

# Text-to-Scene Generation

CMPT-413

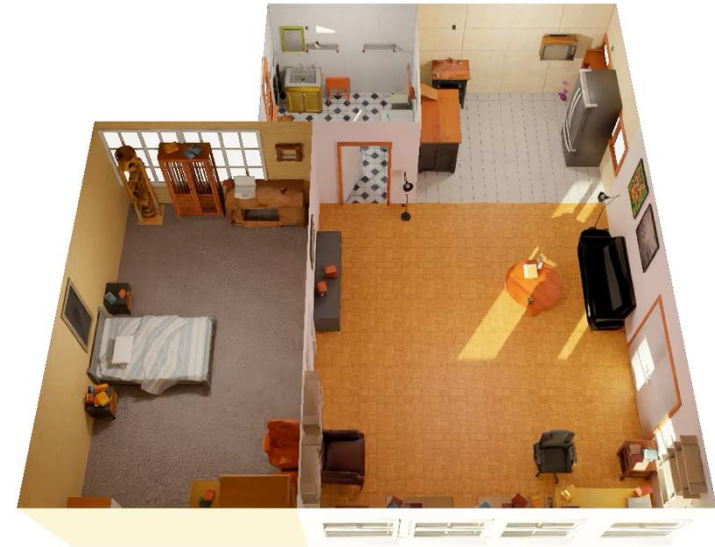
March 26, 2025

# Generate 3D scenes from text descriptions

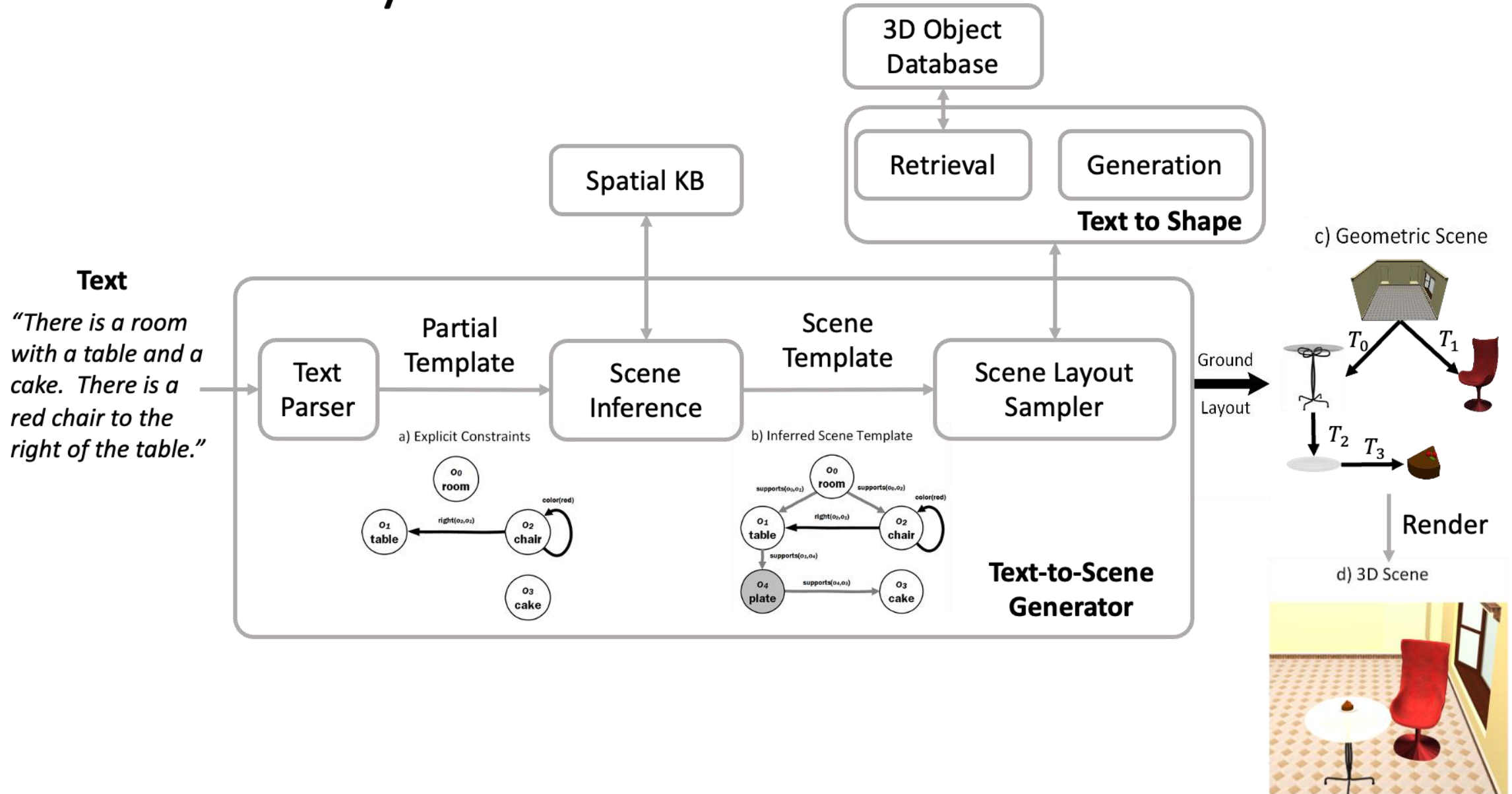
Three professors' office connected to a long hallway. The professor in office 1 is a fan of Star Wars.



