Ybl Journal of Built Environment, Volume 10. Issue 1. (2025)
10.12700/jbe-2025-0011

SOME EXPERIENCES WITH AI-SUPPORTED ALGORITHMIC PARAMETRIC 3D MODELLING

Eszter Katona,[1] János Katona [2]

[1]Faculty of Architecture and Planning, Institute of Architecture and Design,
Technische Universität Wien, Austria
eszter.katona@tuwien.ac.at, katonaeszterkrisztina@gmail.com
[2]Department of Building Informatics, Geodesy and Mathematics, Institute of Civil Engineering, Óbuda University, Budapest, Hungary
katona.janos@uni-obuda.hu

Abstract: *This paper presents our experience on what degree can artificial intelligence (AI) help creatives – such as architects – to program parametric 3D models via Rhino Python. While there are several studies exploring ChatGPT's ability to write code, including Python scripts, its application in generating 3D objects within CAD programs remains unexplored. Our initial question was whether ChatGPT possesses any knowledge of RhinoScriptSyntax at all and if so, how much it can help create parametric 3D models. Therefore, we explored the use of AI in two follow up roles. It had to recognise the spatial structure of objects from photographs and then it had to assist in generating parametric algorithms to visualize these objects within an architectural modelling software. Based on our findings, we offer recommendations for the effective use of AI in similar design and modelling workflows.*

Keywords: *parametric modelling, generative design, Rhino-Python, ChatGPT.*

1. INTRODUCTION

Following a brief introduction to AI and programmable CAD software, this paper presents a detailed workflow example and the results of one of our tests, using a decorative spiral structure as the subject. We begin with a spiral code example that we wrote ourselves and compare it to what and how we were able to achieve through a text-based conversation with AI.

In the first section we write about generative design and artificial intelligence in general. In section 2. we write a code to illustrate the operation of Rhino-Python environment with a concrete example. In the section 3 we specify our test environments, and we investigate ability and limits of AI by an interesting and example. Finally, in the fourth section we summarise our experiences and we offer some recommendations.

1.1. *Parametric 3D Models Constructed with Code*

Many 3D modelling applications (CAD software) are programmable, meaning they can execute scripts or code that define how a model is built. By adjusting the input parameters of the program, a wide range of design variations can be explored in a short amount of time. (For example [1]) This topic has an extensive body of literature. Although this article provides its own insights, article [2] and its 99 references, in our opinion, are a good summary of this issue.

For instance, in AutoCAD, it is possible to run scripts created in any text editor. Users can also utilize built-in environments like Visual LISP (formerly AutoLISP) or Visual Basic for Applications (VBA).

For the experiments discussed in this paper, we used Rhino 8 modelling software. We chose Rhinoceros 3D – commonly referred to as Rhino – because it supports the widely used and modern Python programming language, making it an ideal platform for algorithmic modelling. Additionally, Rhino is one of the most-commonly used software tools in freeform design, which further supported our decision.

1.2. *About artificial intelligence in general*

In 1955, John McCarthy, Marvin L. Minsky, Nathaniel Rochester and Claude E. Shannon proposed the term "artificial intelligence" to describe the process of solving a formulated problem by machine learning. We do not prescribe a set of clear instructions (algorithm) for the computer point by point, but "train" it with many samples and let it build the set of rules by itself.

So, research and applications of AI have been going on for many decades, but the public was first exposed to it on 30 November 2022. That was the date of the ChatGPT prototype, which quickly gained a lot of popularity. It is a large language model (LLM) that requires virtually no prior knowledge. You can "converse" with it, ask questions, make requests in the same way as if you were talking to a live person, an expert or your teacher, and "they" will respond in the same way.

Although the big language models have been in the public domain since 2022, there are now dozens of different public models available. The literature on LLMs is also extensive. We believe this article [3] and its 52 references provide a good summary of the advantages, disadvantages and limitations of LLMs.

