MCP-Driven Parametric Modeling: Integrating LLM Agents into Architectural and Landscape Design Workflows

Xun Liu
University of British Columbia, School of Architecture and Landscape Architecture (SALA)
xliu@sala.ubc.ca

Runjia Tian
Generative Game Inc.
runjiatian@gmail.com

Abstract

We present a novel integration of the Model Context Protocol (MCP) with Grasshopper, enabling Large Language Models to directly interact with parametric modeling workflows for architectural and landscape design. This system allows designers to prompt, iterate, and refine 3D models conversationally through structured symbolic generation, bridging human creative intent and computational form generation. The framework employs a client-server architecture where natural language instructions are parsed into structured commands invoking modular parametric components. Demonstrated in a 9-day DigitalFUTURES workshop with 20 participants across 5 teams, each team developed distinct parametric design lexicons encoding specialized domain knowledge—from generative open spaces to urban functional zoning—that design novices could subsequently operate through natural language interfaces. Beyond accessibility improvements, the protocol-based architecture enables workflow merging through systematic integration of diverse design components into composable ecosystems. We present the system architecture, implementation details, and empirical observations from workshop deployments demonstrating how the approach addresses the theme of *Humanity* through expanded creative agency and knowledge democratization.

1 Introduction

Parametric modeling has become central to contemporary architectural and landscape design, offering procedural control over complex geometries. Yet interacting with these systems typically requires specialized technical knowledge and manual manipulation of component networks, limiting accessibility and slowing early-stage ideation.

Our project introduces an MCP–Grasshopper integration that allows LLMs to act as intelligent design agents, transforming natural language instructions into structured parametric modeling operations. This approach implements what we term *computational lexicalization*—the systematic restructuring of design knowledge into a structured language that is both human-readable (semantic) and machine-executable (computational). The framework consists of a *lexicon* of discrete, reusable design units and a *grammar* that organizes these units into coherent workflows.

This methodology was explored in the DigitalFUTURES July 2025 workshop titled "Landscape Agents," where 20 participants across 5 teams engaged in intensive prototyping. Teams developed distinct approaches ranging from ecological landscape generation to urban functional zoning systems,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Creative AI Track.

demonstrating the framework's versatility. The workflow shifts the designer's role from manual assembly to strategic direction, allowing greater focus on conceptual exploration while maintaining precise parametric control.

2 Background and Motivation

2.1 The Cognitive Gap in Parametric Design

The computational turn in architecture has brought powerful parametric and algorithmic design tools, yet it has also introduced a fundamental cognitive gap. Designers rely on intuition, ambiguity, and iterative thinking during conceptualization, while computational tools demand precision, linearity, and formalized algorithmic logic. This mismatch creates barriers to entry and slows creative exploration.

2.2 Three Pathways for AI-Driven 3D Generation

Current generative AI approaches to 3D design can be categorized into three distinct pathways. **Voxel/Pixel-based Generation** using 3D GANs generates geometry by processing voxelized 3D space but suffers from low precision and poor compatibility with BIM workflows. **Mesh/NeRF-based Generation** using diffusion models produces visually impressive results but generates "dead" geometry lacking construction logic and editability. **Symbolic/Lexical Generation** generates executable domain-specific languages or modeling instruction sequences rather than direct geometry, representing a deep fusion of language models with rule-based parametric systems.

We chose symbolic generation because it preserves design intent, maintains editability, supports lightweight data processing, and integrates seamlessly with existing BIM workflows. The generated structured information can be viewed as intelligent, pre-emptive BIM data creation.

2.3 Related Work

Recent work in AI-assisted design has explored various integration strategies. Xu et al. (2025) developed FUGenerator for multimodal architectural generation, while Ko et al. (2023) focused on automated spatial layout planning. However, most existing systems either lack direct integration with professional tools or remain limited to narrow design tasks. Our MCP-based approach provides a generalizable, open protocol for connecting AI reasoning capabilities with diverse parametric design environments.