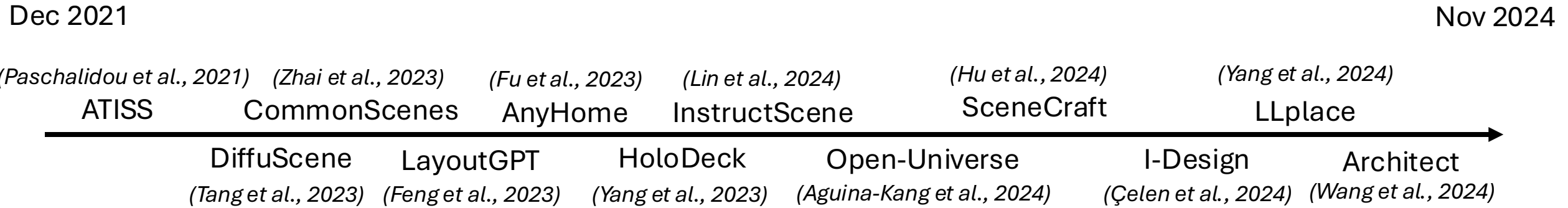
A 1b1b apartment of a researcher who has a cat.



# Text to Scene System



# Timeline



# 3D-FRONT

- 6,813 houses
- 14,629 rooms

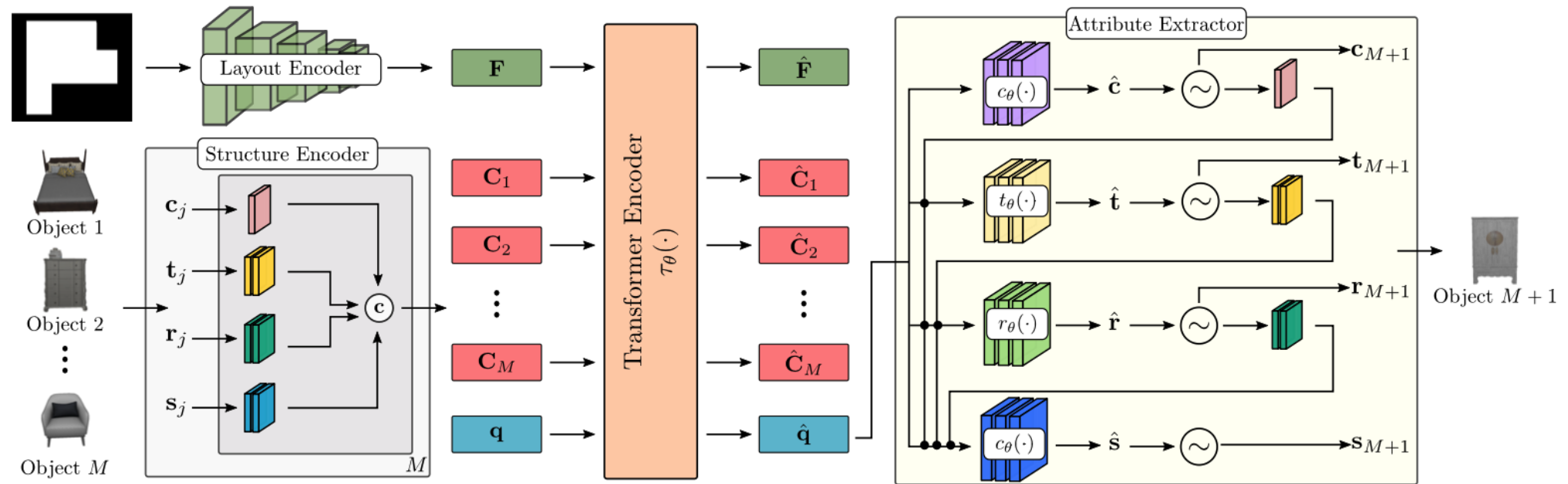
Room type	Number	Object categories
bedrooms	4000	21
living rooms	800	24
dining rooms	900	24