2. PARAMETRIC SPIRAL EXAMPLE

To evaluate the AI-generated script and give the reader a glimpse into how RhinoScriptSyntax environment works, we began our process with a simple script of our own. We developed the following definition for a 3D spiral within the Rhino-Python environment*:*

```
import rhinoscriptsyntax as rs
import math

def draw_3d_spiral (angle = math.pi / 4, growth = 0.5, up = 1, step = 50):
    # variables:
    # angle = angle increment
    # growth = spiral growth factor
    # up = vertical increment
    # step = number of points

    x = y = z = r = theta = 0.0
    points = [ ]

    for i in range(step):
        theta += angle
        r = growth * theta
        x = r * math.cos(theta)
        y = r * math.sin(theta)
        z += up
        points.append(rs.coerce3dpoint((x, y, z)))

    crv = rs.AddCurve(points, degree=2)
    return crv
```

(This code can be downloaded from the link listed in the bibliography as reference [4]: *spiral.py)*

The procedure draws an upward-expanding spiral curve. (As we will see later in this paper, there is a specific reason why this curve was chosen as the example.) The algorithm calculates the coordinates of the control points and then uses Rhino's NURBS curve (B-spline) command to fit a curve through them.

The procedure takes four variables:

The **first** *angle* determines the density of the control points of the B-spline.
(The lower this number, the closer the control points are located.)
The **second** *growth* controls the degree of expansion,
The **third** *up* sets the vertical rise of the spiral,
The **fourth** *step* defines the total number of control points.

The program initially performs calculations in polar coordinates, which are then converted into Cartesian (rectangular) coordinates.

When executed with three different parameter sets,

```
draw_3d_spiral(math.pi / 2, 0.2, 0.5, 40)
draw_3d_spiral(math.pi / 4, 0.5, 0.2, 50)
draw_3d_spiral(math.pi / 6, 0.3, 0.3, 70)
```

the resulting curves can be seen in different view in Figure 1., 2. and 3.

Figure 1. Front view of the curves generated by the *spiral.py* code.

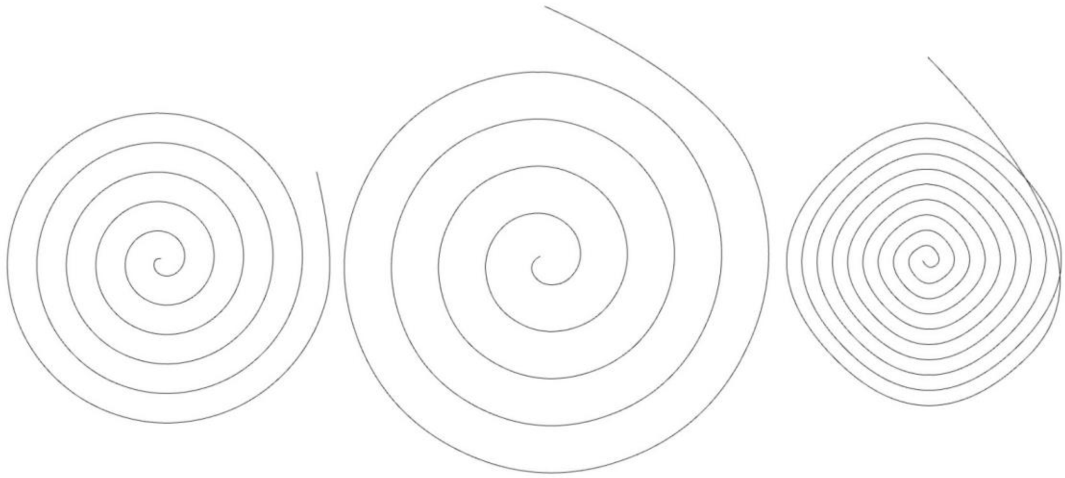

Figure 2. Top view of the curves generated by the *spiral.py* code.