3 System Design and Technical Implementation

3.1 Overall Architecture

The system architecture consists of four tightly integrated layers: natural language input where users formulate conversational prompts; LLM agent interpretation parsing unstructured input into structured commands; MCP protocol providing real-time bridge between AI agent and Grasshopper; and parametric execution generating models with optional visualization. Figure 1 illustrates this complete workflow.

3.2 Natural Language Processing and LLM Interpretation

The workflow begins with designers formulating requests in conversational language, from high-level spatial concepts to precise geometric instructions. We employ a hierarchical prompt structure where system-level prompts define the AI agent's role with explicit functional scope, while stage-specific prompts provide refined instructions for different design phases.

The LLM parses unstructured inputs through component identification (identifying relevant Grasshopper components), operation mapping (translating descriptive language to geometric operations), parameter assignment (extracting numerical values), and relationship inference (determining spatial relationships between entities). Domain-specific terminology is handled through specialized

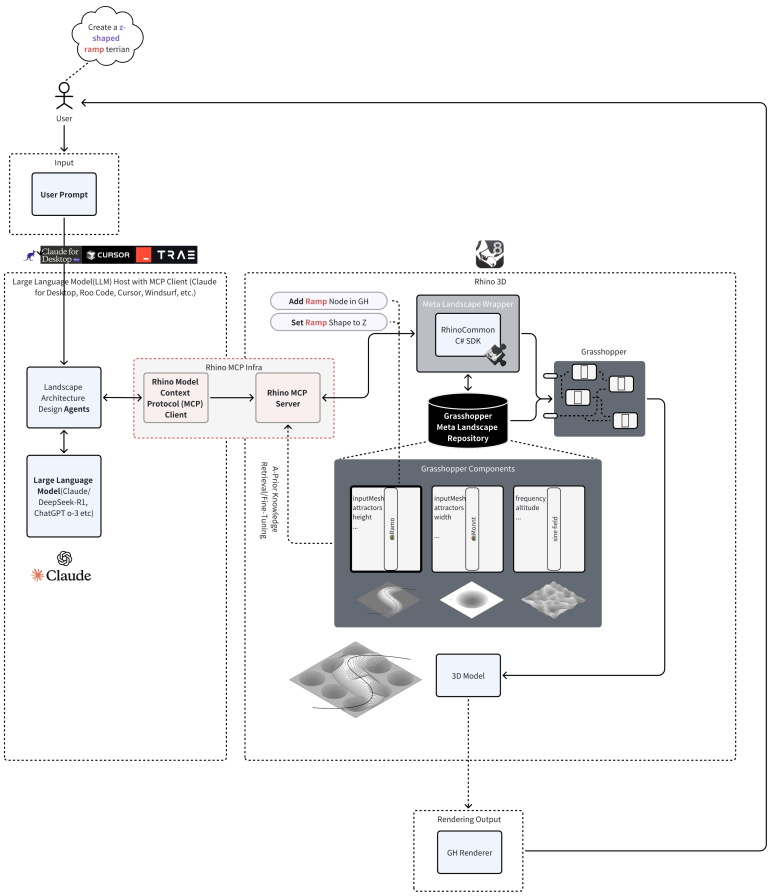

Figure 1: Complete system workflow showing natural language prompts processed by LLM agents through the Rhino MCP Client, sent to the Rhino MCP Server interfacing with Grasshopper Meta Landscape Repository containing modular parametric components.

examples and component libraries during system initialization, improving accuracy in design interpretation.

3.3 Model Context Protocol Integration

The Model Context Protocol serves as a standardized communication layer between the AI agent and design software. The architecture consists of the MCP Host (LLM) functioning as the agent's brain for task planning, the MCP Client acting as middleware formatting requests and managing user approval, and the MCP Server running on Rhino/Grasshopper managing the component library and executing operations.

A complete interaction cycle proceeds from user design request to LLM reasoning, then to client formatting and approval, followed by server execution and result return, finally integrating into a natural language response. This protocol-based architecture decouples LLM reasoning from software-specific implementation and creates an open ecosystem where new tools integrate by implementing the standard protocol. The system deploys using Rhino 8.0 with Python 3.9+, with server-side integration through RhinoCommon SDK enabling direct creation and connection of Grasshopper components, real-time parameter adjustment, and dynamic restructuring of parametric graphs without manual intervention.

