MULTIMAT

Multimodal Program Synthesis for Procedural Materials using Large Multimodal Models

Jonas Belouadi **Tamy Boubekeur** **Adrien Kaiser**

Adobe Research University of Mannheim

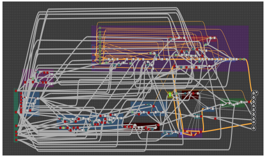
March 30, 2026

WHAT ARE PROCEDURAL MATERIALS?

Procedural materials are programs that offer powerful control over the appearance of 3D objects.

WHAT ARE PROCEDURAL MATERIALS?

Procedural materials are programs that offer powerful control over the appearance of 3D objects.

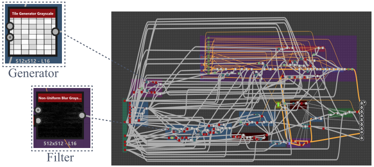


Procedural Material Graph

- acyclic node graphs that describe appearances

WHAT ARE PROCEDURAL MATERIALS?

Procedural materials are programs that offer powerful control over the appearance of 3D objects.

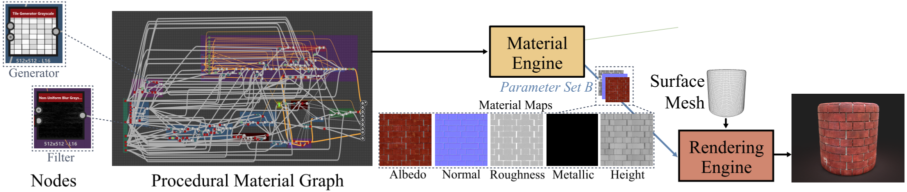


Nodes Procedural Material Graph

- acyclic node graphs that describe appearances
    - nodes represent texture generators or filtering operations

WHAT ARE PROCEDURAL MATERIALS?

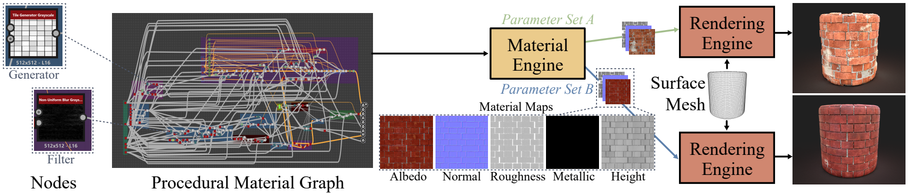
Procedural materials are programs that offer powerful control over the appearance of 3D objects.



- acyclic node graphs that describe appearances
  - nodes represent texture generators or filtering operations
- graphs output material maps (e.g., albedo, normal, roughness, height)

WHAT ARE PROCEDURAL MATERIALS?

Procedural materials are programs that offer powerful control over the appearance of 3D objects.



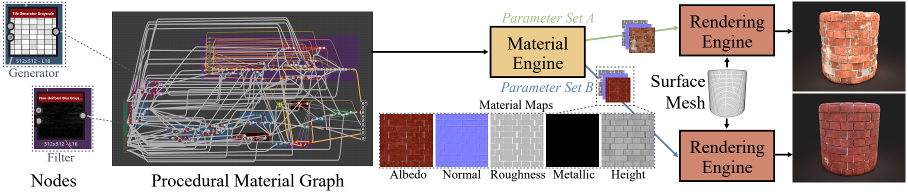
- acyclic node graphs that describe appearances
    - nodes represent texture generators or filtering operations
- graphs output material maps (e.g., albedo, normal, roughness, height)
- compared to image-based maps support arbitrary resolution and high-level parametric appearance control

Adobe

UNIVERSITY OF MANNHEIM

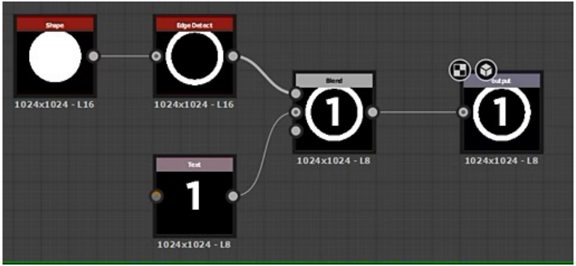
WHAT ARE PROCEDURAL MATERIALS?

