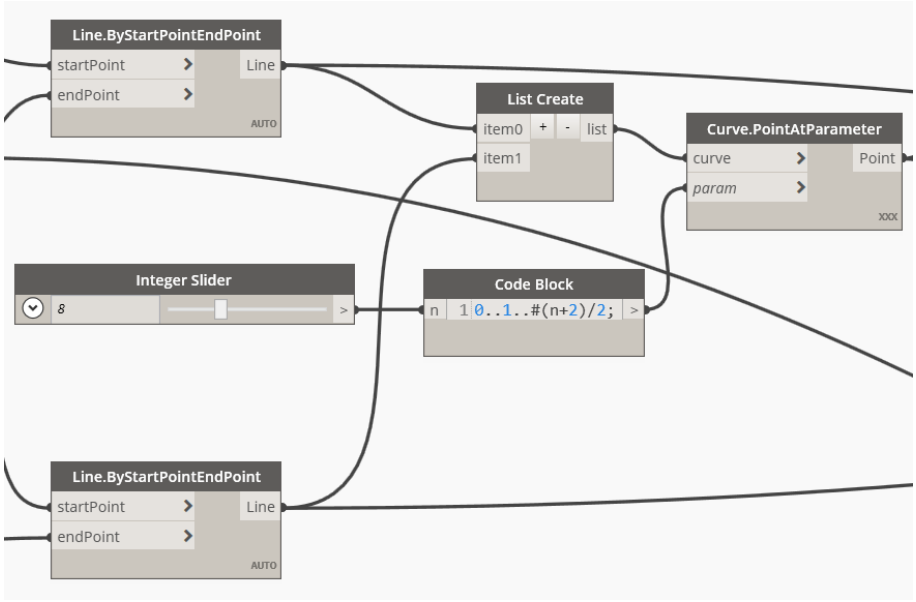


Figure 5. Discretization of chords and definition of verticals.

The next step was to create verticals, which are lines connecting opposite points created on the chords using the procedure described above. Next, diagonals were defined, which, similar to verticals, are lines connecting points on the chords, with the difference that they are not opposite points but shifted relative to each other. To do this, new point lists were created using nodes of type List.RestOfItem. In addition, the True/False logical operator was used to select the tilt of the crosses to the left or right.

The prepared half of the truss was subjected to a mirror reflexion operation. As a result, it was possible to create a flat truss with parallel chords, as shown in Figure 6.

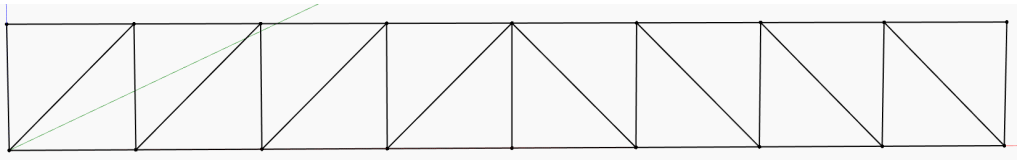


Figure 6. A truss with parallel chords.

The functionality of the script allows for easy generation, even in real time, of other typical truss geometries. By increasing the height, for example, we can generate a vaulted parallel chord truss (Figure 7).

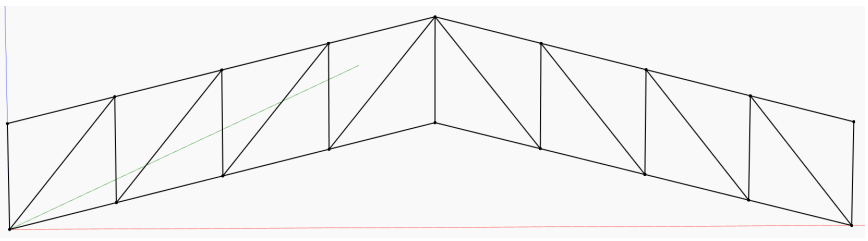


Figure 7. Vaulted truss with parallel chords.

Another possibility is to model the double howe truss, as shown in Figure 8.

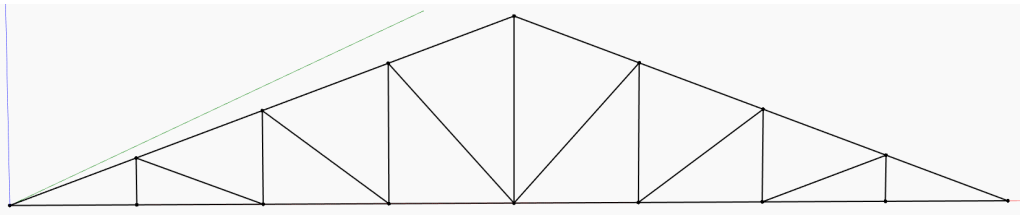


Figure 8. Double howe truss.

As can be seen from the examples presented, a single script allows a variety of roof truss types and geometries to be prepared and analyzed. This approach is much faster and more efficient than the traditional preparation of this in CAD/CAM environments.

4.2. Complex Structures—Space Truss

A further development of the flat truss concept is the space truss. It is used when it is necessary to cover objects with significant spans or for architectural reasons (Figure 9). Various geometric solutions and arrangements are used in the formation of space trusses, characterized by a specific arrangement of the elements of a single module, from which the given structure is composed. In principle, these are spatial chords and diagonals.

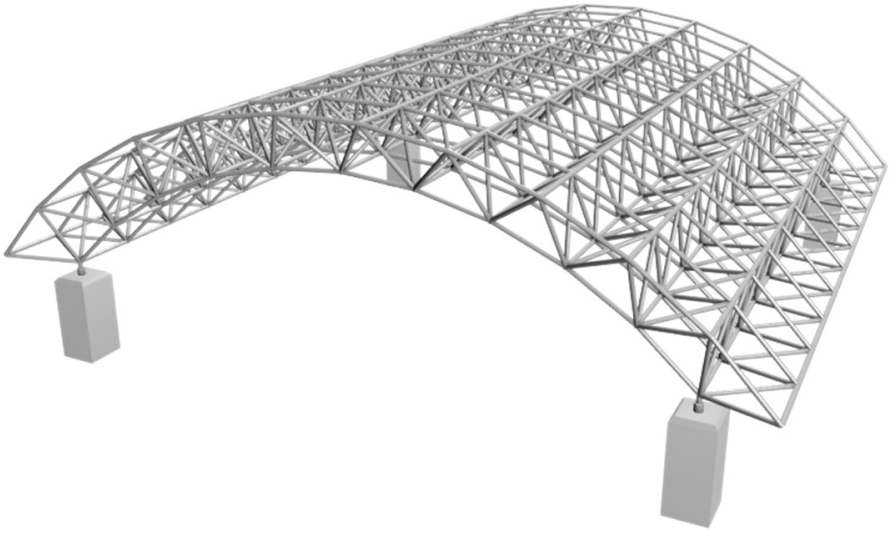


Figure 9. Space truss [27].

An example of a highly efficient script used to create a structurally complex roof is shown below. Note the use of the external LunchBox library, which provides many nodes that significantly shorten and automate the designer's work.

The first step was to prepare a surface using the Surface.ByLoft node, this time defined by two identical boundary elements—arcs (Figure 10).

The WireSpaceTruss node (Figure 11), available in the aforementioned LunchBox library installed in the standard Dynamo environment, was used to define the structural shell. It allows the automatic generation of a spatial truss system in the form of regular pyramidal trusses with a square base. The user can control the division of the structure using the U and V parameters, as well as its height. Importantly, by relying on the structural design, specifically the code of the WireSpaceTruss node on the surface, it is possible to model roofs with a high degree of curvature and waving in a very flexible manner.