3.4 Component Encapsulation and Lexicon Development

Rather than exposing all atomic Grasshopper APIs which would cause information overload, we developed an extensible MCP Server SDK allowing users to encapsulate component combinations into higher-level ”lexical units” with clear semantic descriptions. Each encapsulated tool receives structured JSON parameters from the MCP client and maps them to specific Grasshopper inputs. Once updated, Rhino instantly produces 3D geometry viewable directly, routable to performance simulations, or processable through neural rendering for rapid conceptual visualization.

4 Workshop Demonstration: Five Distinct Approaches

4.1 Workshop Context and Methodology

The DigitalFUTURES ”Landscape Agents” workshop engaged 20 participants organized into 5 teams over 9 intensive days. Participants included students and professionals with varying parametric design experience. Each team developed unique parametric design workflows—creating specialized ”design lexicons” that novices could operate through natural language interfaces. The diversity of approaches demonstrated the framework’s adaptability to different design philosophies and typologies.

4.2 Team Implementations and Design Lexicons

Team 1: Generative Open Spaces. This team developed a comprehensive workflow for creating parametric open spaces using attractor-based path generation. Their lexicon included components like `Create_Bounds`, `Random_Attractor_Generator`, and `Random_Entrances_and_Exits_Placer`. The system automatically handled entrance/exit placement and path finding through fields generated by attraction and repulsion points. Their prompt engineering focused on iterative refinement, with logic for varying seed values to generate multiple design iterations rapidly.

Team 2: Ecological Landscape Systems. Focusing on terrain generation with ecological parameters, this team created a sophisticated workflow incorporating topographic manipulation, water systems, and vegetation distribution. Their components included `Site_Boundary`, `WoolAlgorithmCalculator` for path simulation, and specialized modules for `Linear_Plants`, `Scattered_Plants`, and `Shrubs`. The system intelligently distributed vegetation based on slope analysis and moisture conditions, with tree density parameters inversely correlated to visual density (1 representing maximum density, 9 minimum).

Team 3: Urban Functional Zoning. This team developed an urban block classification system that analyzed plot centroids to determine functional types based on directional logic. Their workflow extracted coordinate data to classify plots as commercial, residential, administrative, or park spaces, then instantiated corresponding MCP Geometry Group components. The system interpreted fuzzy directional commands (”Most East,” ”Southwest”) as relational degrees rather than absolute zones, demonstrating sophisticated spatial reasoning.

Team 4: Urban Park Design. Creating a flexible urban park generation system, this team focused on modular space creation with components for main spaces, road networks, green spaces, and facilities. Their workflow allowed adjustment of parameters like road width (0.5-5 meters for different circulation types), green space ratio (ecological >50

Team 5: Adaptive Landform Modifiers. This team explored terrain modification through sequential application of modifiers. Their workflow emphasized the stacking of multiple disturb geometries (curves, points, rivers, geometric shapes) with corresponding modifiers applied in sequence. The system maintained terrain continuity by connecting outputMesh from one modifier to inputMesh of the next, creating complex layered landscapes. Visualization components included gradient rendering and voxel previews for immediate design feedback.

4.3 Empirical Observations and Outcomes

Student works are shown in Figure 2. Throughout the workshop, several critical patterns emerged. Teams that invested time in creating comprehensive component libraries with clear semantic descrip-