Procedural materials are programs that offer powerful control over the appearance of 3D objects.



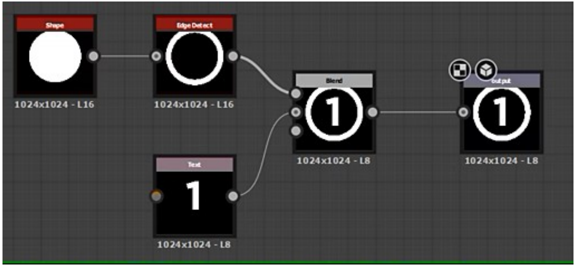
- acyclic node graphs that describe appearances
  - nodes represent texture generators or filtering operations
- graphs output material maps (e.g., albedo, normal, roughness, height)
- compared to image-based maps support arbitrary resolution and high-level parametric appearance control
- we target the Adobe Substance Designer node graph system

PROPERTIES OF PROCEDURAL MATERIALS



- humans edit procedural material programs visual-spatially

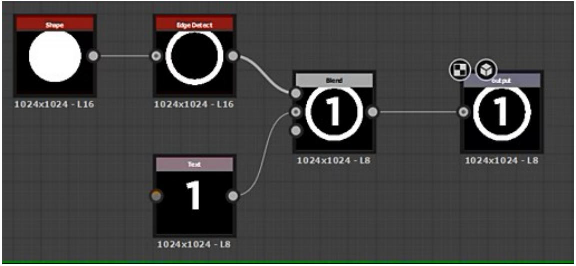
PROPERTIES OF PROCEDURAL MATERIALS



- humans edit procedural material programs visual-spatially
  - different from text-based programming paradigms

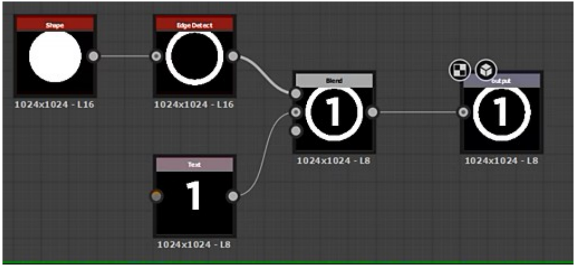
Adobe UNIVERSITY OF MANNHEIM

PROPERTIES OF PROCEDURAL MATERIALS



- humans edit procedural material programs visual-spatially
  - different from text-based programming paradigms
- node graphs are compositional

PROPERTIES OF PROCEDURAL MATERIALS



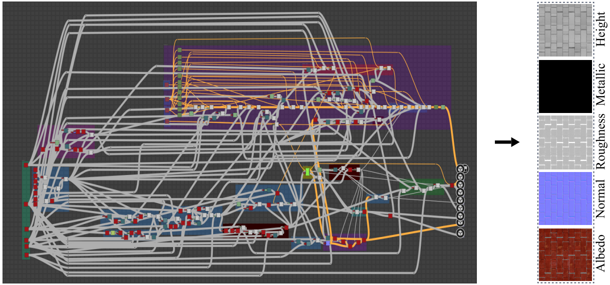
- humans edit procedural material programs visual-spatially
  - different from text-based programming paradigms
- node graphs are compositional
  - if we traverse graphs topologically we can inspect intermediate states