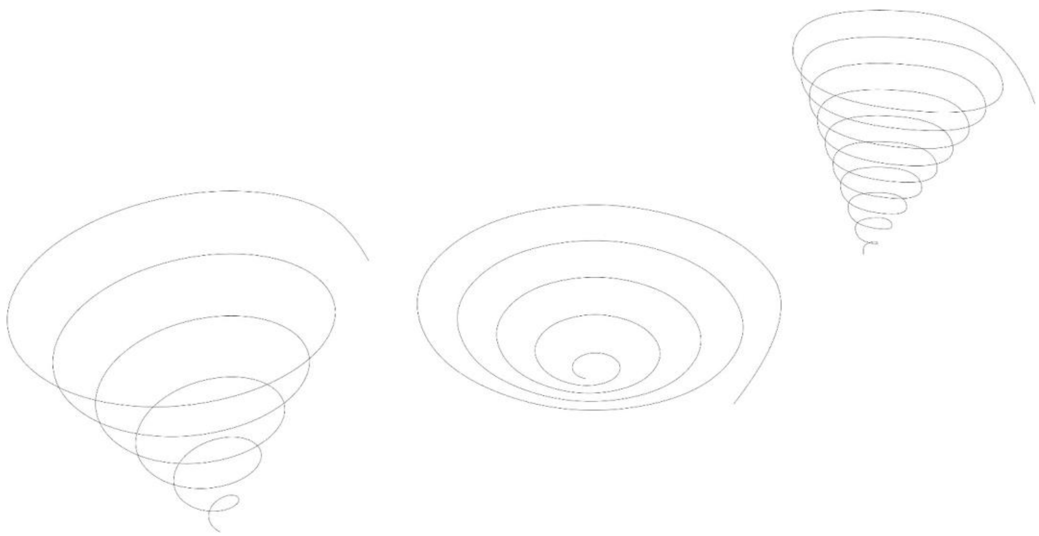

Figure 3. Perspective view of the curves generated by the spiral.py code.

In particular, the right-hand side of the top view clearly shows a flaw in the procedure: we did not specify the terminating tangent of the curve at the last control point. In the case of B-splines, the curve is fitted to the control points in such a way that the curve-fitting algorithm considers the two neighbours of the current control point. Since the initial and final control points have only one immediate neighbour, the curve will only be clear and correct if the tangent line of the spline at these points is also given.

Rather than manually developing and testing the algorithm to generate this curve, we took an exploratory approach by involving artificial intelligence. This allowed us to see how effectively AI could handle the task and what kind of solutions it might propose. AI is expected to perform well in writing program code because programming has a clearly described instruction set, syntax and semantics. (Spoiler: indeed.)

3. TESTING AI CAPABILITIES WITH ARCHITECTURAL MODELLING SOFTWARE

3.1. *The specific test environment*

ChatGPT formally supports 58 languages, although the relevance of its answers (according to ChatGPT's own claims) may vary for each language. Obviously, during its training, it encountered many more English texts than Hungarian (the authors' native language), so we used English. We worked with ChatGPT version 4, we subscribed to the "Plus" package.

Our work style was as follows: first, we uploaded one or more images of a specific object or building. This image could be a photo downloaded from the internet, but of course we also used a picture we had taken ourselves with our mobile phone. We then asked the AI to generate a Rhino-Python program code to implement a 3D model of the object. We ran the code in Rhino-Python, and if we were not satisfied with the result, we asked the AI to improve the algorithm, and so on. These iteration steps were repeated until we were satisfied with the acceptable result.

3.2. *Example of a concrete task*

As an example, we analyse the task described in detail below. We chose this particular problem because it clearly illustrates how AI can be applied, and we believe it encapsulates many of the typical strengths and limitations of current versions of artificial intelligence.

We came across an interesting spatial composition in a shopping mall and took the two photos shown in Figure 4. and 5.