*Approximate counts of each room type and object category in 3D-FRONT, as used by DiffuScene after post-processing [Tang et al, 2023]*



# Objects can be generated autoregressively in a scene

- E.g., ATISS
- Conditioned on room type and floorplan



# ATISS: Examples

Bedroom

Living room

Dining room

From Paper

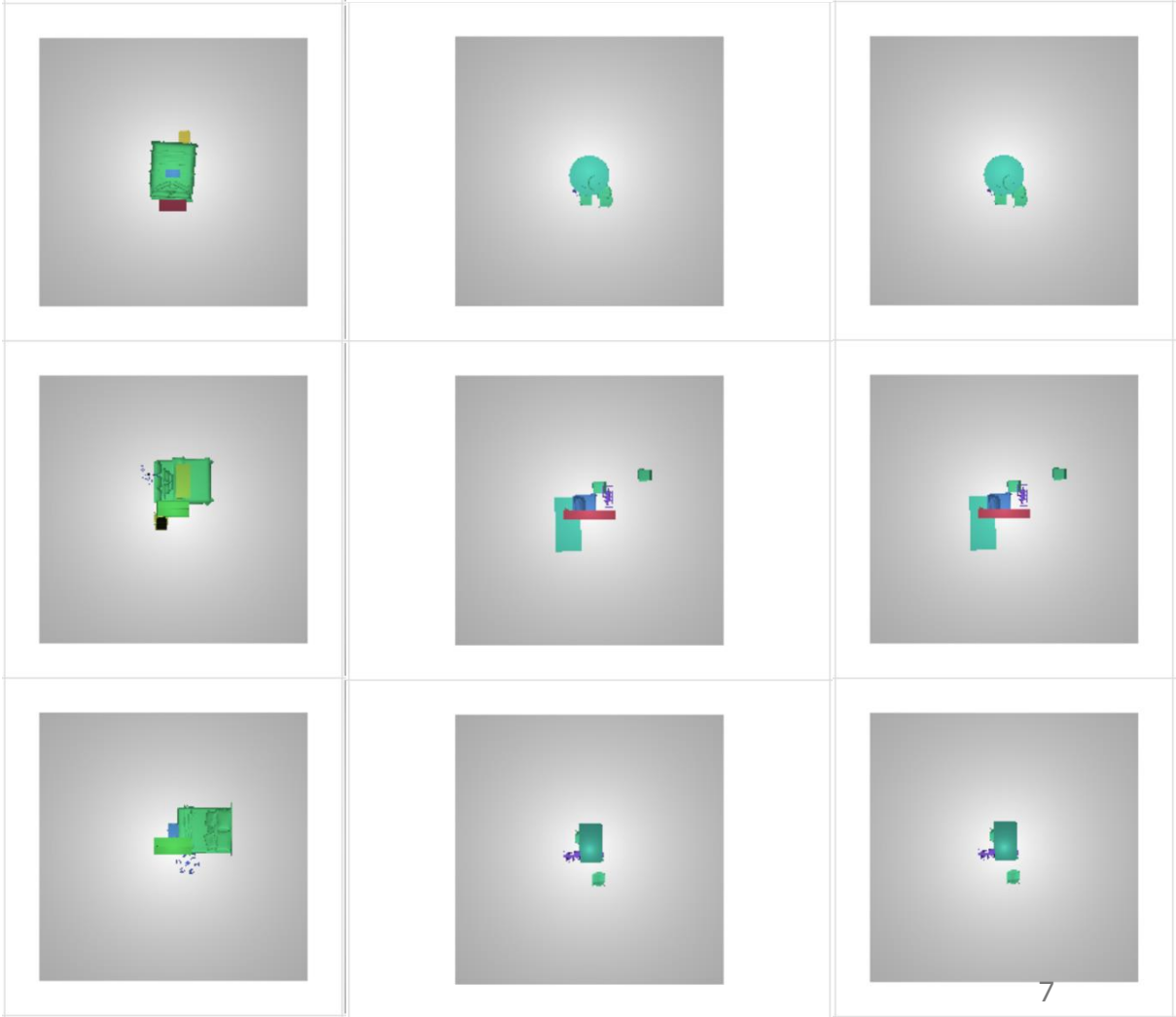