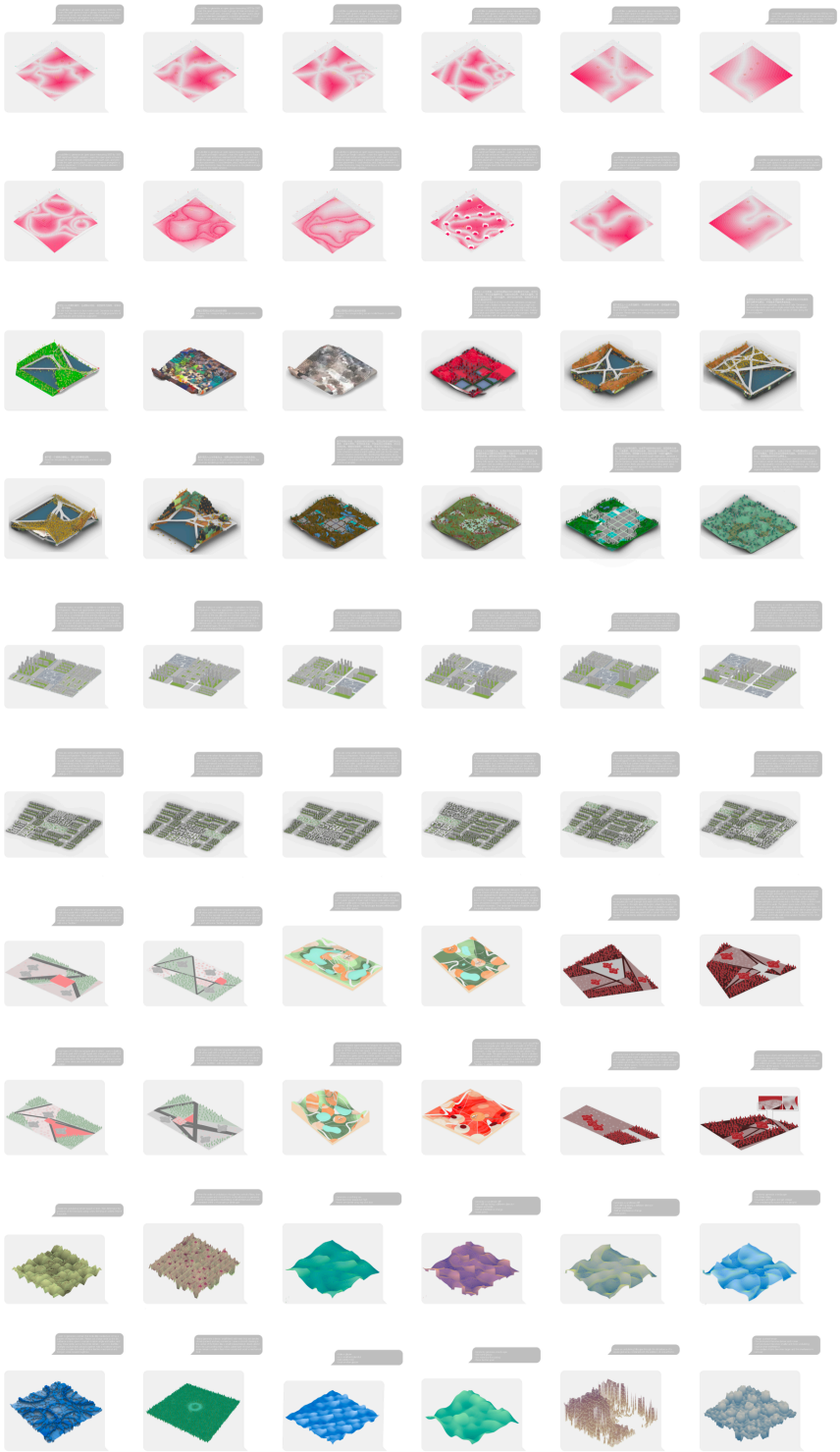

Figure 2: Complete Student Work

tions achieved significantly higher success rates in AI interpretation. The process of encoding design knowledge into lexicons forced participants to articulate implicit design rules explicitly, deepening their understanding of parametric logic.

Natural language interfaces enabled remarkable acceleration in design iteration, with teams reporting 20-40 second cycles from prompt to generated result—approximately 10x faster than manual Grasshopper manipulation. More importantly, by workshop end, design novices could operate sophisticated parametric workflows created by experienced users, effectively democratizing access to computational design tools. The quality of AI-generated results was directly proportional to lexicon sophistication, with well-documented components producing more consistent and creative outcomes.