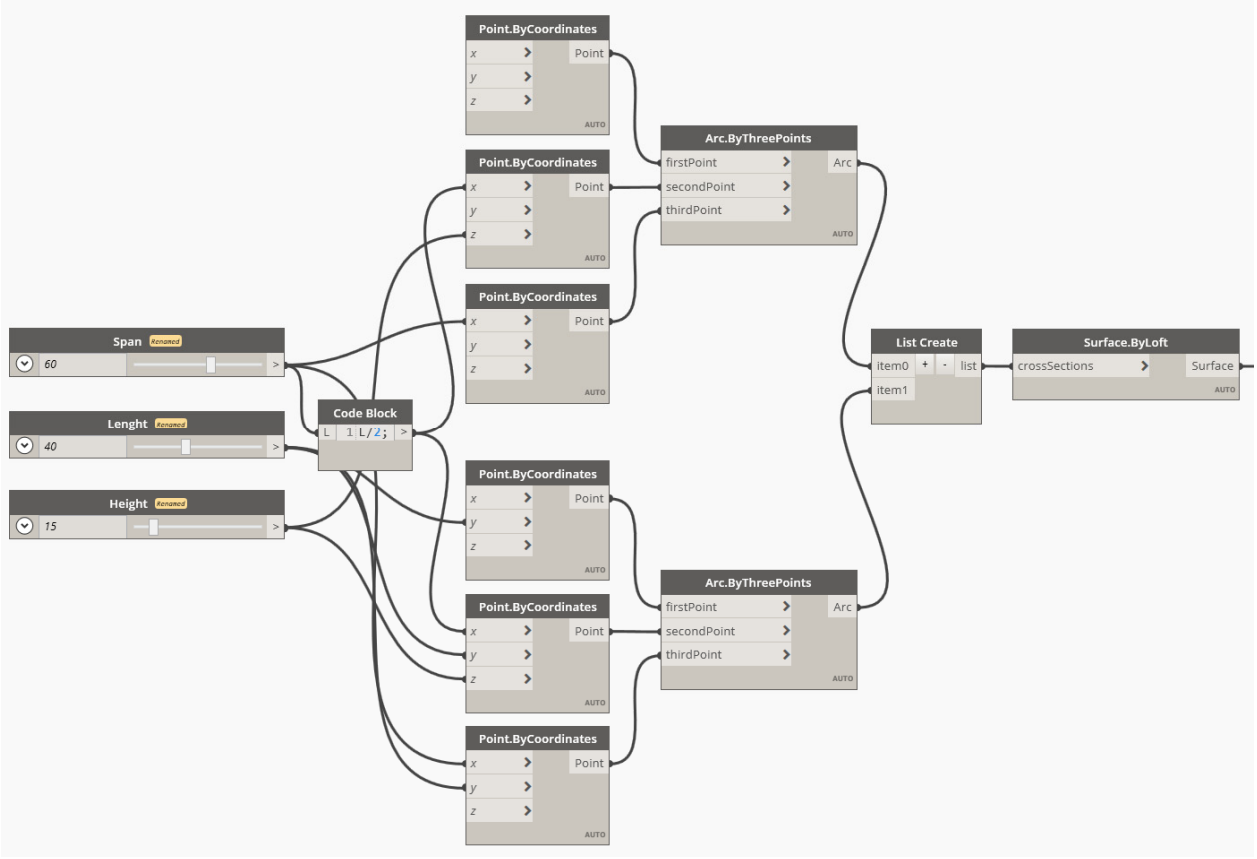


Figure 10. Definition of points and arcs that describe the surface of a space truss.

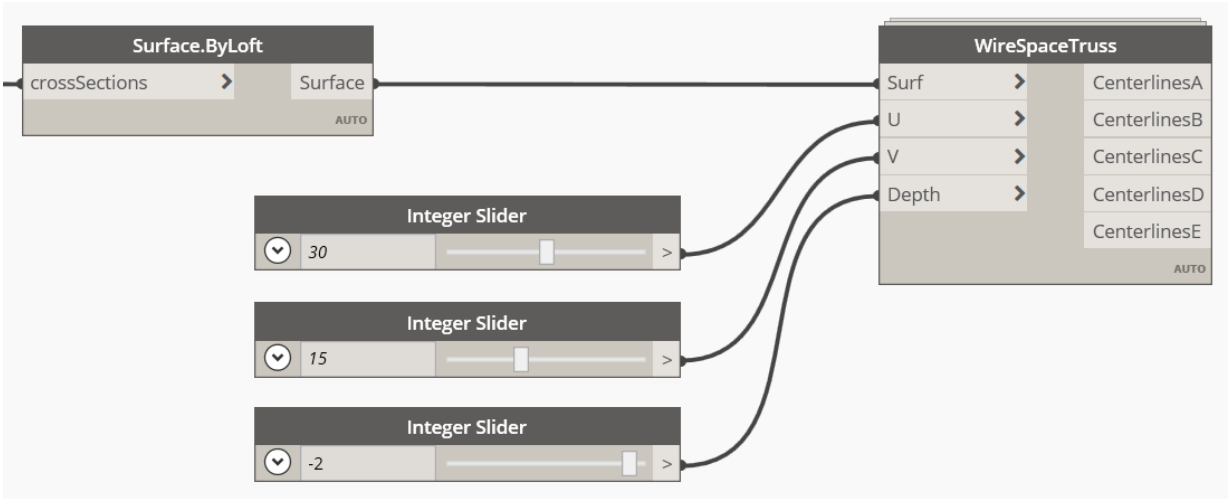


Figure 11. Definition of the geometry of a space truss.

Finally, the structural roof was generated in the form of a space truss shown in Figure 12.

The shape and form of the roof structure, shown in Figure 13, is relatively simple. A key benefit of using a parametric approach is that the geometry of this type of object can be described using curves and arbitrary B-spline surfaces. This is demonstrated below with an example of the use of B-spline curves only. In this case, the definition of bounding curves has been changed from curves to B-splines (Figure 13).

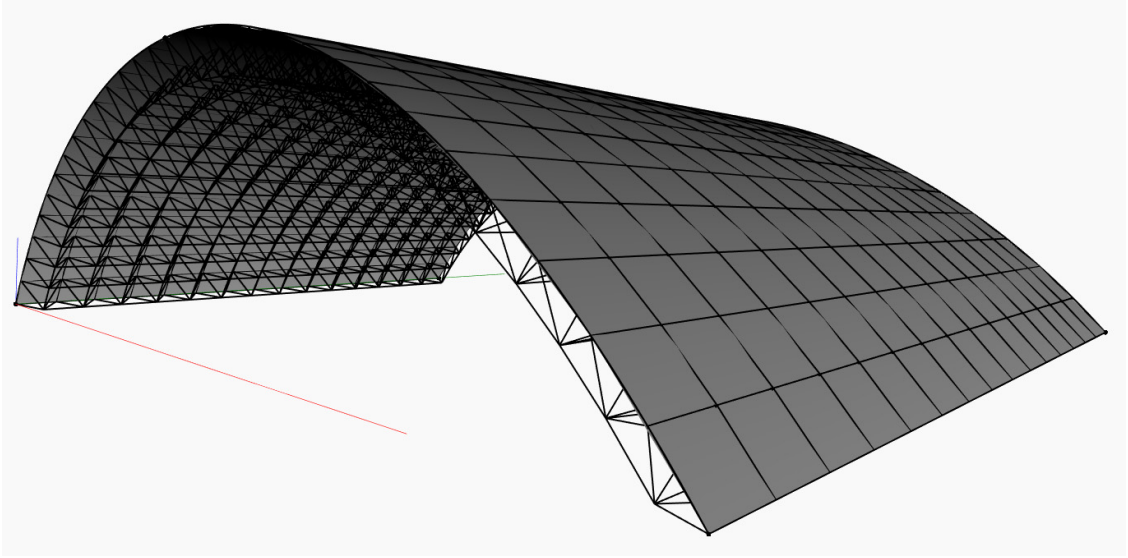


Figure 12. Space truss structural roof.

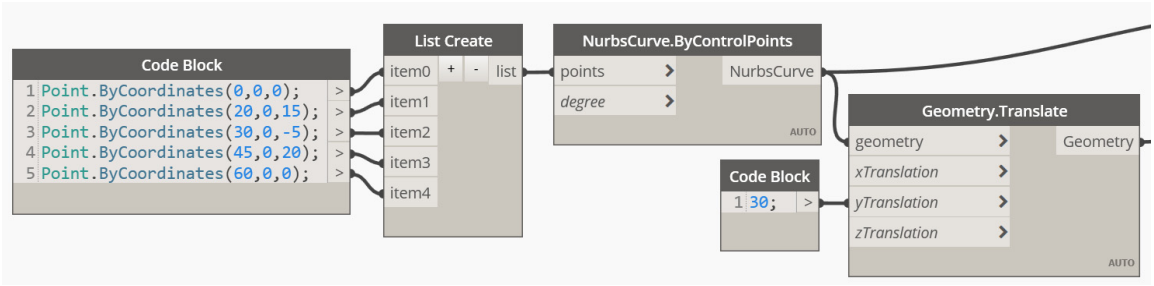


Figure 13. Definition of the geometry of a structural roof using B-splines.

This allowed free curves to be modeled in a free form. Thanks to the control points, it is possible to create a strongly wavy surface, resulting in the structural roof shown in Figure 14.

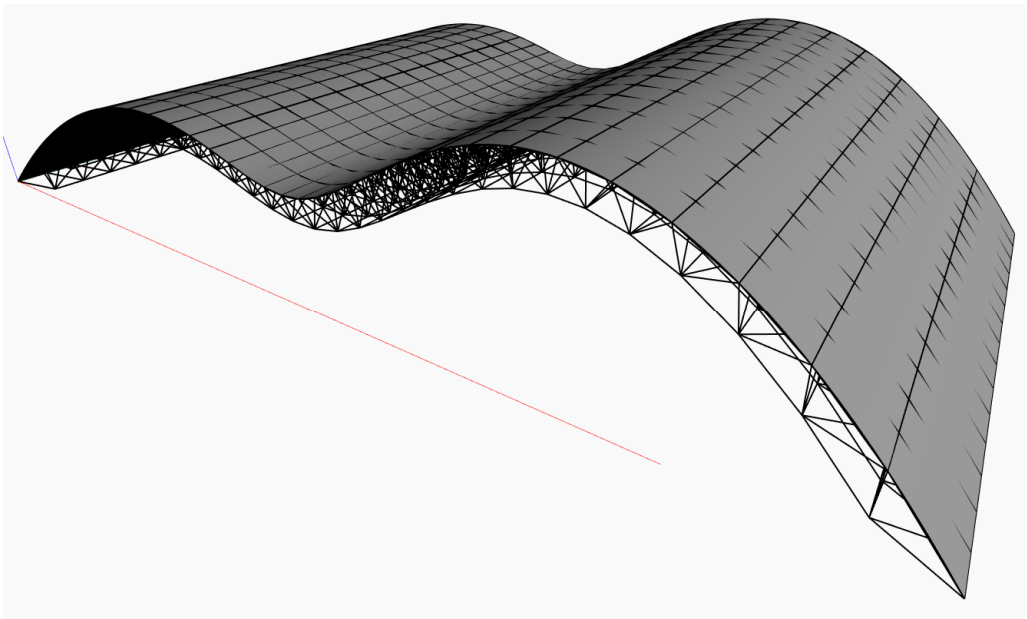


Figure 14. Geometry-free space truss structural roof.