Adobe

UNIVERSITY OF MANNHEIM

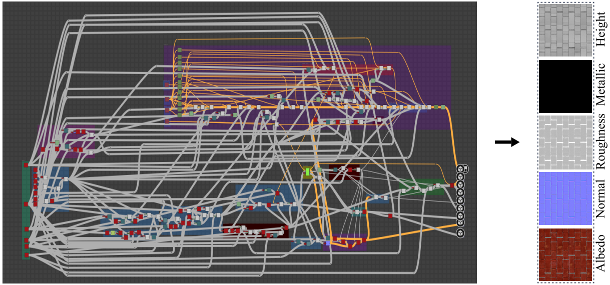
THE PROBLEM

- creating these node graphs manually can become challenging very fast



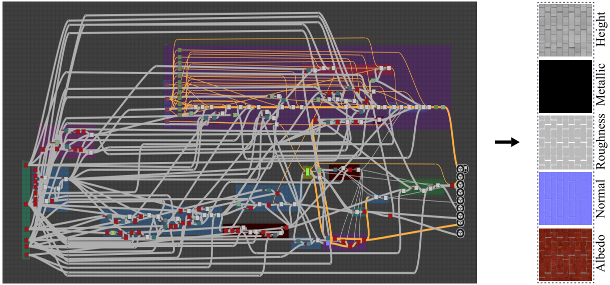
THE PROBLEM

- creating these node graphs manually can become challenging very fast
- requires professional training



THE PROBLEM

- creating these node graphs manually can become challenging very fast
- requires professional training
- automated synthesis of procedural materials (e.g., conditioned on images) could aid users



Adobe UNIVERSITY OF MANNHEIM

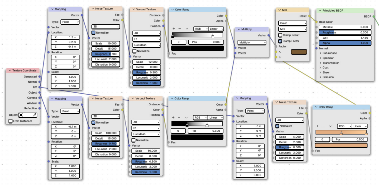
RESEARCH GAP

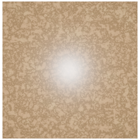
Input → VLM → Python code → Execute → Shader node graph → Render → Output

```python
import bpy

def shader_material(material: bpy.types.Material):
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Create nodes
    material_output = nodes.new('ShaderNodeOutputMaterial')
    mix_shader = nodes.new('ShaderNodeMixShader')
    principled_bsdf = nodes.new('ShaderNodeBsdfPrincipled')
    mix = nodes.new('ShaderNodeMixRGB')
    multiply = nodes.new('ShaderNodeVectorMath')
    colorramp = nodes.new('ShaderNodeValToRGB')
    voronoi_texture = nodes.new('ShaderNodeTexVoronoi')
    noise_texture = nodes.new('ShaderNodeTexNoise')
    mapping = nodes.new('ShaderNodeMapping')
    texture_coordinate = nodes.new('ShaderNodeTexCoord')
    colorramp_1 = nodes.new('ShaderNodeValToRGB')
```





- past approaches (e.g., VLMaterial) generate materials solely as textual programs

Source: Li, Beichen, et al. VLMaterial: Procedural material generation with large vision-language models. ICLR 2025

Adobe

UNIVERSITY OF MANNHEIM

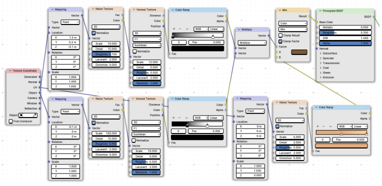
RESEARCH GAP

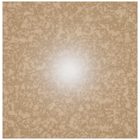
Input → VLM → Python code → Execute → Shader node graph → Render → Output

```python
import bpy

def shader_material(material: bpy.types.Material):
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Create nodes
    material_output = nodes.new('ShaderNodeOutputMaterial')
    mix_shader = nodes.new('ShaderNodeMixShader')
    principled_bsdf = nodes.new('ShaderNodeBsdfPrincipled')
    mix = nodes.new('ShaderNodeMixRGB')
    multiply = nodes.new('ShaderNodeVectorMath')
    colorramp = nodes.new('ShaderNodeValToRGB')
    voronoi_texture = nodes.new('ShaderNodeTexVoronoi')
    noise_texture = nodes.new('ShaderNodeTexNoise')
    mapping = nodes.new('ShaderNodeMapping')
    texture_coordinate = nodes.new('ShaderNodeTexCoord')
    colorramp_1 = nodes.new('ShaderNodeValToRGB')
```





- past approaches (e.g., VLMaterial) generate materials solely as textual programs
- fails to capture inherent visual-spatial nature that makes them accessible to humans

Source: LI, Beichen, et al. VLMaterial: Procedural material generation with large vision-language models. ICLR 2025