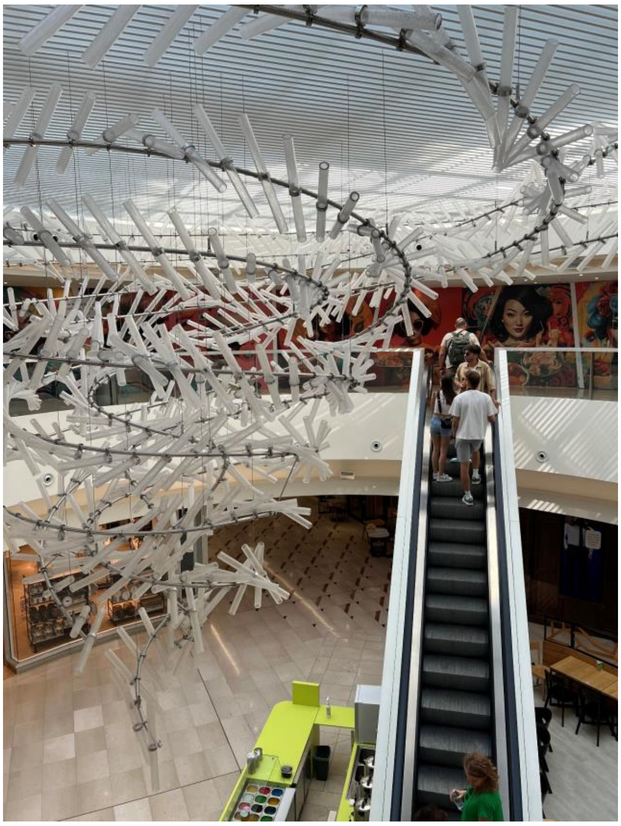

Figure 4. A photo of a spatial composition with escalator and people

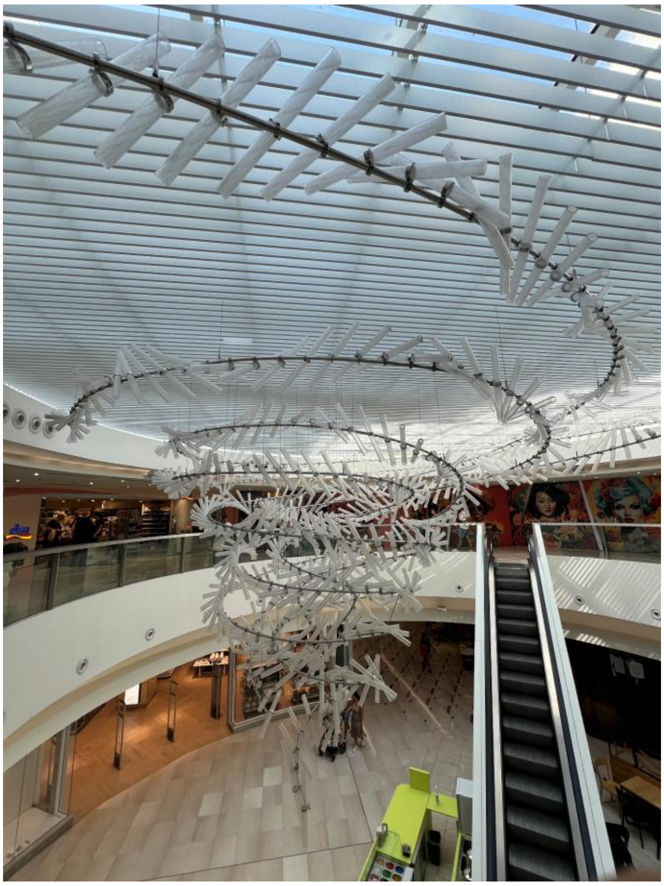

Figure 5. A second photo of the same composition

These two photos were uploaded to ChatGPT and asked to generate a Rhino-Python code using the command below:

Write me a Rhino Python program which generate this object, please. :)

This is the ChatGPT-generated answer and code we got to the command above:

This sculpture has a striking helical/spiral form, composed of repeating translucent tube-like elements arranged around a curved backbone. Based on the images, we can approximate the structure as a parametric helix with “ribs” (cross rods) distributed along it.