5 Discussion: AI's Role and Human Agency

AI plays multiple integrated roles throughout this framework, functioning not as replacement for human creativity but as cognitive amplifier. At the interpretive stage, LLMs process unstructured language into precise modeling instructions while making intelligent decisions about parameter values and geometric relationships. The framework maintains human agency by requiring designers to define the lexicon—essentially teaching the AI their design philosophy—while the AI handles execution and variation generation.

This addresses the NeurIPS 2025 Creative AI theme of *Humanity* by expanding creative access to computational design while placing human judgment at the center. The conversational interface enables designers to engage parametric modeling through strategic guidance rather than technical execution. In educational contexts, learners can focus on spatial logic and design intent rather than software mechanics. In professional contexts, it enables collaborative design where domain experts contribute specialized lexicons that non-experts can utilize, creating a more inclusive design process.

6 Conclusion and Future Directions

We have introduced an MCP–Grasshopper integration for LLM-driven parametric modeling, demonstrated through intensive workshop deployment with diverse team implementations. The system transforms parametric design from technical task into conversational process, enabling rapid development of custom design lexicons that novices can operate through natural language.

Beyond lowering technical barriers, this work points toward workflow merging through protocol unification. The shared MCP protocol allows lexicons from multiple teams to combine, creating sophisticated design pattern libraries encoding collective knowledge. Imagine landscape architects contributing terrain manipulation lexicons, structural engineers adding load-bearing patterns, and environmental consultants providing climate-responsive libraries—all accessible through unified conversational interface.

Future work will explore workflow merging capabilities, developing protocols for lexicon versioning and collaborative vocabulary development. We will refine domain-specific AI agents, incorporate 3D-aware reasoning, and explore mixed-reality interfaces. By combining human-centered design values with AI capabilities and protocol-based interoperability, this approach points toward more inclusive, dynamic, and collaborative creative environments in architecture and landscape architecture.

Acknowledgments

We thank DigitalFUTURES 2025 workshop participants at Tongji University, particularly the five teams whose diverse approaches validated the framework's versatility. We acknowledge Anthropic for developing the Model Context Protocol specification.

References

[1] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.

[2] M. Carpo. *The Second Digital Turn: Design Beyond Intelligence*. The MIT Press, Cambridge, MA, 2017.

[3] J. Ko, B. Ennemoser, W. Yoo, W. Yan, and M. J. Clayton. Architectural spatial layout planning using artificial intelligence. *Automation in Construction*, 154:105019, 2023.

[4] X. Liu, N. Yang, and R. Tian. Reinventing planting design in landscape architecture: A generative AI approach. *Journal of Digital Landscape Architecture*, pages 202–209, 2024.

[5] Anthropic et al. Model Context Protocol Specification, 2024. `https://modelcontextprotocol.io/specification/latest`.

[6] W. J. Mitchell. *The Logic of Architecture: Design, Computation, and Cognition*. The MIT Press, Cambridge, MA, 1990.

[7] R. Woodbury. *Elements of Parametric Design*. Routledge, London, 2010.

[8] X. Xu, T. Feng, Y. Zhang, Z. He, and P. F. Yuan. FUGenerator: multimodal-AI platform for architectural design. *Architectural Intelligence*, 4(1):1–15, 2025.

A Workshop Implementation Details

The following provides additional technical detail on the five team implementations from the Digital-FUTURES 2025 ”Landscape Agents” workshop, demonstrating the diversity of approaches enabled by the MCP-Grasshopper framework.

A.1 Team 1: Generative Open Spaces

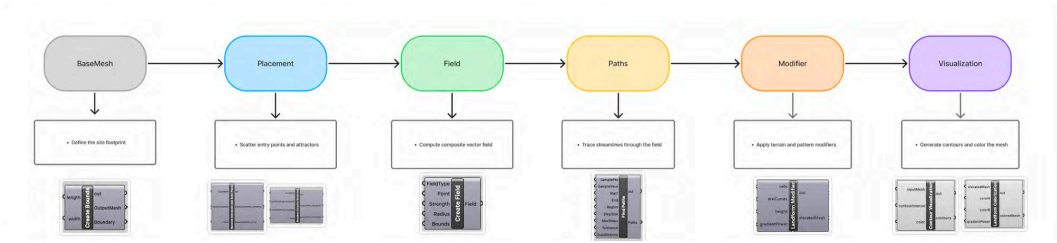

Figure 3: Team 1’s generative open space workflow showing attractor-based path generation with multiple entrance/exit configurations and field-based path finding algorithms. The system demonstrates emergent circulation patterns from simple attractor and repulsor rules.