The greatest efficiency here is the automatic generation of the spatial truss, which is the supporting structure. Defining its geometry in the traditional way seems almost impossible. Of course, a further modification would be to define the other two bounding curves in the same way. This would allow the creation of an arbitrary structure in two directions, resembling, for example, a shell floating in the wind.

4.3. Free-Geometry Structures—Described by Trigonometric Functions

The parametric description of geometry that has been used in the modeling of structural systems is based on the description of component elements using basic geometric shapes such as lines, arcs, circles, ellipses, etc. Their shape and placement are controlled by changing the parameters describing their function, and although this had the characteristics of mathematical parameterization, in practice, it was implemented implicitly. The user manually or smoothly changed the characteristic points using number slider nodes, effectively causing a change in the functions that describe the above elements.

The parametric design gives designers possibilities limited only by the mathematical apparatus. Examples of interesting objects are those where the shape or geometry of the component elements is described by more specific functions than, for example, a parabola, ellipse, or exponential function.

In the Dynamo environment, as in other text-based programming languages, the user has the ability to define his own equation describing the geometry of the structure. This example shows how this can be used in practice. Below is a script that allows the generation of an organically shaped roof, the geometry of which is described by trigonometric functions. By skillfully adjusting their course, it is possible to generate a structure reminiscent of the South Pond Pavilion in Chicago (Figure 15).

Figure 15. South Pond Pavilion in Chicago [28].

The skeleton of the structure is the outline of its section generated by an arc. It was defined on the basis of three characteristic points using the Arc.ByThreePoints node. The designed structure is stretched between two curves created by copying the first curve to a given length. The roof is closed by a SurfaceByLoft type surface defined using NURBS cross-section curves (Figure 16).

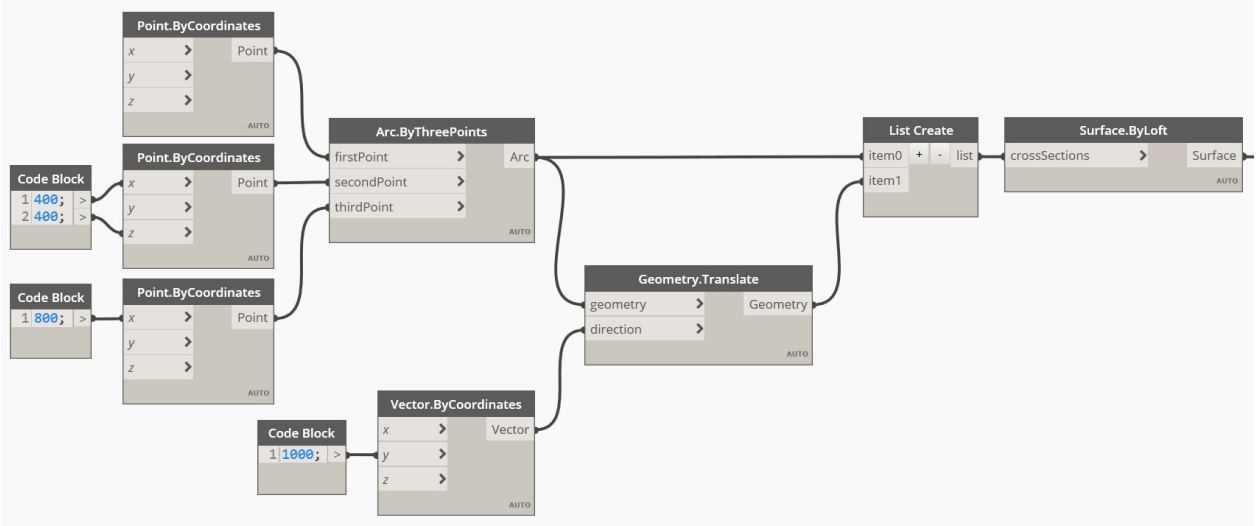


Figure 16. Definition of the roof structure.

The second part of the script is the layout of the definition of the elements that fill the walls of the roof. The following formula was used to describe their geometry:

$$u = 0..1..\#x,$$

$$i = 0..x - 1,$$

$$arc1 = (1/a) + (1/a) \times \text{Math.Cos}(i \times \pi \times 360/(x - 1)),$$

$$arc2 = (3/a) - (1/a) \times \text{Math.Cos}(i \times \pi \times 360/(x - 1)).$$

As you can see, trigonometric functions can be used to shape real shapes and profiles of structural elements. Points were extracted from these defined curves for specific parameters u and v using the Surface.PointAtParameter node (Figure 17).

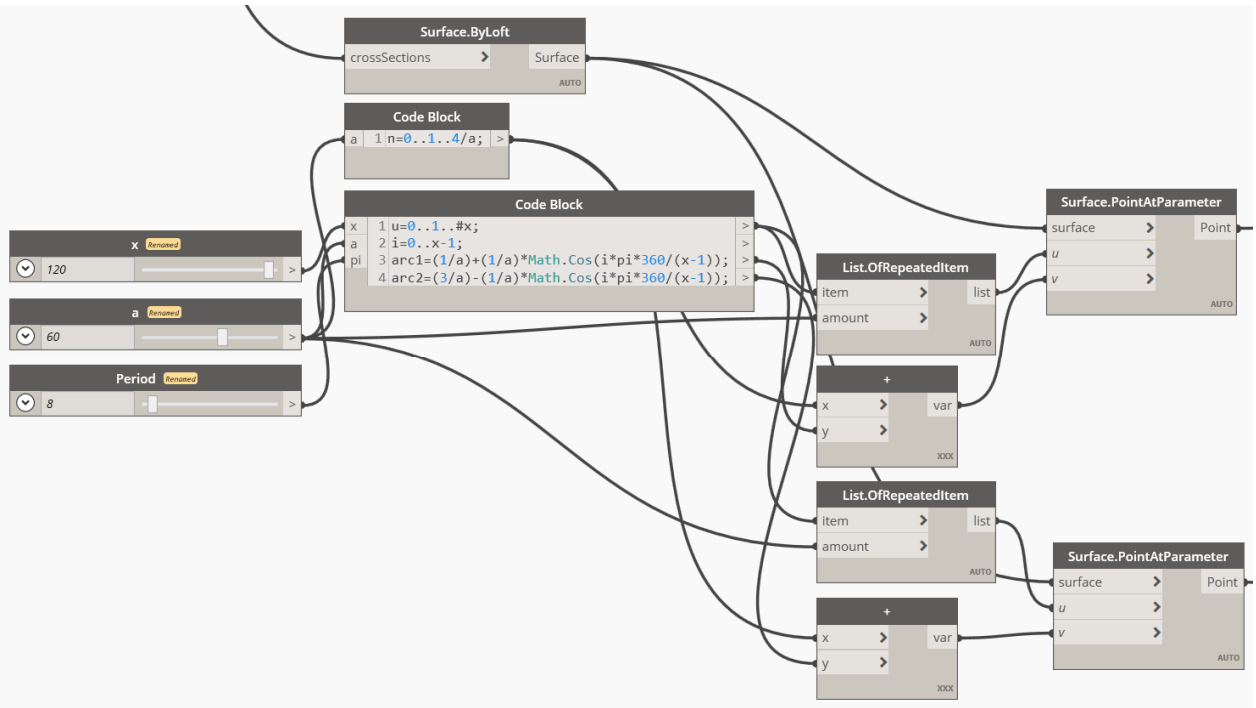


Figure 17. Definition of shapes of the arcs.

In the final stage, B-spline curves were generated between the aforementioned characteristic points using the NURBSCurve.ByControlPoints node. Their cross-sections were given using the Solid.BySweep and Rectangle.ByWidthLength functions. As a result, a roof with a shape similar to that of the South Pond Pavilion in Chicago was generated (Figure 18).

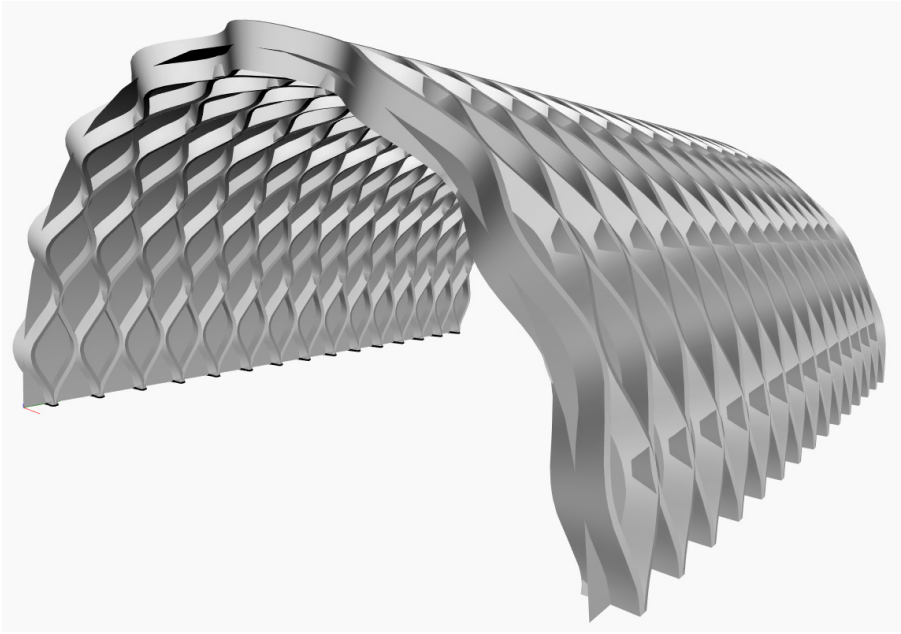


Figure 18. Free-geometry passage defined by trigonometric functions.