Adobe UNIVERSITY OF MANNHEIM

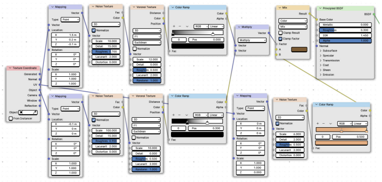
RESEARCH GAP

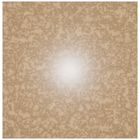
Input —VLM→ Python code —Execute→ Shader node graph —Render→ Output

```
import bpy

def shader_material(material: bpy.types.Material):
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Create nodes
    material_output = nodes.new('ShaderNodeOutputMaterial')
    mix_shader = nodes.new('ShaderNodeMixShader')
    principled_bsdf = nodes.new('ShaderNodeBsdfPrincipled')
    mix = nodes.new('ShaderNodeMixRGB')
    multiply = nodes.new('ShaderNodeVectorMath')
    colorramp = nodes.new('ShaderNodeValToRGB')
    voronoi_texture = nodes.new('ShaderNodeTexVoronoi')
    noise_texture = nodes.new('ShaderNodeTexNoise')
    mapping = nodes.new('ShaderNodeMapping')
    texture_coordinate = nodes.new('ShaderNodeTexCoord')
    colorramp_1 = nodes.new('ShaderNodeValToRGB')
```



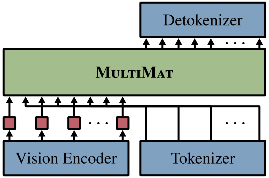


- past approaches (e.g., VLMaterial) generate materials solely as textual programs
- fails to capture inherent visual-spatial nature that makes them accessible to humans
- we aim to bridge the gap between procedural material synthesis and the human workflow

Source: LI, Beichen, et al. VLMaterial: Procedural material generation with large vision-language models. ICLR 2025

PRESENTING MULTIMAT

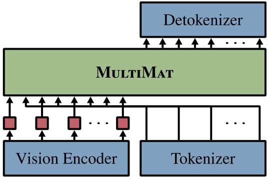
We present MULTIMAT, a multimodal program synthesis VLM for procedural materials with a visual memory.



PRESENTING MULTIMAT

We present MultiMat, a multimodal program synthesis VLM for procedural materials with a visual memory.

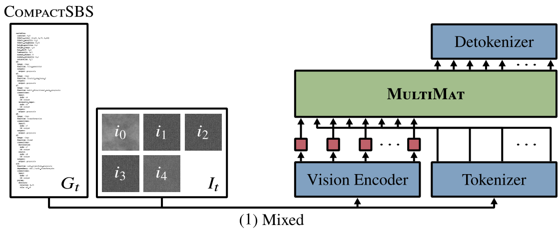
- previous nodes represented visually as context



PRESENTING MULTIMAT

We present MULTIMAT, a multimodal program synthesis VLM for procedural materials with a visual memory.

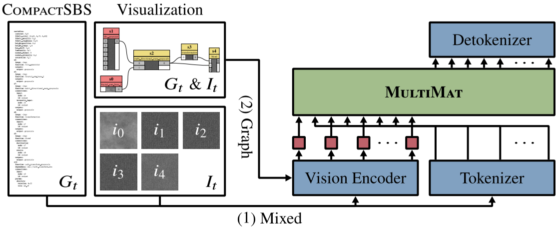
- previous nodes represented visually as context
  - mixed: Node definitions interleaved with visual representations of intermediate states



PRESENTING MULTIMAT

We present MultiMat, a multimodal program synthesis VLM for procedural materials with a visual memory.

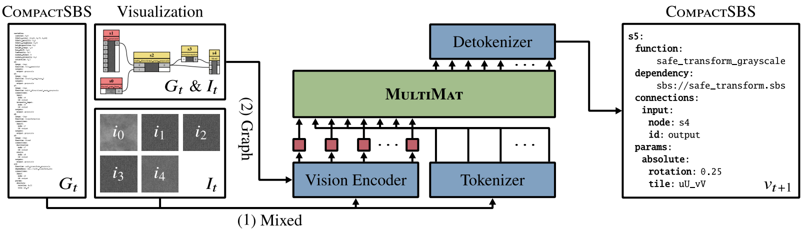
- previous nodes represented visually as context
  - mixed: Node definitions interleaved with visual representations of intermediate states
  - graph: As visualized graphs like in human-targeted user interfaces