A.1.1 System Prompt Structure

Team 1’s system prompt emphasized iterative exploration through seed variation. Key components of their prompt engineering included:

- **Component Discovery:** Always start by retrieving all available Grasshopper components using `get_all_component_proxies`
- **Default Parameters:** If users don’t specify dimensions, default to 3000x3000 units for the base bounds
- **Error Handling:** Issue warnings for dimensions below 1000 units that might cause geometric errors
- **Iteration Logic:** Automatically assign random integers between 1-1000 for seed values to enable rapid variation generation

Their prompt validated that the number of generated entrances matched user specifications, implementing a feedback loop that requested parameter adjustments when constraints weren’t satisfied.

A.2 Team 2: Ecological Landscape Systems

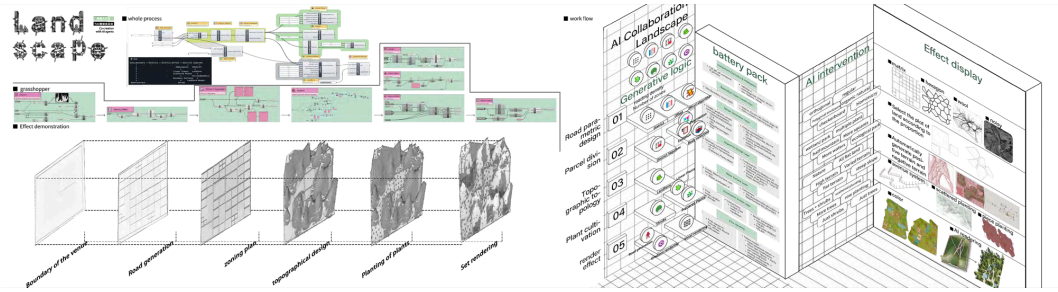

Figure 4: Team 2's ecological landscape system featuring terrain generation with integrated water systems, slope-based vegetation distribution, and multi-layered ecological parameters. The visualization shows contour-based terrain representation with water bodies and vegetation clusters.

A.2.1 Component Architecture

Team 2 developed sophisticated ecological logic with components including:

- `Site_Boundary`: Base terrain definition with size parameters
- `WoolAlgorithmCalculator`: Path simulation with toggle and reset mechanisms
- `Linear_Plants, Scattered_Plants, Shrubs`: Vegetation modules with slope sensitivity
- `WaterSurface`: Hydrological system integration

Their inverse density parameter system meant lower values indicated higher planting density (1 = maximum, 9 = minimum). Different tree species were assigned based on terrain conditions: Melia for flat areas, Camphor for slight slopes, Tung for steep terrain.

A.3 Team 3: Urban Functional Zoning

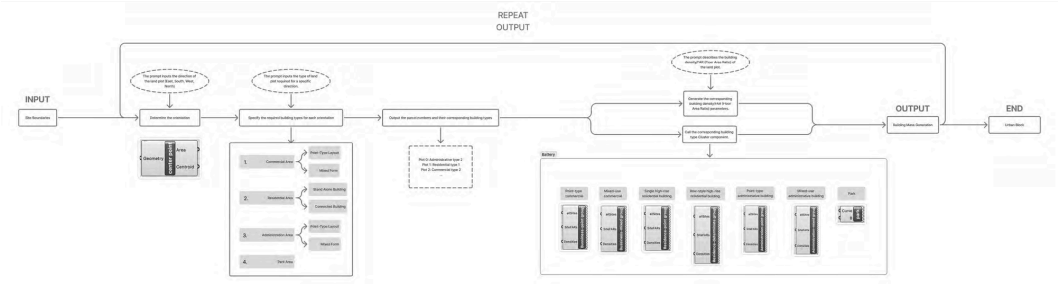

Figure 5: Team 3's urban functional zoning system showing coordinate-based plot classification and automatic generation of corresponding building typologies. Different colors represent different functional zones (commercial, residential, administrative, park).