Of course, other mathematical functions can also be used to describe the waves of individual beams, yielding other interesting shapes.

4.4. Structural Analysis

An example of the integration of the Dynamo visual programming environment is the aforementioned inclusion of it as an add-on to one of the widely used programs for engineering calculations Autodesk Robot Structural Analysis 2022. This integration demonstrates both the search for ways to develop integrated design and the increasing awareness of designers in parametric design. Analogous to the case of the Revit environment, the user has the ability to work in parallel in Dynamo and Robot and interact between the script (model generated in Dynamo) and the numerical model in Robot.

As an example, a script will be presented, with the help of which a load-bearing arch model, its computational model (FEM), including boundary conditions (supports), cases and loads, and cross-sections, was generated, and its static analysis was run. The whole thing is coded in a vision script and generated directly in Robot after running.

The first step is to model the arc using the Arc.ByThreePoints node. Split points were generated using the Curve.PointAtPArameter option so that FEA computational elements were modeled between them (Figure 19).

Generated elements—the individual sections of the curve allow one to begin defining their numerical properties, i.e., finite element parameters. To this end, they were first given computational element attributes using the AnalyticalBar.ByCurves node (Figure 20). The material attributes and cross-sections of the elements were specified using, respectively, the options AnalyticalBar.SetMaterialByName and AnalyticalBar SetSectionByName (Figure 21).

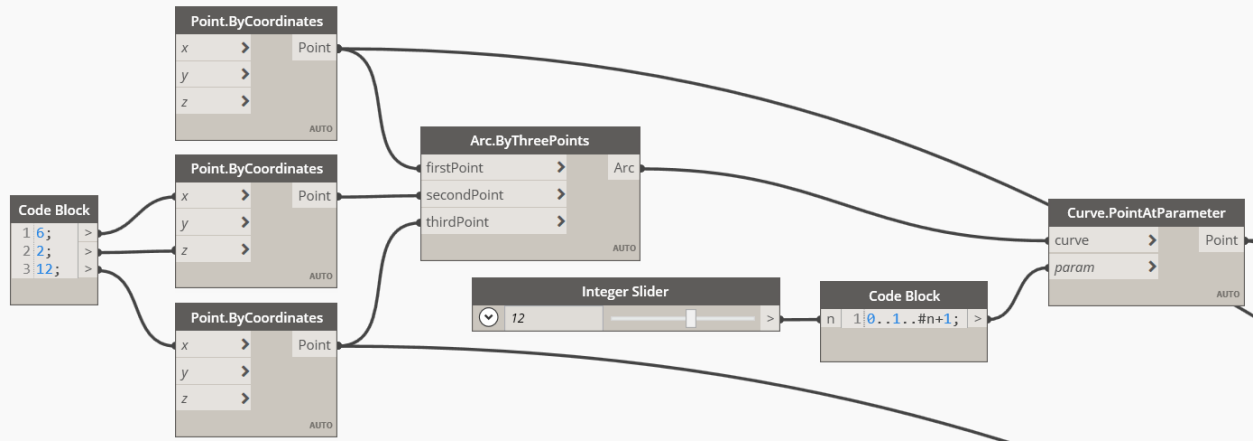


Figure 19. Definition of arch geometry.

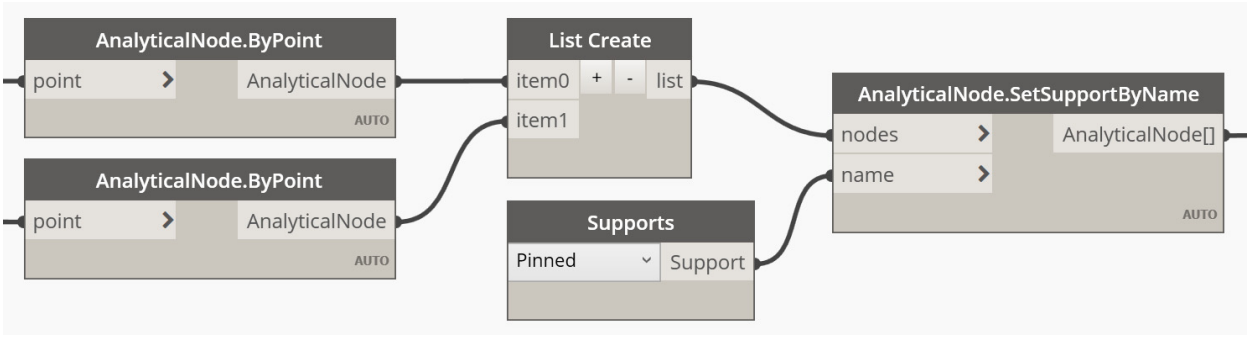


Figure 20. Definition of supports.

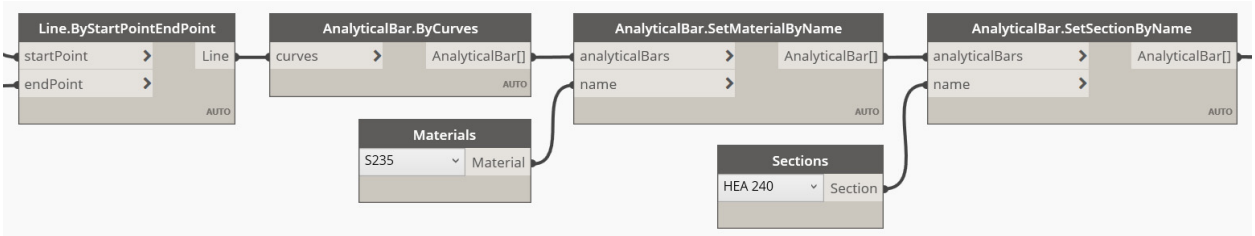


Figure 21. Definition of the FEM properties of arc elements.

Similarly, other elements of the FEM model are defined, such as supports, releases, loads (Figures 21 and 22), etc.

The last phase is to run the calculation. This is performed using the ElementsComposer node ByElements (Figure 23). This node collects all the batch information necessary to merge the modeled structure into the FEM model and run the calculation.

Ultimately, the user is provided with a model in the Robot environment, shown in Figure 24, where the static analysis of the designed structure can be performed based on the current calculations, as well as its structural verifications.

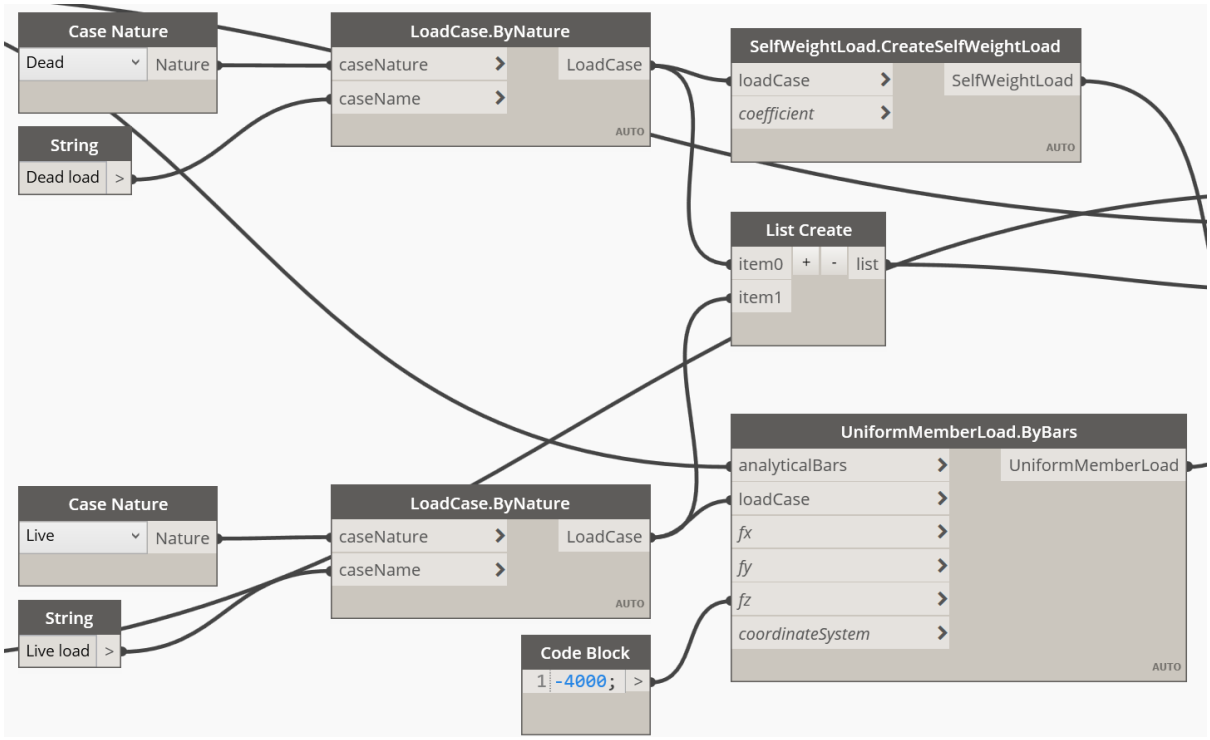


Figure 22. Definition of loads and load cases.