Bedroom

Living room

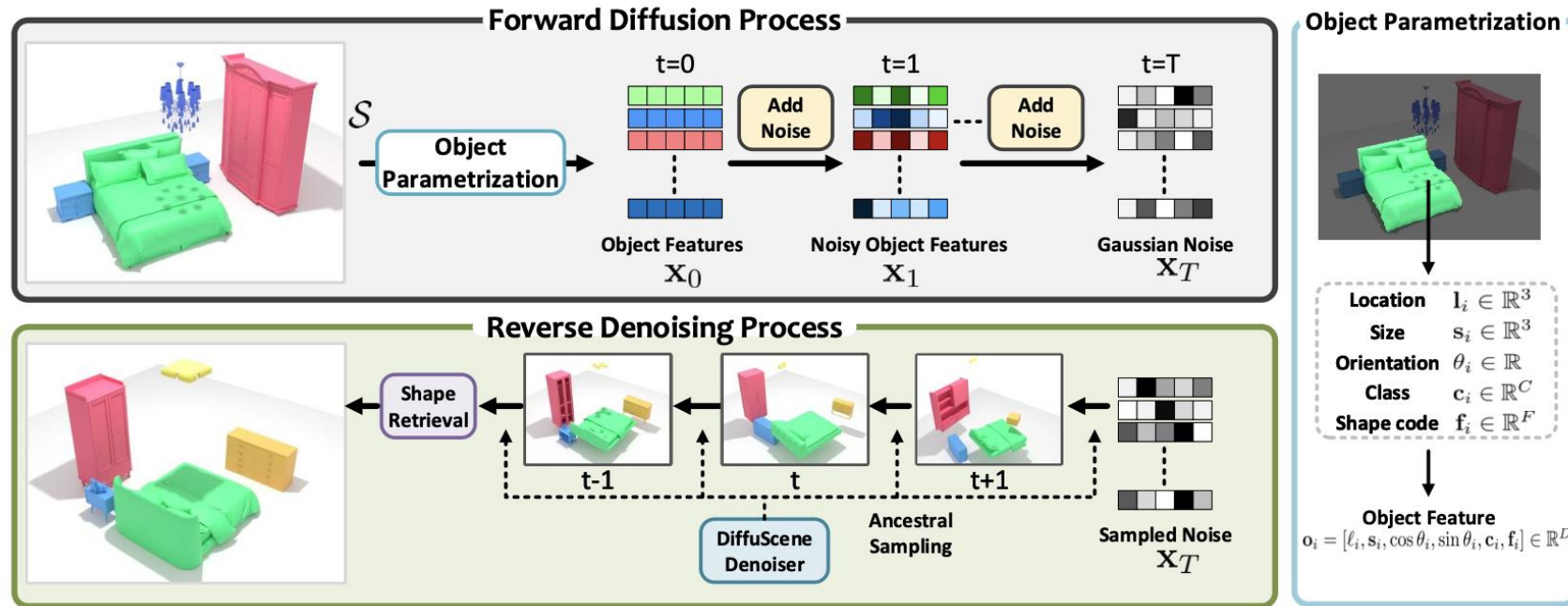
Dining room

Generated



# Diffusion can be applied directly to object features

- E.g., DiffuScene
- Conditioned on room type and floorplan; text optional



[DiffuScene: Scene Graph Denoising Diffusion Probabilistic Model for Generative Indoor Scene Synthesis](#)

[Tang et al, 2023]