*Here's a **Rhino Python** script using RhinoCommon that will help generate a similar form. The design is broken into:*

- *A helix-like base curve*
- *Repeating elements (rods) oriented perpendicularly around the base curve*
- *Optional: variation and noise for a more organic result*

Python code:

```
import rhinoscriptsyntax as rs
import math
import random

def create_spiral_structure():
    # Parameters
```

```
    turns = 3                  # Number of turns in the helix
    points_per_turn = 30       # Number of ribs per turn
    radius = 10                # Radius of the helix
    height = 30                # Total height
    rod_length = 3             # Length of each rib rod
    randomness = 0.4           # Random angle variation

    # Generate helix base curve
    points = [ ]
    for i in range(turns * points_per_turn):
        t = float(i) / points_per_turn * 2 * math.pi
        z = float(i) / (turns * points_per_turn) * height
        x = radius * math.cos(t)
        y = radius * math.sin(t)
        points.append(rs.coerce3dpoint([x, y, z]))

    # Create rods (ribs) along the helix
    for i, pt in enumerate(points):
        # Get tangent by finite difference
        if i < len(points) - 1:
            next_pt = points[i+1]
            tangent = rs.VectorCreate(next_pt, pt)
            tangent = rs.VectorUnitize(tangent)
        else:
            tangent = rs.VectorCreate(pt, points[i-1])
            tangent = rs.VectorUnitize(tangent)

        # Create perpendicular vector to the tangent
        perp = rs.VectorCrossProduct(tangent, [0, 0, 1])
        if rs.VectorLength(perp) == 0:
        perp = rs.VectorCrossProduct(tangent, [1, 0, 0])
        perp = rs.VectorUnitize(perp)

        # Add a random rotation for a more organic look
        angle = random.uniform(-randomness, randomness) * 180
        perp = rs.VectorRotate(perp, angle, tangent)

        # Create rod as a line
        start_pt = rs.PointAdd(pt, rs.VectorScale(perp, rod_length / 2))
        end_pt = rs.PointAdd(pt, rs.VectorScale(perp, -rod_length / 2))
        rs.AddLine(start_pt, end_pt)

# Run the function
create_spiral_structure()
```

How to use this in Rhino:

1. *Open Rhino.*
2. *Open the Python script editor via EditPythonScript.*
3. *Paste the above code.*

4. *Run the script.*

You can tweak the turns, radius, rod_length, and randomness to better match the actual sculpture's form.

Would you like me to create a variation with clustered branching elements or bifurcations like the denser section in the lower part of the sculpture?

This code can be downloaded from the link listed as reference [5] in the bibliography: *Lamp_v1.py*
When executed in Rhino, the result shown in Figure 6 was produced.