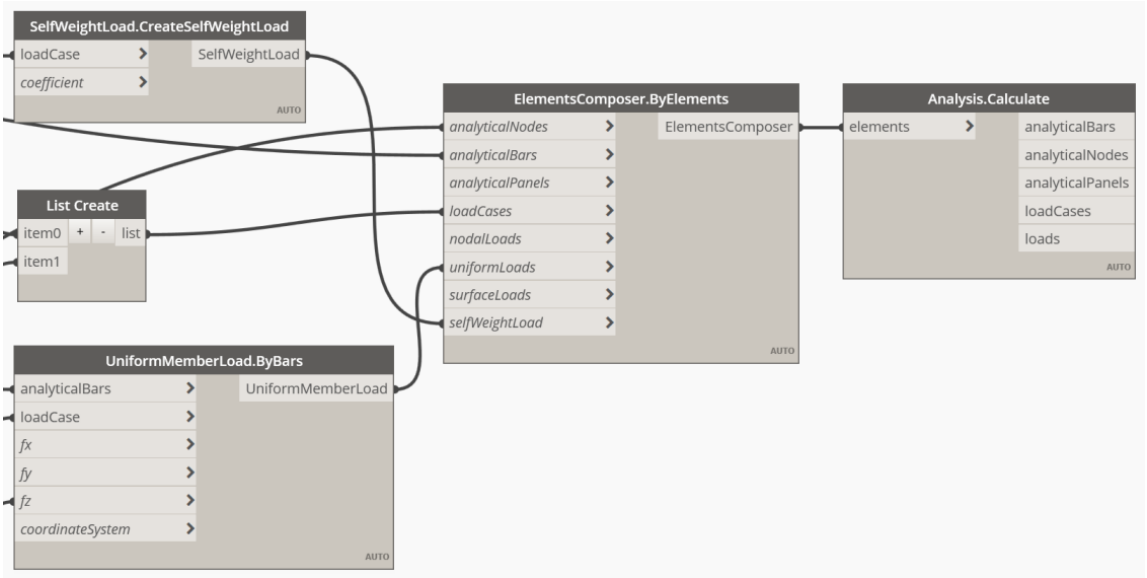


Figure 23. Run the calculations section.

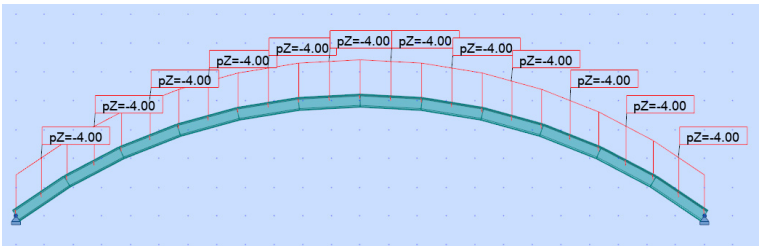


Figure 24. FEM arch model in Autodesk Robot Structural Analysis 2022.

5. Opportunities, Challenges, and Future Developments of VPL-Based Design

5.1. VPL-Based Design—Opportunities

5.1.1. Form Creation in General

At present, it seems that application in modeling the geometry of building objects is the greatest advantage of visual programming-based design environments. It is a very useful tool to design all those geometries that, on the one hand, can be easily described—broken down into simple shapes and geometric elements—but on the other hand, shapes that are so arbitrary that such a description is very complicated. As explained in the previous section and illustrated with examples, geometric elements such as splines and NURBS have very broad applications here, as they allow any editing of the final geometry using control points. The ease and dynamism of this editing allow for the creation of many variants of the designed objects.

A separate category is the design of organic forms. This applies not only to buildings, as many engineering objects such as bridges and footbridges have a bionic form, but also often refers to the plant or animal world, and even to the deepest layers of tissue structure, such as DNA. A good example is the Helix Bridge footbridge in Singapore (Figure 25), whose structure is described by two helical curves. A helix is a surface formed by the uniform rotation of a straight line about a fixed axis while moving parallel to that axis. Such a surface models well the shape of a human DNA spiral.

Figure 25. The Helix Bridge pedestrian walkway in Singapore [29].

The description of such structures can be continuous, using mathematical functions, or fluid and flexible, using splines or NURBS surfaces. Therefore, a very good solution is to use visual programming-based modeling, which offers all these possibilities. Figure 26 shows the model of the Helix Bridge walkway created in the Dynamo environment.

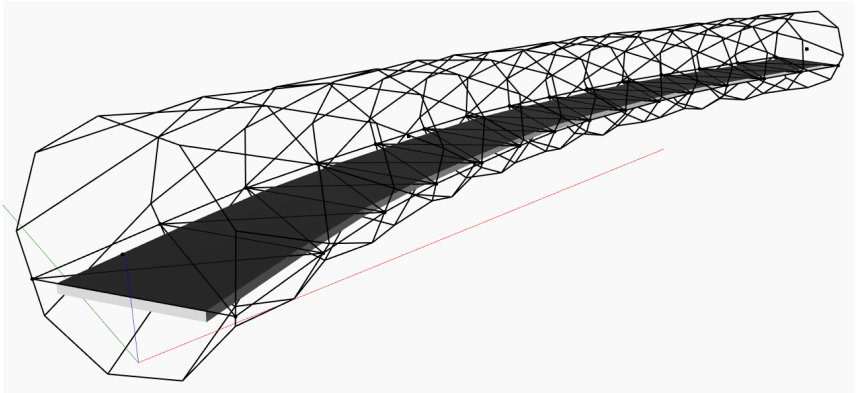


Figure 26. A model of a pedestrian walkway with a helical structure created in Dynamo.

Design of the shape of objects using visual programming is also effective in designing repeating fills that function modularly within the designed structure of the object.

5.1.2. Modelling the Structure

The use of visual programming to create an object's form is also effective in the design of repetitive infills that function modularly within the object's structure. In the case of a designed object with a stable shape, it must of course be based on a load-bearing structure that serves as its skeleton. In this case, visual programming can be used to model the structure itself. The designer has a range of options that allow for the definition of both basic elements and entire structural systems. Packages are available that automate the modeling of the most commonly used structures.

Once the structural system of the designed object has been selected, the fundamental advantage of visual programming is the ability to model and adjust the geometry of the structure almost at will. The modeling method is similar to that used in the design of the general shape of the object. However, the supporting structure of the object often uses protective elements such as walls, roofs, etc., to generate individual structural elements. In such situations, structural elements are modeled in subsequent stages rather than initially. Objects that form the "shell" of the designed object are used, allowing the definition of structural elements by operations performed on the already defined elements. This is a significant advantage as it allows the structure to be arranged and adapted to the geometry of often asymmetric and irregular surfaces.

Another possibility used here is the dynamic generation of different support systems, for example, by using the possibility of flexibly changing construction dimensions or modular subdivisions. In general, basing the geometry of the designed object on a mesh of nodes and using a dynamic approach represents a fundamental difference from the classical concept and capabilities of CAD.

5.1.3. Structural Analysis

The possibility of direct integration of the VPL design programs is a very important direction in the development of visual programming. As a result, for some time now, it has been possible to observe a closer and wider integration of parametric programming environments (Dynamo, Grasshopper) with programs or even platforms for modeling and computational analysis of construction objects, and thus their structures. Analyzing the evolution of the Dynamo environment, which has been available as a standalone program for many years, starting with its release as an Autodesk product under the name Dynamo Studio 2017, then through its introduction as a permanent part of the main Revit BIM environment and its addition to Robot 2022, one can conclude that the direction in this regard has long been set. With the continuous development and increasing dissemination of BIM technology, tools are sought that allow for more and more automation of the process of not only design but in fact handling the entire process of investment implementation. Hence, among other things, the introduction of visual programming as an add-on to the central BIM information management program Revit.

In conclusion, while there are currently opportunities for engineering computing based on the integration of VPL and computing systems, there is much to improve and develop. It seems that one direction should be the implementation of intelligent algorithms for structural optimization.

5.1.4. BIM Applications, Interoperability

As described in the Introduction, visual programming is one of the fundamental tools of BIM technology. It is gradually being introduced into successive modeling environments and related programs (detailing, calculations, etc.) and is beginning to become one of their basic add-ons and components. In this respect, VPL programs offer basic functionality that is fully compliant with the BIM standard. With the ability to create scripts independently within the available nodes and packages, VPL programs allow the creation and manage-

ment of BIM models in multiple dimensions, 4D, 5D, and beyond. Today, two main areas of VPL-based design programs in function and integration with BIM are:

- Modeling the geometry of objects.
- Information management.

These two areas are more or less integrated, depending on the specific nature of the object being designed. For example, in the case of a building, this integration may be greater because of the ability to shape the geometry and manage the key parameters of the object. In the case of an isolated structure, such as a support structure, visual programming will provide greater support for shaping the geometry.

As mentioned above, Dynamo is integrated into the Revit environment as a plug-in. It largely supports the core BIM functionality of this environment. The Dynamo node library includes the following categories: Analysis, application, elements, filters, geometry references, schedules, steel connections, transactions, and views. As you can see, Dynamo's level of integration and functionality allows you to do more than just create and manage the geometry of designed objects. For example, it is possible to create schedules, provide BIM 5D-level functionality, and detail steel connections. Importing data into Revit and then exporting to other parts of the system is also essential. These functionalities go much further [30] as energy and sensitivity analyses are performed [31]. Specialized applications of VPL in BIM are also of interest, such as bird collisions on building facades [32] or the Parametric Adaptive Skin System (PASS), which consists of kinetic facade components [33].

The capabilities of the VPL tools described here are nothing more than one of the core functionalities of BIM technology, i.e., interoperability. When linked together and meeting the requirements of certain standards, these tools can operate at the highest level of BIM technology, Level 3, known as iBIM.