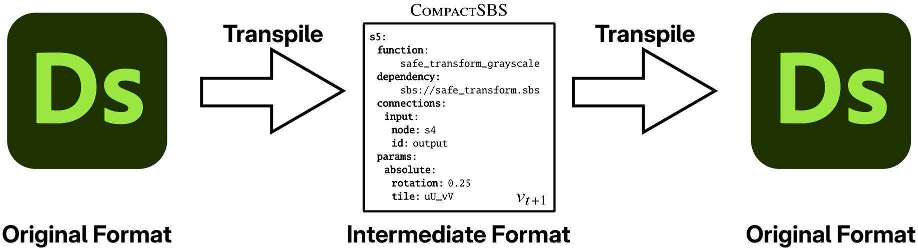
PRESENTING MULTIMAT

We present MULTIMAT, a multimodal program synthesis VLM for procedural materials with a visual memory.

- previous nodes represented visually as context
  - mixed: Node definitions interleaved with visual representations of intermediate states
  - graph: As visualized graphs like in human-targeted user interfaces
- iteratively generates nodes in topological order and updates context

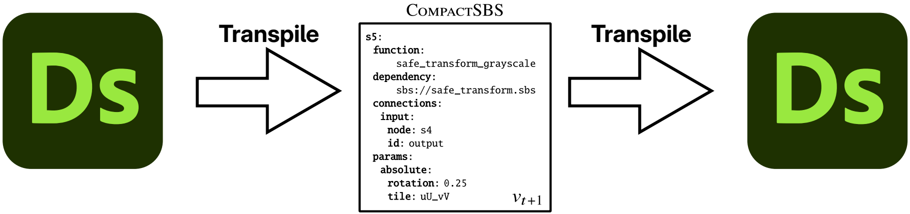


CUSTOM INTERMEDIATE REPRESENTATION



- MultiMat generates outputs in an intermediate representation

CUSTOM INTERMEDIATE REPRESENTATION

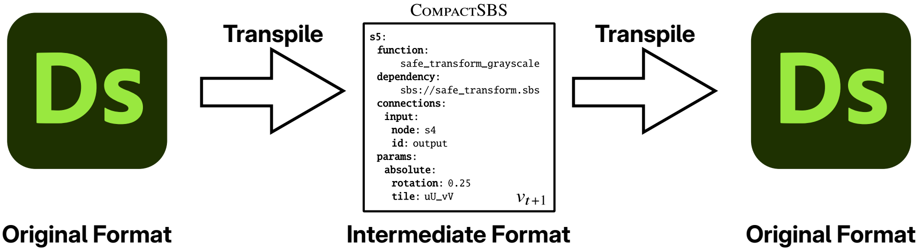


Original Format **Intermediate Format** **Original Format**

- MultiMat generates outputs in an intermediate representation
  - original Substance Designer format not suitable for language modeling

Adobe UNIVERSITY OF MANNHEIM

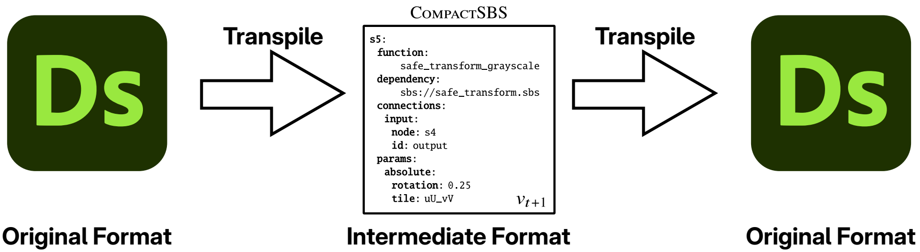
CUSTOM INTERMEDIATE REPRESENTATION



- MultiMat generates outputs in an intermediate representation
  - original Substance Designer format not suitable for language modeling
- we define a new, human-readable format that is more than 80% shorter