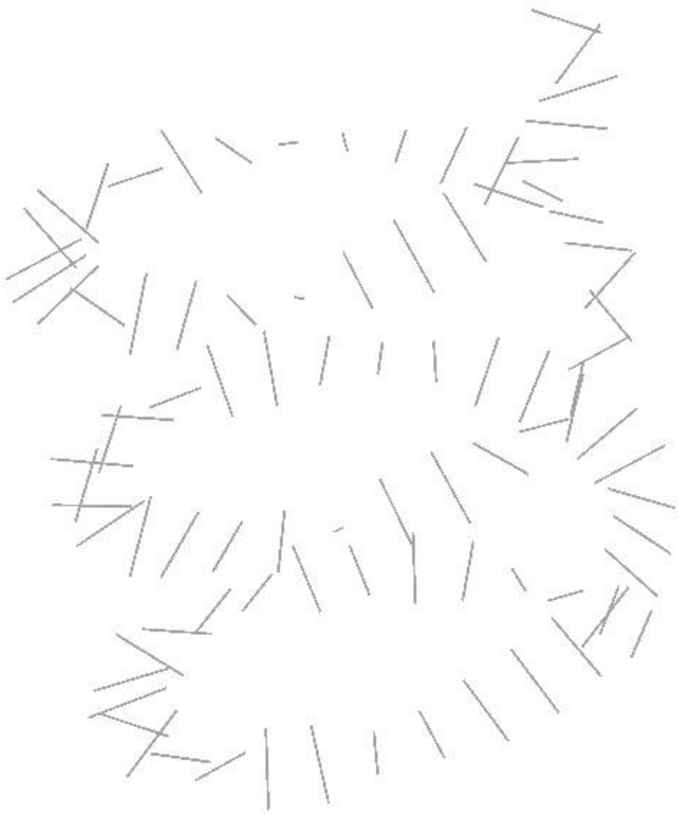

Figure 6. Result generated from the first prompt:
Single lines randomly oriented and arranged along an invisible spiral

As you can see, all we did was issue a single command, and the result was truly remarkable in the following points:

- The machine did a great job of separating the subject from the background in the two downloaded photos.
- It recognised the structure of the shape.
- It wrote the program code for Rhino Python.
- The program code was well structured; it was as clear as it is written in textbooks.
- He placed comments in the code to make it more readable and easier to study.
- In the code, he used the usual notations from the literature to indicate distances, angles, coordinates, etc.
- In the code, both the procedure itself and the parameters have "colloquial" names, so that the parameter name itself describes its function.
- The code is written in a parameterizable way, so the user can freely set the variables.
- He described in detail how to copy the code from the ChatGPT window, paste it into Rhino and run it there.
- It asked whether we wanted to make the lower part denser, indicating that it recognized this might be necessary.

Although the AI-generated code offered a solid starting point, it did not yet resemble the object depicted in our reference photographs. In the following, we explain how the script can be further developed by providing the AI with critical feedback and more specific design requirements.

It is worth comparing it with the spiral.py code we wrote in paragraph 2.

3.3. *AI created 3D spiral final resulting*

Reading the first response program code and looking at the finished model, you notice that the direction of the rods on the curve is random, even though it looks regular in the photo. So, we asked the AI to correct this. We also asked it to draw the curve, not just the rods. We also pointed out that the rods were only shown as lines, but we wanted real cylinders. It was already implied in the perspective picture, but from a top view it was clear that the curvature of the space curve does not change, it does not expand. Of course, we asked for this to be corrected as well. We made several requests at the same time, with points, and received the answers in the same structure, with the corrected code added.

If we were not satisfied with something, we wrote down in plain language, in round sentences, why we didn't like the result and what we would like to get instead. We tried to avoid jargon and very specific requests. We tried to act as lay users, simply stating our wishes for the acceptable result. So, we did not help the AI to decide which part of the program should be rewritten and how. After the seventh iteration, we got the result shown in Figure 7. and 8., which we were happy with:

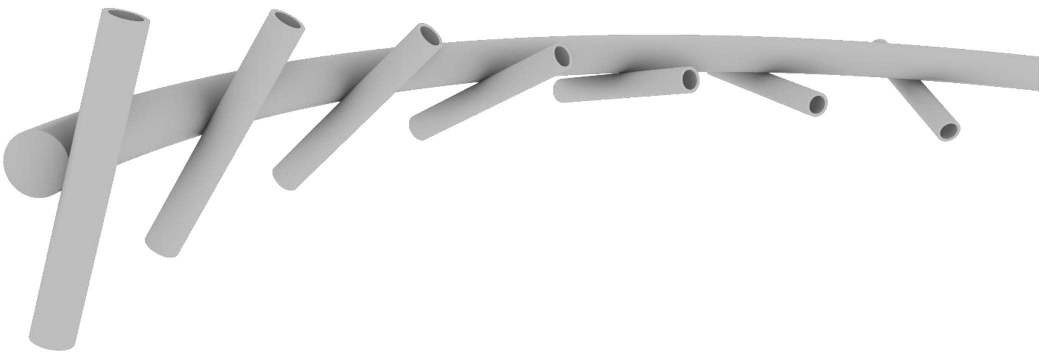

Figure 7. top part of the result after the seventh iteration.