5.1.5. Parametric Design

Although the idea of visual programming is based on parameterization, it can be implemented to a greater or lesser extent. A given structure can be described by the coordinates of points entered manually or by a mathematical description where the coordinates of these points are generated according to a given formula. The latter approach offers a very wide range of possibilities for creating the shape of the designed object, as well as all its components, including, of course, the structure. Parameterization, as shown in the examples, allows the engineer to use a discrete description, where the structure is divided into primitive geometries, but a continuous description can also be used, using a very large database of mathematical functions. This is, in a sense, an attempt to describe the world through mathematics, where for each geometry, a function can be defined that describes it. This feature is important both from a utilitarian, purely technical point of view and from a conceptual point of view, where a given structure reflects a non-physical world that is a strictly abstract description. It is worth remembering this when applying parametric design, which is being developed by many creators, designers, and architects.

5.1.6. Generative Design

The possibilities offered by the computerization of the design process naturally open up a wider range of possibilities, allowing the preparation of many variants and the analysis of their advantages and disadvantages. Generative design is a natural evolution in this area, successfully complementing and extending the possibilities of engineering design [34]. It covers more and more areas of application, including architecture and, more importantly, structural design [35].

The optimization problem can be described by a visual script, which is then executed and used by programs designed specifically for this purpose. For example, Revit offers an option called Generative Design. This allows the user to optimize the design process in the areas described, as well as to create their own scripts and adapt them to their individual needs. There are many examples in Revit documentation of the use of this optimization tool, such as route analysis to understand the paths of travel or evacuation in a building,

the layout of a workspace in terms of the arrangement of desks, and several others. It seems that generative programming may soon become one of the basic tools of ordinary engineering practice, alongside strictly conceptual work and studies. This depends largely on the inclusion of these tools in the most popular design environments, as well as on the awareness and education levels of the engineering designers, which are discussed below.

5.1.7. Possibilities for Other Applications of VPL Systems

The areas of visual programming support for design described in this section are currently the most developed. Of course, they are not the only ones. Visual programming environments are used wherever you can observe and capture the dependencies between the constituent elements of the object being studied. In fact, visual programming can be used wherever a given problem can be described in a more or less parameterized way. It can be said that the whole world can be described mathematically, and therefore the whole world can be parameterized. In reality, it is only up to the engineer, designer, or programmer to decide where and how to use tools such as Grasshopper, Dynamo, or similar. Just as music can be created based on the Fibonacci sequence, any physical object, process, or phenomenon can be encoded. This can be seen by looking at case studies in visual programming, not necessarily from the architecture or construction industry.

5.2. VPL Design—Challenges

Although VPL environments have existed in design for many years, several aspects remain challenging. Currently, the most relevant of these are:

- Education at the elementary and university level.
- Integration of programming and design environments.
- Few concise and comprehensive publications.

5.2.1. Education at the Elementary and University Level

Having observed the development of computerized and digitized design technology for many decades, it is impossible to begin a discussion on this topic without mentioning education in this area. Although it may not seem to be closely related to the topic at hand, it is the awareness and level of education of engineers from various industries that determine the prevalence of design support systems such as CAD, which has become an unwritten standard, as well as currently developed and gradually implemented environments and systems such as BIM technology. The development and popularization of modern design support technologies in engineering practice depend on the level of education in their implementation.

With regard to visual programming, it should be noted that it is not a common and easily implemented subject in some national educational systems adopted by the faculties that train civil engineers. The main problem is the fact that it does not work in all educational pathways, but only in selected ones, usually related to BIM technology or related fields. Another problem is a kind of substantive or perhaps more mental barrier related to the approach of civil engineering students to programming as such. Although learning one of the basic programming languages in the first semester is common practice, a civil engineering student who is also a programmer is a rarity. At this point, another phenomenon and a kind of contradiction should be noted, because programming is taught at the primary school level, in high schools as part of the computer science subject, and students are introduced to the C++ programming language at an early age. However, it seems that the teaching of programming should be implemented during the studies in construction to a greater extent. In conclusion, the problems discussed above are the basic barriers that effectively limit the future possibilities of using visual programming in engineering practice.

These obstacles can certainly be overcome. What is necessary for the development of visual programming is its widespread adoption in the environment of future and practicing engineers. This process may be slow, but it is happening, as at least in some universities

the subject of visual programming application in design and practical exercises to learn this technology are being taught.

Another way of learning is through training. Recently, there has been an increase in activity in this area. First of all, the range of training courses covering many areas of what is conventionally called computer-aided design is gradually expanding. At the same time, there is an increasing number of companies or institutions offering training that includes the topic of visual programming, mainly in the Dynamo environment with integration into the Revit environment. Online forums and video sites are also important sources of manuals and instructional videos. These are increasingly being incorporated into training. Many examples, such as the construction of trusses or complex bridges (e.g., the Helix Bridge) and the corresponding dynamo scripts, are available online free of charge and can be adapted for personal use. All this applies both to the idea of visual programming orientated towards the modeling of building objects and to the "programming" of subbranches, e.g., infrastructure. With regard to the development of this field, the issue of continuous training, not only for designers but also for engineers already working in the profession, should be highlighted in a positive way. Such a path of competence development is in line with the trend of the Western model, where an engineer (and not only) acquires his or her competences through self-improvement in the form of training courses, postgraduate studies, complementary studies, or even studies in other fields, or even pursuing a doctorate. This is another area and direction in which the development and popularization of visual programming as such is heading. As mentioned earlier, the basis of any development is, among other things, the awareness and dissemination of modern and innovative technological tools, in this case, the efficient support and automation of the process of design, implementation, investment service, and object management throughout its life cycle. As already mentioned, parametric programming, which will also be discussed in the following section in the context of the development direction, offers completely new, often abstract possibilities that go far beyond engineering design. Therefore, the training process should not be forgotten as an important aspect.

5.2.2. Integration of Programming and Design Environments

The natural consequence of developing programming environments, including enriching them with new features, is the ability to use them directly in other programs. This is a very important direction for the development of visual programming. As a result, for some time now, we have seen a closer and wider integration of parametric programming environments (Dynamo, Grasshopper) with programs or even platforms for modeling and computational analysis of building objects and their structures. If we analyze the evolution of the Dynamo environment, which has been available for many years as a standalone program, starting with its release as an Autodesk product called Dynamo Studio 2017, then its permanent integration into the main BIM Revit environment and its inclusion in Robot Structural Analysis 2022, we can conclude that the direction in this field has long been established. As BIM technology continues to develop and become more widespread, tools that will allow greater automation not only of the design process but also of the entire investment implementation process are being explored. This has led, among other things, to the introduction of visual programming as an add-on to the core BIM information management program Revit.

The implementation of visual programming in other environments facilitates the design process on several levels, not just object modeling. An engineer no longer needs to export the results of his work via intermediate means, such as universal formats, because the output of the script is automatically present in the given model. Of course, the process of creating a script is the same as creating a standalone application and requires the same programming skills as the engineer. However, simply incorporating a visual programming language into a design environment makes the work easier and faster, and also encourages people who have no previous experience to try creating visual scripts.

Another area of discussion is the integration of programming environments with design environments. At this point, it seems that the modeling of geometry and its management are the most effective aspects. The latter is fundamental considering the concept of BIM technology, where the management of information about the model is crucial. The operations that can be performed using visual scripts offer great possibilities in terms of editing, developing, or organizing the BIM model. As a result, the process of automating BIM modeling is accelerated and simplified. The designer can perform operations on data contained in the model, as well as on external data. The latter option is widely used in the implementation of engineering objects that can be modeled and located based on the real coordinates of characteristic points surveyed in nature. Another category of such operations is working on a cloud of points obtained by 3D scanning, which is becoming the basic method for even building inventory. The database operations that can be performed based on visual scripts are much broader than the possibilities of the modeling environment alone.

As far as integration of visual programming with other environments is concerned, packages containing programming options prepared with a specific scope, as described above, are also an important element. This is an important direction and element in the development of programming environments, without which it is difficult to imagine the simplification of the designer's work today.

5.2.3. Publications

The main deficiency, however, is the fact that there are few coherent scientific or scientific-didactic publications in the form of academic textbooks that would comprehensively present the idea of visual programming itself, its practical applications in structural design, and also present at least a few elementary examples of scripts on the basis of which a lay engineer could start working in a given environment. These are still rare. This fact, among others, became the main impetus for writing a certain book [36], which the author hopes will be developed and expanded in the future. In the specialist literature, there are compilations such as [37,38], but their form and scope are not adequate from the point of view presented above. However, the manual [39] is one of the most interesting in this area. Regarding the use of visual programming in architecture, as well as the idea of parametric and algorithmic architectural design, very good publications, namely [12] and [16], should be mentioned. As already mentioned, there is a need for publications related to the design and modeling of the structures of constructions based on visual programming. Therefore, this is one of the directions for future work.

5.3. VPL Design—Future Developments

5.3.1. Development of Software Environments

The development of visual programming tools, which can be used for more than just civil engineering, is one of the factors that contributes to their widespread use. The environments and interfaces required for programming are constantly being developed and updated. Suffice it to mention, for example, the number of development versions of the Dynamo SandBox program, which is updated monthly at the level of several stable versions and even dozens of development versions [40].