Adobe UNIVERSITY OF MANNHEIM

CUSTOM INTERMEDIATE REPRESENTATION

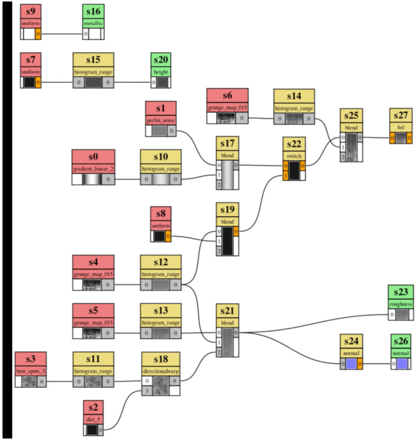


- MultiMat generates outputs in an intermediate representation
  - original Substance Designer format not suitable for language modeling
- we define a new, human-readable format that is more than 80% shorter
  - and implement an accompanying transpiler to compile between formats as needed

Adobe

UNIVERSITY OF MANNHEIM

EXAMPLE



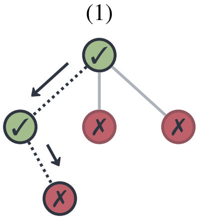

```yaml
s28:
  function: levels
  connections:
    input1:
      node: s27
      id: output
  params:
    parent:
      levelinlow:
        f0:
          function: const_float1
        f1:
          function: const_float1
          params:
            const_float1: 0.5
        f2:
          function: get_float1
          params:
            get_float1: contrast
        f3:
          function: max
          connections:
            b: f0
            a: f2
        f4:
          function: mul
          connections:
            a: f3
            b: f1
        f5:
          function: vector2
          connections:
            componentsin: f4
            componentslast: f4
        f6:
          function: vector2
          connections:
            componentsin: f4
```

IMMEDIATE ERROR HANDLING

Investigating intermediate node states also allows for early error handling.

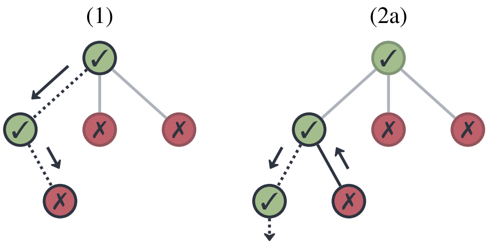
IMMEDIATE ERROR HANDLING

Investigating intermediate node states also allows for early error handling.



IMMEDIATE ERROR HANDLING

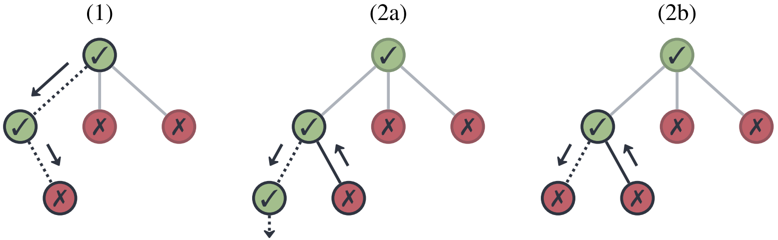
Investigating intermediate node states also allows for early error handling.



- backtrack immediately in case of errors and find valid output subspaces

IMMEDIATE ERROR HANDLING

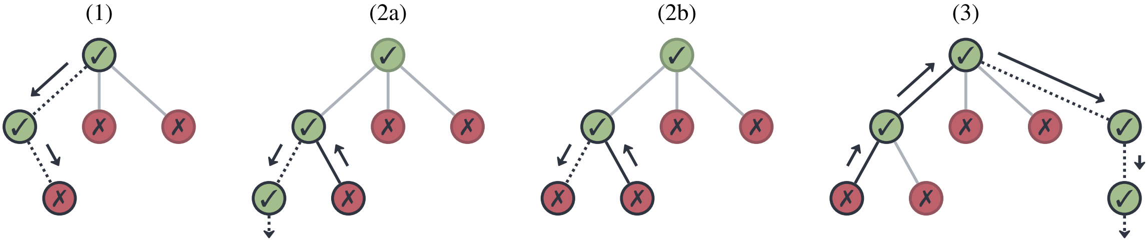
Investigating intermediate node states also allows for early error handling.