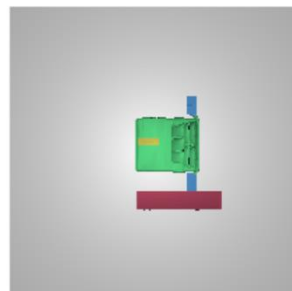
# DiffuScene: Examples

From Paper

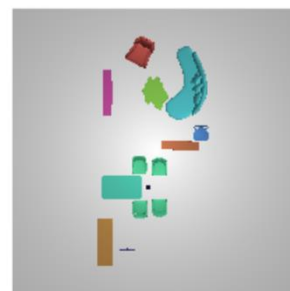


Generated

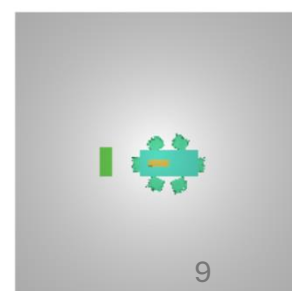
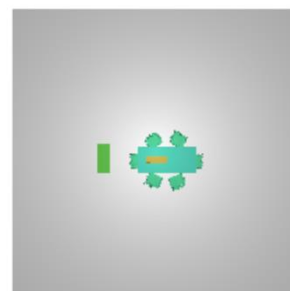
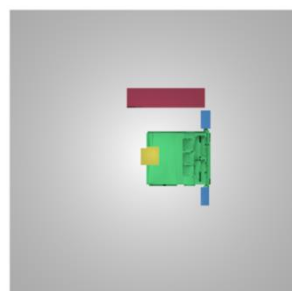
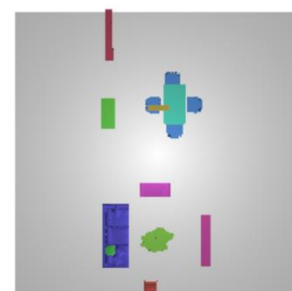
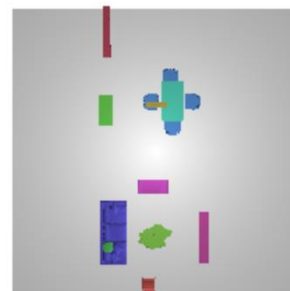
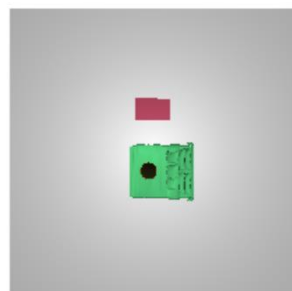
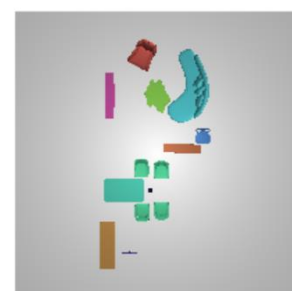
Bedroom