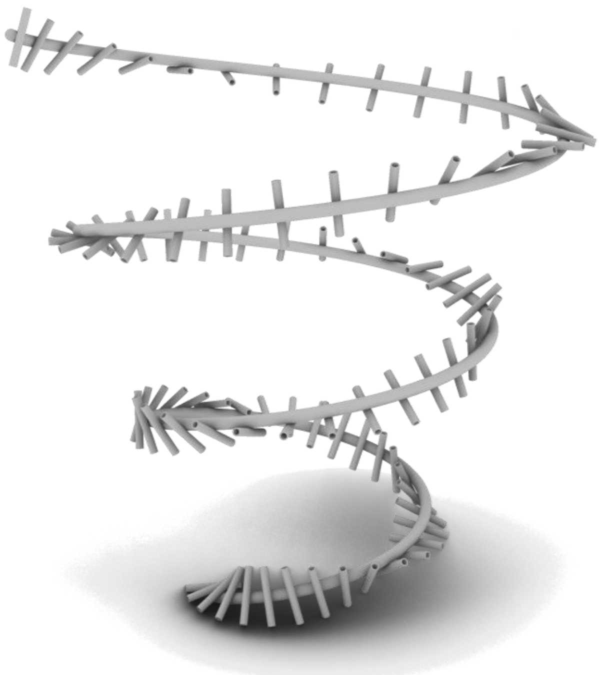

Figure 8. perspective result after the seventh iteration

During our tests we had issues in all three areas: python programming language, Rhino parameter, and geometric calculation. In our chosen spiral example, all these three problems appeared, so we needed our programming and 3D modelling skills to achieve the acceptable result three times.

Three times we had to make a request to the AI, describing in very concrete terms what part of the code needed to be rewritten or improved and in which parts. Probably at these three points, a completely lay user would have either got stuck or would have taken a long time to formulate and experiment with a prompt that would give the AI the correct result. In this spiral example, these 3 critical points were:

1. Basic programming language mistake: On one occasion, Rhino stopped due to a syntactic error, indicating that it had found a character in a line that was not in the ASCII character set. We looked up the specific line of code indicated and found that the error was in a comment. It was discovered that the AI had used a long dash instead of a dash. This is a strange error, because the ASCII character set is precisely defined in the standard. (This one compared to other problems, it could correct itself, based on only the error message! Not critical, but unusual since it is a basic programming language mistake.)
2. Rhino Command parameter mistake: The other point where we had to intervene by the end of the small cylinders attached to the spiral. First the AI made the only lines. We specified that we wanted volume, not curves. For that input it made a pipe with cap=2, meaning the ends of the small cylinders are closed with a half sphere. We than specified that we need flat cap and hollowed pipes. Answering that comment, it made a mistake by stating "We will use cap=0 in AddPipe() to get flat caps." The correct parameter for a flat cap is neither 2(round), nor 0 (none), but 1(flat). That means not only the rods still were not flat caped, but the code tried to apply Boolean operation on two open surfaces, which didn´t had any intersection, since they had the same axis but different diameter. This operation made the rods automatically disappear. Syntactically, the code was correct, it ran, but the acceptable result was a strong regression from the previous iteration step. Only after telling the exact parameter, did it manage to change the script accordingly.
3. Geometric mistake: The rods were not positioned correctly in the initial attempt. At first the spiral had intersected the rods at their midpoints. After we provided feedback stating that the rods should be placed next to the spiral, touching it at only a single point on their surfaces, the AI corrects the script. However, the transformation was applied in the wrong direction. The AI moved rods parallel to the axes of the roads, but not perpendicularly. Another issue in this iteration was that the offset was calculated using only the radius of the spiral pipe, rather than the correct distance, which should have included both the radius of the spiral pipe and the radius of the rod. At this stage, we had to precisely define, using mathematical terms, the vector direction we wanted and the required offset distance. Throughout the process, the syntax of the code was consistently correct, but the AI struggled to determine the proper geometric calculations without our intervention.