- backtrack immediately in case of errors and find valid output subspaces

IMMEDIATE ERROR HANDLING

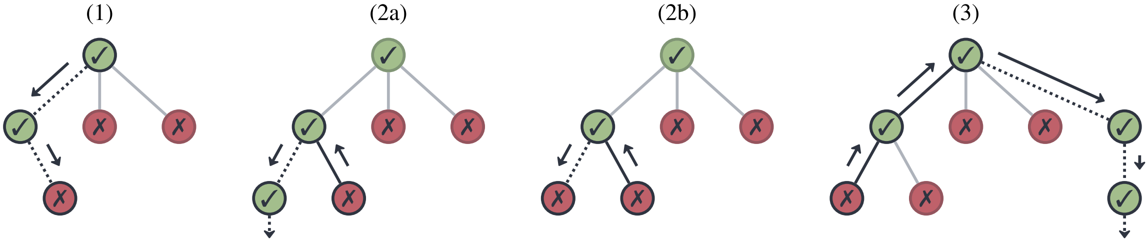
Investigating intermediate node states also allows for early error handling.



- backtrack immediately in case of errors and find valid output subspaces

IMMEDIATE ERROR HANDLING

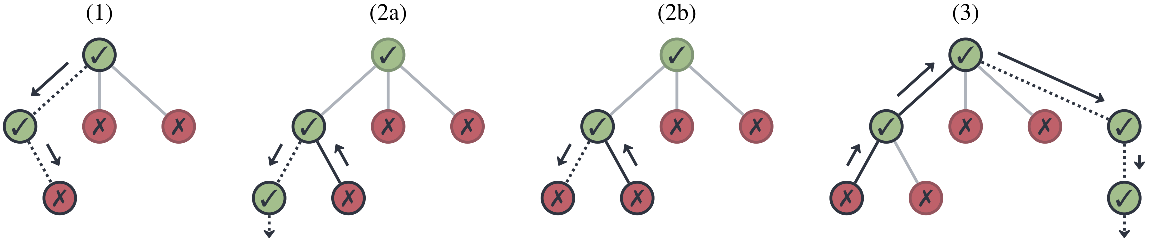
Investigating intermediate node states also allows for early error handling.



- backtrack immediately in case of errors and find valid output subspaces
  - rather than having to sample complete programs before validation can commence (previous work)

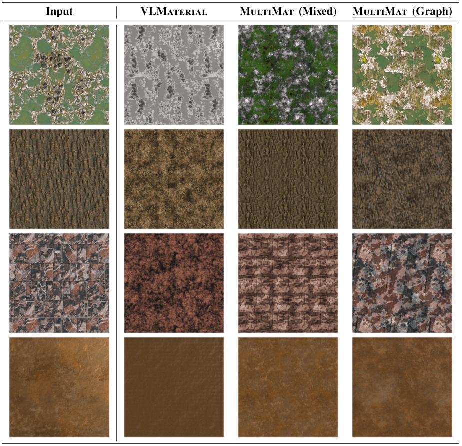
IMMEDIATE ERROR HANDLING

Investigating intermediate node states also allows for early error handling.



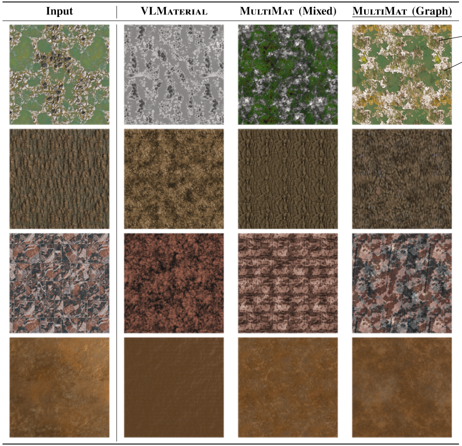
- backtrack immediately in case of errors and find valid output subspaces
  - rather than having to sample complete programs before validation can commence (previous work)
  - transforms inference step into a tree search