Living room

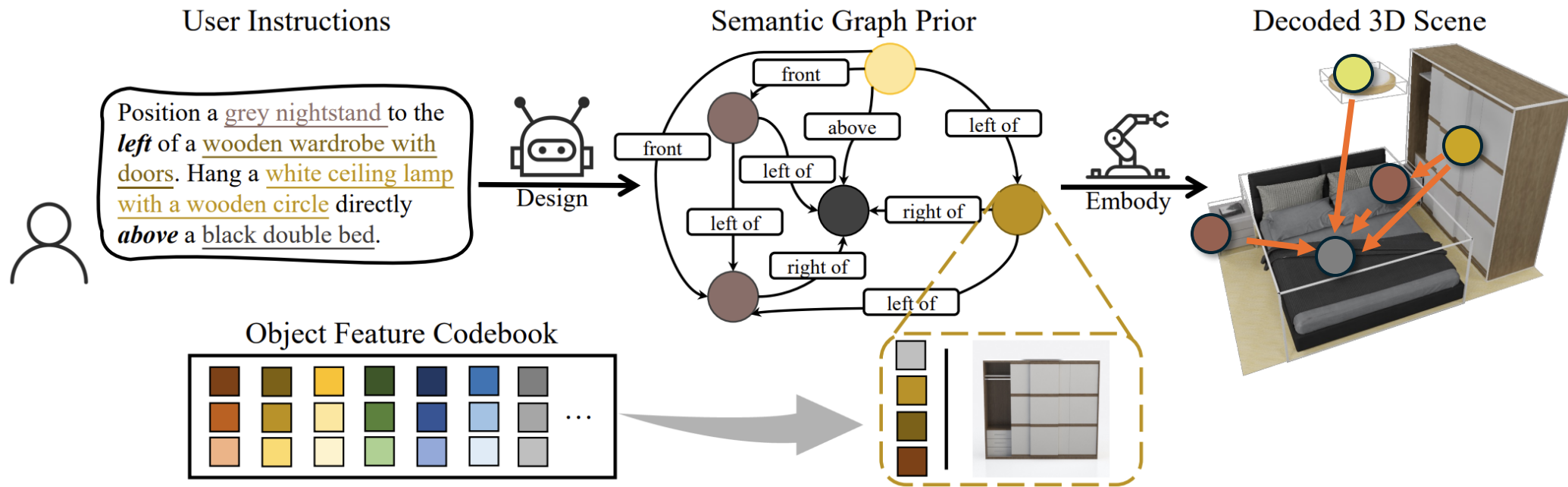


Dining room

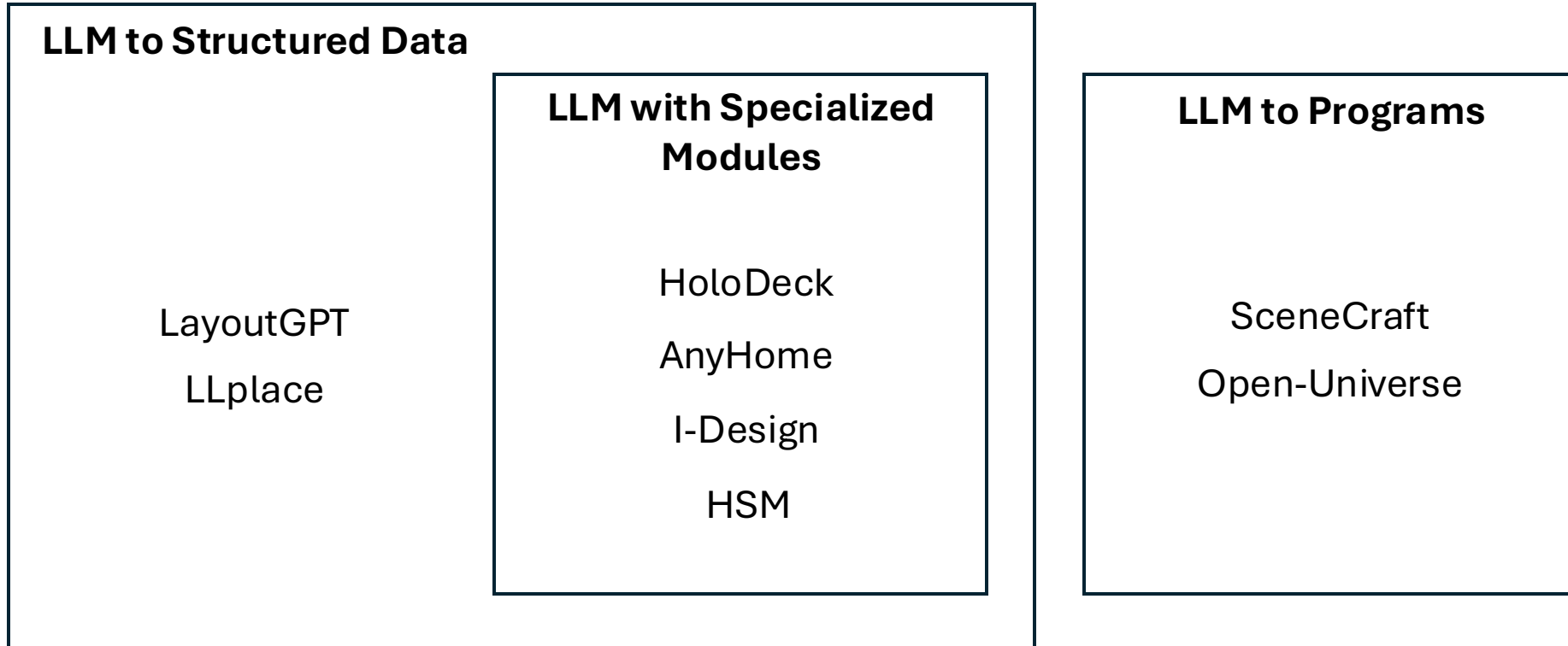


# Diffusion can be used to generate scene graphs

- E.g., InstructScene, CommonScenes



# LLMs empower *open-vocabulary* scene generation



# LLM to Structured Data

## Method:

- Use LLM to generate JSON or CSS to represent a 3D scene

## Advantages:

- Leverages LLM's broad knowledge for diverse outputs with a user-friendly natural language interface

## Disadvantages:

- Limited visual understanding may lead to physically implausible or inconsistent outputs
- Rather limited scene types (bedroom, living room, dining room)
- [LLplace] Does not support specifying relationships

Bedroom



Here is a bedroom with a brown wardrobe, two brown nightstands with a lid, a wooden bowl with a metal handle and a grey double bed with cover and pillows.

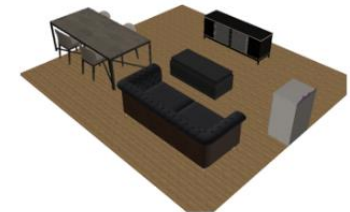


I have a bedroom with a dressing table with a drawer and a shelf, two black nightstands with a vase and books, a black and gray cabinet with two drawers, a green blanket stool, a black and white double bed, a black pendant lamp with metal rods and a black wardrobe with doors and a drawer.

Livingroom



This living room have a wooden coffee table and two wooden corner side tables with a pinecone on top, a metal cage pendant lamp, a wooden cabinet and a wooden tv stand with a drawer. And this room have a gray and yellow multi-seat sofa and a lounge chair with a pillow.