To summarize, ChatGPT performs excellently in the Python programming language, but is more often wrong in Rhino-specific commands. In all cases, we were satisfied with the speed of the answers, the detail of the results and the clarity of the explanations.

4. CONCLUSION AND RECOMMENDATIONS

In conclusion: As can be seen in the example in the previous section, but our previous tests have also confirmed that AI is a great help in coding, but it not a substitute for humans (but we cannot expect this when solving such a complex, constructive problem.)

Based on our experience, our recommendations in this area are as follows:

- The first prompt rarely gives the desired result, so further refinements will often be needed. Accept this, as well as the fact that sometimes refinement is more of a step backwards in the solution.
- Let us formulate all our expectations at the first prompt. It is not a problem if you have a lot of expectations, just put them into clear points. Then we will need fewer iterations to get to the acceptable result.
- If we have a working and structured code snippet that solves a similar task, it is worth uploading it to the AI. (We didn't do this; the purpose of the testing was exactly to measure how the AI performs based on the photo). After uploading, it is a good idea to ask the AI to analyse, complete, rewrite the code to give us the result we want.

- If Rhino indicates a syntactic error in the code, copy it to the AI prompt window. Complete the prompt with your own thoughts and ask for a correction.
- In the event of an error, ask the AI to explain how the critical part works. This will help our debugging, but it is also easy for the AI to find out the nature of the error.
- Even if the result is acceptable, ask the AI if it has any suggestions to simplify, improve or make the code more efficient.
- Also ask how the code could be extended, for example, to be more parameterizable and to solve more general tasks.
- Finally, it's worth being polite and nice, because then the AI will take on a similar style and you can work with it in a more pleasant atmosphere.

In this article, we chose the sample task described in section 3.2 because the solution process and our experiences can be considered typical in other cases as well. Other architects have used different AI, tested it with different types of tasks, but their experiences are similar to ours. [6] [7] These results will likely determine the direction of further development of AI systems.

REFERENCES

[1] **LEVENTE GYULAI, VILMOS KATONA,** A Methodological Overview of Parametrici, *Symmetry: Culture and Science, Volume 31 (2020) No. 3, pages 365-381* https://doi.org/10.26830/symmetry_2020_3_365

[2] **ZHI XIAN CHEW, JING YING WONG, YU HOE TANG, CHUN CHIEH YIP, TOMAS MAUL,** Generative Design in the Built Environment, *Automation in Construction, Volume 166 (2024) pages 105638 https://doi.org/10.1016/j.autcon.2024.105638*

[3] **NGO CONG-LEM, ALI SOYOOF & DIKI TSERING,** A Systematic Review of the Limitations and Associated Opportunities of ChatGPT, *International Journal of Human–Computer Interaction Volume 41 (2025) Issue 7* https://doi.org/10.1080/10447318.2024.2344142

[4] https://www.katonajanos.hu/Spiral.py

[5] https://www.katonajanos.hu/Lamp_v1.py

[6] **DUYGU SAVUR, ŞEYDA EMEKCI,** Evaluating Generative AI Capabilities for Architectural Visualization, *4th International Civil Engineering & Architecture Conference (2025)* https://www.researchgate.net/publication/392194919_Evaluating_Generative_AI_Capabilities_for_Architectural_Visualization

[7] **TAMÁS LUKOVICH,** Artifical Intelligence and Architecture Towards a New Paradigm, *Ybl Journal of Built Environment, Volume 8. Issue 1. (2023)* https://doi.org/10.2478/jbe-2023-0003