A.3.1 Directional Logic Implementation

Team 3's coordinate analysis system interpreted spatial relationships:

- Higher X values indicated more Eastern positions
- Higher Y values indicated more Northern positions
- Fuzzy commands like "Most East" were interpreted as max(X)
- "Southwest" was interpreted as low X and low Y relative to median values

The system then instantiated corresponding MCP Geometry Group components based on classification: Business-point group for point commercial, Residence-mixed group for townhouses, etc.

A.4 Team 4: Urban Park Design

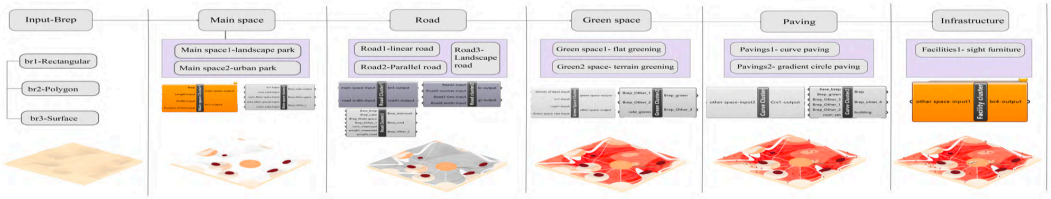

Figure 6: Team 4's modular urban park design system with flexible components for circulation, green space, facilities, and decorative paving patterns. The image shows various configurations of park elements with different green space ratios and circulation patterns.

A.4.1 Parameter Definitions

Team 4 established clear parameter ranges for urban park elements:

- **Green Space Ratio:** Ecological (>50%), Urban (<35%), Standard (35-50%)
- **Road Width:** Main roads (2-5m), Narrow paths (0.5-2m), Maximum (6m)
- **Tree Density:** High (>45%), Low (<20%), Standard (20-45%)
- **Paving Patterns:** Circle paving with adjustable size, count, and distribution

A.5 Team 5: Adaptive Landform Modifiers

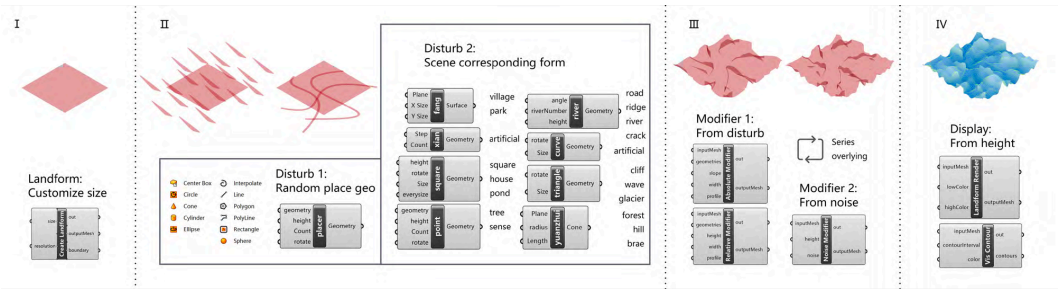

Figure 7: Team 5's adaptive landform modification system featuring stackable terrain operations with gradient visualization and voxel-based analysis. The workflow shows progressive terrain transformation through multiple modifier applications.

A.5.1 Modifier Stacking Methodology

Team 5's workflow emphasized sequential terrain modification:

1. **BaseMesh Creation:** Initial terrain grid with resolution and size parameters
2. **Disturb Geometries:** Multiple types (curves, points, rivers, shapes) as influence factors
3. **Modifier Chain:** Each modifier's outputMesh connects to the next modifier's inputMesh
4. **Noise Application:** Always add noise modifier before final visualization
5. **Visualization:** Gradient rendering and voxel previews for immediate feedback