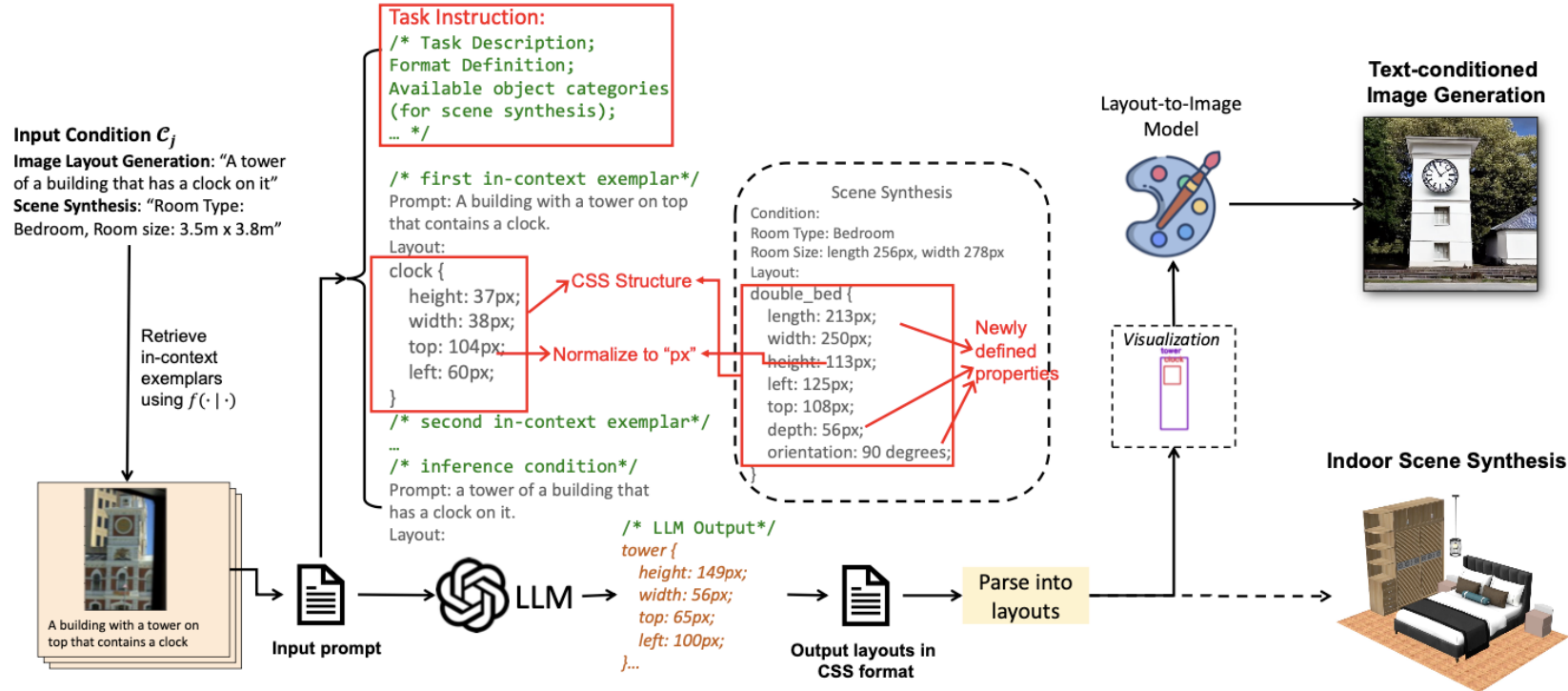


In this living room, there are four black dining chairs with a frame, a black and gold tv stand with doors, a black and gold coffee table, a black pendant lamp with a shade, a black metal dining table, a purple and pink flower children cabinet and a loveseat chesterfield sofa.

Figure from LLplace

# Ex: LayoutGPT

- Tasks: image generation, scene generation
- Using LLM for scene layout planning, with CSS as the layout syntax



[LayoutGPT: Compositional Visual Planning and Generation with Large Language Models](#)

[Feng et al. 2023]



# LLM with Specialized Modules

Method:

- Use multiple modules of specialized LLM to generate 3D scene

Advantages:

- Combines LLM's knowledge with domain-specific constraints and rules
- Can generate whole floor with multiple rooms

Disadvantages:

- Struggle with complex layout



Figure 1. Example outputs of HOLODECK—a large language model powered system, which can generate diverse types of environments (arcade, spa, museum), customize for styles (Victorian-style), and understand fine-grained requirements (“has a cat”, “fan of Star Wars”).

Figure from Holodeck

# LLM to Programs

## Method:

- Use LLM to generate executable programs to generate a 3D scene

## Advantages:

- Highly flexible and expressive
- Able to generate more diverse and complex scenes
- Isolates parameters of layout for optimization

## Disadvantages:

- Dependency on LLM's ability to generate correct, executable code
- Accurately representing and satisfying complex spatial relationships between objects can be challenging when working through code generation

**Query:** Three trees in a row besides a neighborhood

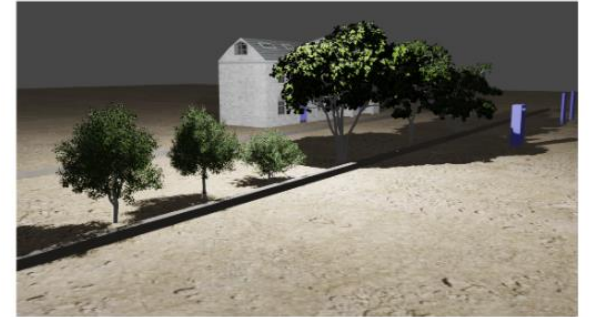
```
assets = [tree1, tree2, tree3, neighborhood]

# Alignment along x axis
x_align_score = alignment_score(assets, 'x')

# Perpendicularity between trees and neighborhood
perpendicularity_score(tree1, neighborhood)

# Parallelism between trees
tree_parallelism = parallelism_score([tree1, tree2, tree3])

constraints = [
    x_align_score, perpendicularity_score, tree_parallelism
]
```