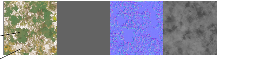
RESULTS



- MultiMat supports conditional and unconditional inference

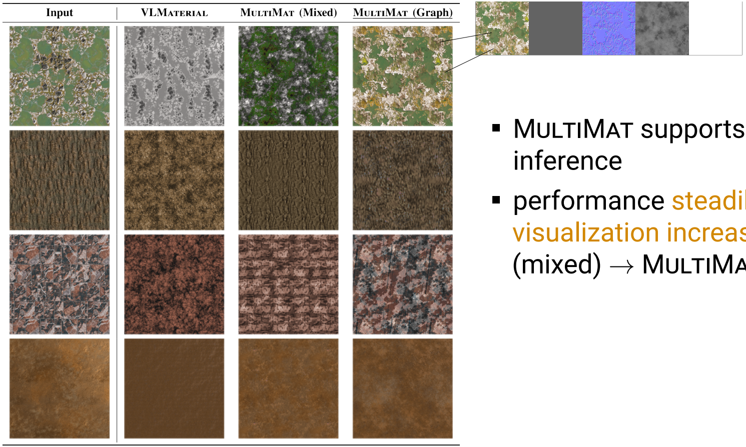
RESULTS





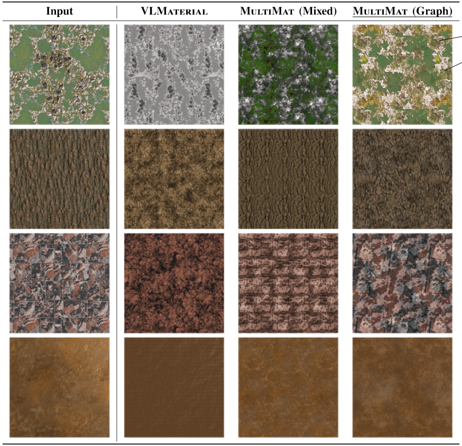
- MultiMat supports conditional and unconditional inference

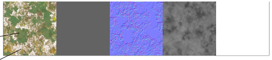
RESULTS



- MultiMat supports conditional and unconditional inference
- performance steadily improves as the degree of visualization increases (VLMaterial → MultiMat (mixed) → MultiMat (graph))

RESULTS

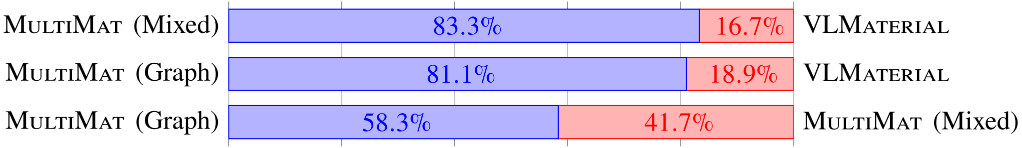




- MultiMat supports conditional and unconditional inference
- performance steadily improves as the degree of visualization increases (VLMaterial → MultiMat (mixed) → MultiMat (graph))
- finding aligns with how humans interact with procedural materials

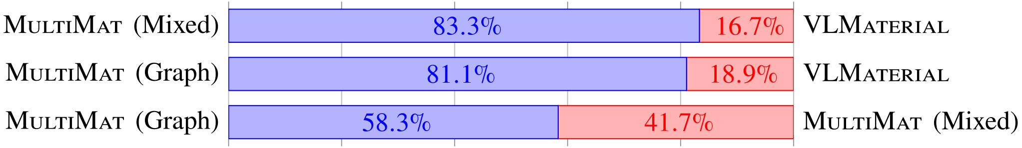
HUMAN EVALUATION

Humans receive triplets of rendered generated materials from each model and identify which output best and least resembles the input image and preferences are visualized as a diverging bar chart.



HUMAN EVALUATION

Humans receive triplets of rendered generated materials from each model and identify which output best and least resembles the input image and preferences are visualized as a diverging bar chart.



According to humans, MultiMat (Graph) is the most preferred model overall, while VLMaterial is consistently the least preferred.

INTERESTED? THERE'S MORE!

In our paper, we additionally conduct an automatic evaluation including for unconditional models, discuss our training data collection, show that MultiMat is less susceptible to errors than its baseline, and showcase many more examples and also failure cases.