This development occurs in several ways. New procedures (nodes) are developed and updated to extend functionality in many different areas. This applies both to the modification or creation of new options and the creation of nodes. The latter area mainly concerns the preparation of packages containing nodes intended for narrow, specialized use in particular industries or even their parts. An example of this is the Structural Analysis for Dynamo package, which makes it possible to prepare a script in a standalone installation of Dynamo SandBox that allows a parameterized computational model of a structure to be run, launched, and the results obtained in the Robot Structural Analysis Professional program. Regarding this package specifically, there is a need for its development due to the very fact of integrating these two environments, as well as functional deficiencies in

the package itself. This is a natural direction for the development of visual programming. There are many other purpose-built packages that are continually updated (often with several updates per year), and new packages are being designed. In this respect, it seems that this will be the main direction of development for visual programming environments in the next few years, extending functionality and programming-specific problems.

Another direction of development for programming environments is their integration and implementation into other design environments or platforms. This process is gradual, and in recent years, we have seen the introduction and coupling of programming software with BIM or FEM environments. As mentioned above, the Dynamo system has become a component of environments such as Robot Structural Analysis and Revit and has also been incorporated into generative design tools.

One of the effects of the above development work is the simplification of the programming process itself through the creation and improvement of programming environments. Users are provided with increasingly powerful tools that can significantly shorten and simplify the design process.

Another extremely important element in relation to the popularization of the technology of visual programming for engineering applications is access to documentation. This includes issues related to program support, such as installation, operation, and extension of functionality (e.g., installation of add-ons), as well as knowledge of visual programming itself. It should be noted that the documentation available in these areas is quite extensive, and a person who has the ability to independently search for information, apply it, and verify it in practice should be able to handle the correct operation of environments and prepare basic scripts. Basic knowledge is usually provided free of charge by software vendors, e.g., [41–43]. Since the environments discussed here have primarily utilitarian applications, sample files—scripts, commonly referred to as samples—are an essential part of the documentation. The database of available examples is constantly being expanded by scripts created by users, either privately or by commercial companies. In the latter case, there is, unfortunately, a cost involved, as they are sometimes made available for a fee. In addition to the free knowledgebase, instructional videos are prepared and shared on the most popular video services available on the Internet. At the same time, there is a market for training courses offered by commercial companies, which sometimes organize free, usually short training sessions, such as webinars.

5.3.2. Design Automation

A natural benefit of using any software in engineering design is the degree of automation it brings to the process. The same is true for visual programming, which, if used properly, can increase efficiency in this area to an extent that cannot be matched even by CAD or even BIM systems. The natural direction of development for this environment is therefore the development of the parameterization of the entire object design process and even the implementation of the entire investment. Section 4 presents three examples of the use of scripts to model parametric structures, from the simplest to the most complex, showing the evolution of this process. Visual programming seems to offer the greatest potential for application in this area. Creating complex, asymmetric, and nonmodular geometries is one of the biggest problems designers face. Global architectural trends have been moving toward the design of organic, bionic forms for some time now, as evidenced by the structures of various built objects such as buildings or bridges designed by architects such as Santiago Calatrava [44] or Zaha Hadid [45]. It seems that the intensification of activities in the field of parameterization of an increasing number of already typical structures should take place precisely in this area. Therefore, the use of parameterization in the design process itself is one of the basic directions in which the visual programming trend should develop more intensively. Although this direction has already been established quite some time ago in the field of programs and now applications not necessarily intended for computers, it seems that it should be significantly developed and orientated in the field of visual programming.

5.3.3. Application of Artificial Intelligence in Engineering Design

In recent years, certain software producers and, more generally, companies that develop IT technologies and products, have stepped up their efforts to develop and disseminate the technology commonly known as artificial intelligence (AI). In fact, AI is already being used in everyday life, often without people realizing it. Unlocking a smartphone with a fingerprint or using facial or iris scanning as a unique way to secure devices seems imperfect because an AI-based application that can generate a so-called universal fingerprint was developed fairly quickly. In the next few years, AI will become increasingly involved in our lives, trying to relieve us of many activities that require more or less intellectual effort.

So why not use AI in design? Work in this area has been going on for some time, and one of the results is the development of generative design tools for engineers. In this case, AI does not take the work out of our hands but allows us to perform work that is impossible to do manually. Just as it is impossible to compare the computing power of a well-equipped computer with the capabilities of cloud computing, it makes no sense to compare computing power in terms of generating variants of situations that can be obtained using generative design. Why not go further and try to develop intelligent programming tools that would optimize the design process to some extent? Tools that have been used for some time, such as genetic algorithms used to optimize various structural designs [46,47], and others [48], could be applied here. By applying the appropriate criteria, the engineer would obtain a product that would be tailored to his needs. This may seem like too big a leap in terms of the development of programming tools, which are currently not very advanced. In reality, it is simply a matter of developing software for the existing environment. Obviously, this requires much work, but given the opportunities it offers, it should happen sooner or later.

An example of an AI application in the areas discussed above is the Python code generated by ChatGPT to design a roof truss, as shown in Figure 27. Due to current limitations, this code is intended to be executed within the Dynamo environment.

python Copy code

```
import clr
clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

# Create a new Revit project
doc = DocumentManager.Instance.CurrentDBDocument

# Define the roof truss parameters
span = 20  # Span of the roof truss
pitch = 6  # Pitch of the roof truss
depth = 2  # Depth of the roof truss

# Create the roof truss geometry
point1 = XYZ(0, 0, 0)
point2 = XYZ(span, 0, 0)
point3 = XYZ(span / 2, pitch / 12, 0)

line1 = Line.CreateBound(point1, point2)
line2 = Line.CreateBound(point2, point3)
line3 = Line.CreateBound(point3, point1)

# Create the roof truss as a structural framing element
truss = StructuralFramingUtils.CreateFraming(doc, line1, StructuralType.Truss)
truss.Symbol.Activate()
```

Figure 27. Python code generated by AI to design a roof truss.

It creates a roof truss as a structural framing element in Revit, based on the specified span, pitch, and depth parameters. It sets various properties of the truss, such as depth, pitch, span, family, and type, and assigns it to a specific level. Finally, the code outputs the created truss element. This code assumes to have the necessary Revit and Dynamo packages installed and referenced in the Dynamo environment.

The possibilities offered by software manufacturers for some time now give rise to the hope that, by improving the tools offered and expanding their areas of application, it will be possible to use this technique in everyday applications, such as optimizing the weight of a designed structure.

6. Results

The main findings of this study can be summarized as follows:

Current applicability: The study positively evaluates the current applicability of VPL technology in structural design. It is found to be highly effective in creating complex geometries for building structures, particularly those with asymmetrical shapes that are challenging to design using traditional methods. This is highly effective when multiple design options are at the conceptual stage.

Optimization: By using an algorithmic-parametric approach to design, and in particular generative design, there are great opportunities for structural optimization. The ongoing development of optimization methods, such as the use of genetic algorithms, in VPL-based design is noteworthy.

VPL-BIM environments and integration: The development of VPL environments dedicated to structural design, along with their integration with design systems such as Building Information Modeling (BIM), is seen as a positive advancement. This integration improves the overall design process. A noticeable development is the use of VPL in BIM, 4D, 5D, and higher. This allows for interoperability in iBIM.

Limitations in structural computing: While VPL technology is beneficial for form creation, it has some limitations in structural computing. Existing systems mainly allow for simple static analyses, and more advanced capabilities, such as automatic structural optimization, are lacking. Current optimization tasks often require the use of individual scripts and sophisticated methods.

Designer competencies: The awareness and experience of a wide range of designers of the use of VPL in structural design need to be improved. The theoretical foundation for VPL-based design requires increased educational efforts during academic studies and professional work. This also applies to the need for preparing and updating teaching literature.

Continuous development: VPL environments are continuously evolving, and their theoretical operating algorithms are being improved over time. This highlights the ongoing progress in VPL technology.

7. Conclusions

Summarizing the results of this study, they can be divided into two scopes, strictly practical and theoretical. In the first case, the current applicability of VPL technology in structural design should be evaluated positively. It is most effective for the form creation of building structures, which nowadays often adopt asymmetrical geometries that are practically impossible to design in the traditional way. Also positive is the development of VPL environments dedicated to structural design and their integration with design systems, including BIM. In the field of structural computing, it seems that the dynamic development of capabilities based on visual programming is necessary. Currently, existing systems mostly allow one to perform simple static analyses. What is lacking are at least automatic options for structural optimization. This requires the preparation of individual scripts and the use of sophisticated methods, such as genetic algorithms. When discussing the theoretical aspects of the application of VPL in structural design, the first thing to note is the rather low awareness and experience of designers in this area. It is necessary to establish a theoretical basis for design based on VPL. This requires the need for more educational

activities during studies and at work. Another element is the observed development of VPL environments, whose theoretical operating algorithms are being developed all the time.

The future will show which of the directions described in this study for the development of visual programming in engineering design will develop most rapidly and intensively. This development will, of course, vary according to the natural needs of the design industry. Certain paths of development will certainly lead to the abandonment of activities and a change in the concepts used so far. However, it is to be hoped that visual design will be on an upward trend because the fundamental and natural direction of development is the automation and optimization of the design process. An important element is also the issue of technical documentation media, which will sooner or later be created in digital form, which will become the basic form. The digital model of a building, which will become the only complete source of information even in BIM technology, will therefore be the only complete image and carrier of technical documentation and more. In this respect, visual programming offers and will continue to offer more and more possibilities, not only allowing the generation of increasingly complex geometries but also offering incredible possibilities for managing the information encoded in the model.

In general, the study indicates that VPL technology shows promise in structural design, especially for complex geometries. However, there are challenges to address, such as enhancing structural computing capabilities and increasing designers' awareness and expertise in using VPL effectively. The ongoing development of VPL environments and algorithms will play a significant role in modeling its future applications in engineering design.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Weisberg, D.E. The Engineering Design Revolution: The People, Companies and Computer Systems That Changed Forever the Practice of Engineering. 2008. Available online: https://images.designworldonline.com.s3.amazonaws.com/CADhistory/85739614-The-Engineering-Design-Revolution-CAD-History.pdf (accessed on 30 May 2023).
2. Bangwal, D.; Tiwari, P.; Chamola, P. Workplace design features, job satisfaction, and organization commitment. *SAGE Open* **2017**, *7*, 21582440177. [CrossRef]
3. Bangwal, D.; Tiwari, P. Environmental design and awareness impact on organization image. *Eng. Constr. Archit. Manag.* **2019**, *26*, 29–45. [CrossRef]
4. Bangwal, D.; Tiwari, P. Workplace environment, employee satisfaction and intent to stay. *Int. J. Contemp. Hosp. Manag.* **2019**, *31*, 268–284. [CrossRef]
5. Bangwal, D.; Suyal, J.; Kumar, R. Hotel building design, occupants' health and performance in response to COVID-19. *Int. J. Hosp. Manag.* **2022**, *103*, 103212. [CrossRef] [PubMed]
6. Yares, E. 50 Years of CAD. *Des. World* **2013**, 66–71.
7. Kossakowski, P. Zastosowanie systemu obiektowej informacji o konstrukcji w projektowaniu CAD. *Syst. J. Transdiscipl. Syst. Sci.* **2012**, *16*, 279–288.
8. Eastman, C. The Use of Computers Instead of Drawings in Building Design. *AIA J.* **1975**, *63*, 46–50.
9. Sacks, R.; Eastman, C.; Lee, G.; Teicholz, P. *BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers*, 3rd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2018.
10. Tomana, A. *BIM—Innowacyjna Technologia w Budownictwie. Podstawy, Standardy, Narzędzia*; PWB Media: Warsaw, Poland, 2016.
11. Kasznia, D.; Magiera, J.; Wierzowiecki, P. *BIM w Praktyce: Standardy, Wdrożenia, Case Study*; Wydawnictwo Naukowe PWN: Warsaw, Poland, 2017.
12. Bonenberg, W.; Giedrowicz, M.; Radziszewski, K. *Współczesne Projektowanie Parametryczne w Architekturze*; Wydawnictwo Politechniki Poznańskiej: Poznań, Poland, 2019.
13. Wassim, J. *Parametric Design for Architecture*, 1st ed.; Laurence King Publishing: London, UK, 2013.
14. Woodbury, R. *Elements of Parametric Design*; Routledge: London, UK, 2010.
15. Tomoko, S.; Ferré, A. *From Control to Design: Parametric/Algorithmic Architecture*, 1st ed.; Actar-D: Barcelona, Spain, 2008.

16. Helenowska-Peschke, M. *Parametryczno-Algorytmiczne Projektowanie Architektury*; Wydawnictwo Politechniki Gdańskiej: Gdańsk, Poland, 2014.
17. Burry, J.; Burry, M. *The New Mathematics of Architecture*; Thames and Hudson: London, UK, 2010.
18. Burry, M. *Scripting Cultures: Architectural Design and Programming*; Wiley: Chichester, UK, 2011.
19. Caneparo, L. *Digital Fabrication in Architecture, Engineering and Construction*, 1st ed.; Springer: Dordrecht, The Netherlands, 2014.
20. Adu, M.K.; Abe, O.E. Improving Structural Designs with Computer Programming in Building Construction. *IOSR J. Comput. Eng.* **2014**, *16*, 10–16.
21. Collao, J.; Lozano-Galant, F.; Lozano-Galant, J.A.; Turmo, J. BIM Visual Programming Tools Applications in Infrastructure Projects: A State-of-the-Art Review. *Appl. Sci.* **2021**, *11*, 8343. [CrossRef]
22. Funari, M.; Spadea, S.; Ciantia, M.; Lonetti, P. Visual programming for the structural assessment of historic masonry structures. In Proceedings of the 8th Euro-American Congress Construction Pathology, Rehabilitation Technology and Heritage Management—REHABEND 2020, Granada, Spain, 24–27 March 2020; pp. 1–8.
23. Korus, K.; Salamak, M.; Jasiński, M. Optimization of geometric parameters of arch bridges using visual programming FEM components and genetic algorithm. *Eng. Struct.* **2021**, *241*, 112465. [CrossRef]
24. Salah, M.; Elbeltagi, E.; Elsheikh, A. Dynamo Visual Programming-Based Generative Design Optimization Model for Construction Site Layout Planning. *Mansoura Eng. J.* **2021**, *46*, 31–40.
25. Visual Programming Language from FOLDOC. Available online: https://foldoc.org/visual+programming+language (accessed on 30 May 2023).
26. Available online: https://levstal.com/steel-structures/steel-trusses-and-frames/ (accessed on 30 May 2023).
27. Available online: https://www.setareh.arch.vt.edu/safas/007_fdmtl_21_spatial_structure.html (accessed on 30 May 2023).
28. Available online: https://th.bing.com/th/id/R.fd9338be83c668381c7d061f0b4c01ef?rik=s9P2%2fCpDJZj0Pg&pid=ImgRaw&r=0 (accessed on 30 May 2023).
29. Available online: https://th.bing.com/th/id/R.93b349fbf8b03060ba5de24a5e790143?rik=IW6t%2bEmT6c%2bpYg&pid=ImgRaw&r=0 (accessed on 30 May 2023).
30. Shi, A.; Shirowzhan, S.; Sepasgozar, S.M.E.; Kaboli, A. 5D BIM Applications in Quantity Surveying: Dynamo and 3D Printing Technologies. In *Smart Cities and Construction Technologies*; Shirowzhan, S., Zhang, K., Eds.; IntechOpen: London, UK, 2020. Available online: https://www.intechopen.com/chapters/71143 (accessed on 30 May 2023).
31. Shahsavari, F.; Koosha, R.; Yan, W. *Uncertainty and Sensitivity Analysis Using Building Information Modeling*; Schnabel, M.A., Fukuda, T., Haeusler, M.H., Eds.; The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA): Hong Kong, China, 2019; Volume 1, pp. 615–624.
32. Kensek, K.; Ding, Y.; Longcore, T. Green building and biodiversity: Facilitating bird friendly design with building information models. *J. Green Build.* **2016**, *11*, 116–130. [CrossRef]
33. Shen, Y.T.; Lu, P.W. Development of Kinetic Facade Units with BIM-Based Active Control System for the Adaptive Building Energy Performance Service. In Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA, Melbourne, Australia, 30 March–2 April 2016.
34. Krish, S. A practical generative design method. *Comput.-Aided Des.* **2011**, *43*, 88–100. [CrossRef]
35. Dapogny, C.; Faure, A.; Michailidis, G.; Allaire, G.; Couvelas, A.; Estevez, R. Geometric constraints for shape and topology optimization in architectural design. *Comput. Mech.* **2017**, *59*, 933–965. [CrossRef]
36. Kossakowski, P. *Application of Visual Programming in the Design of Building Structures [Zastosowanie Programowania Wizualnego w Projektowaniu Konstrukcji Budowlanych]*, 1st ed.; Wydawnictwo Politechniki Świętokrzyskiej: Kielce, Poland, 2022. (In Polish)
37. Dynamo Language Manual. Available online: https://dynamobim.org/wp-content/uploads/forum-assets/colin-mccroneautodesk-com/07/10/Dynamo_language_guide_version_1.pdf (accessed on 30 May 2023).
38. Dynamo: Visual Programming for Design. Available online: https://help.autodesk.com/sfdcarticles/attachments/Dynamo_Visual_Programming_for_Design.pdf (accessed on 30 May 2023).
39. Sgambelluri, M. *Dynamo and Grasshopper for Revit Cheat Sheet Reference Manual*; Marcello Sgambelluri: Los Angeles, CA, USA, 2020.
40. Dynamo Builds. Available online: https://dynamobuilds.com (accessed on 30 May 2023).
41. Learn—Dynamo BIM. Available online: https://dynamobim.org (accessed on 30 May 2023).
42. Rhino—Learn to Use Rhino. Getting Started. Available online: https://www.rhino3d.com/learn/?query=kind:%20grasshopper&modal=null (accessed on 30 May 2023).
43. Rhino—Learn to Use Rhino. Grasshoper. Available online: https://www.rhino3d.com/learn/?keyword=kind:%20rhino_win (accessed on 30 May 2023).
44. Carrillo de Albornoz Fisac, C. *Santiago Calatrava (Ultimate Collection)*; Assouline: New York, NY, USA, 2013.
45. Jodido, P. *Zaha Hadid. Complete Works 1979—Today*; Taschen: Cologne, Germany, 2020.
46. Toropov, V.; Mahfouz, S. Design optimization of structural steelwork using a genetic algorithm, FEM and a system of design rules. *Eng. Comput.* **2001**, *18*, 437–460. [CrossRef]

47. Łodygowski, T.; Szajek, K.; Wierszycki, M. Optymalizacja konstrukcji półskorupowej z użyciem algorytmu genetycznego. *Górnictwo Odkryw.* **2010**, *51*, 88–92.
48. Łodygowski, T.; Szajek, K.; Wierszycki, M. Optimization of dental implant using genetic algorithm. *J. Theor. Appl. Mech.* **2009**, *47*, 573–598.