**Query:** A new solar system with planets orbiting around a star

```
star_layout = import_obj("star.obj")[0]
planet_layouts = [import_obj(f"planet{i}.obj")[0] for i in range(1,4)]

assets = [star_layout] + planet_layouts

# Randomize planet positions
for planet in planet_layouts:
    r = random.uniform(10, 30)
    theta = random.uniform(0, 360) # angle in degrees
    x = r * math.cos(math.radians(theta))
    y = r * math.sin(math.radians(theta))
    planet.location = (x, y, 0)

# Orient planets towards star
for planet in planet_layouts:
    angle = vector_to_angle(planet.location, star_layout.location)
    planet.orientation = angle_to_euler(angle)

# Calculate rotational uniformity
uniformity = rotation_uniformity_score(planet_layouts, star_layout.location)
```



**Query:** Raccoon is lying on a bed, listening to music,

```
# Lie racoon down
raccoon.orientation = (90, 0, 0)

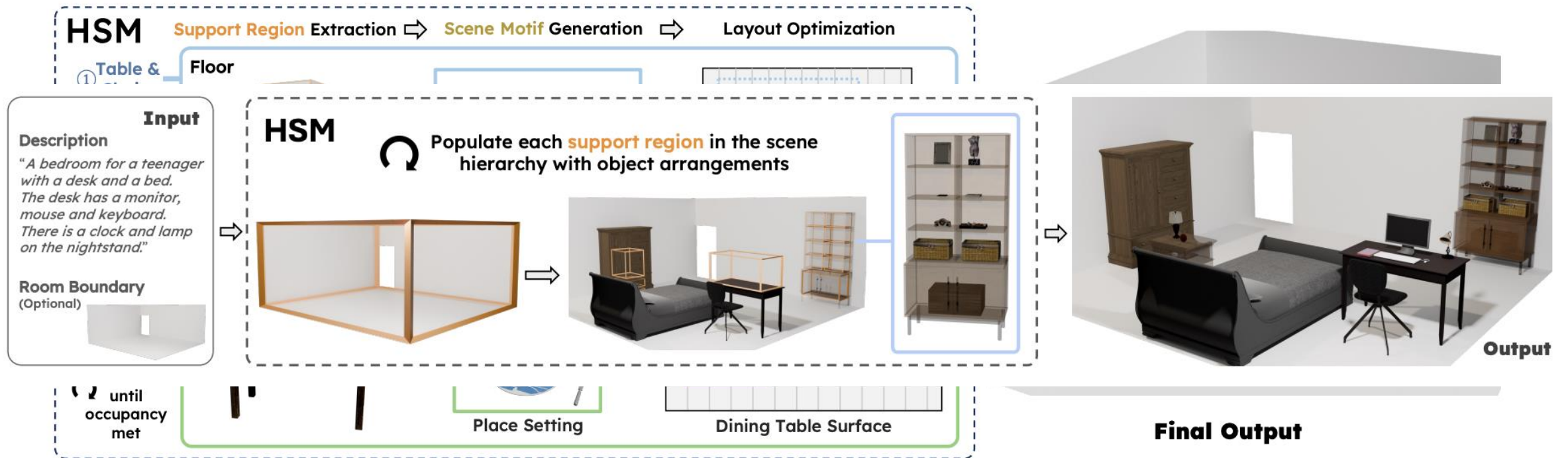
# Initial headphone position
headphones.position = (raccoon.position[0], raccoon.position[1], raccoon.position[2] + 1)

# Iteratively adjust headphones
proximity_threshold = 0.9
align_threshold = 0.9
for i in range(10):
    # Check proximity and alignment
    proximity_score = proximity_score(headphones, raccoon, 0, 0.5)
    align_score = direction_score(headphones, raccoon)
    # If constraints not met, adjust headphones
    if proximity_score < proximity_threshold or align_score < align_threshold:
        # Update position
        headphones.position = {
            raccoon.position[0] + random.uniform(-0.1, 0.1),
            raccoon.position[1] + random.uniform(-0.1, 0.1),
            raccoon.position[2] + random.uniform(0.8, 1.2)
        }
```



Figure from SceneCraft

# HSM leverages recurring “motifs” to arrange objects at all scales

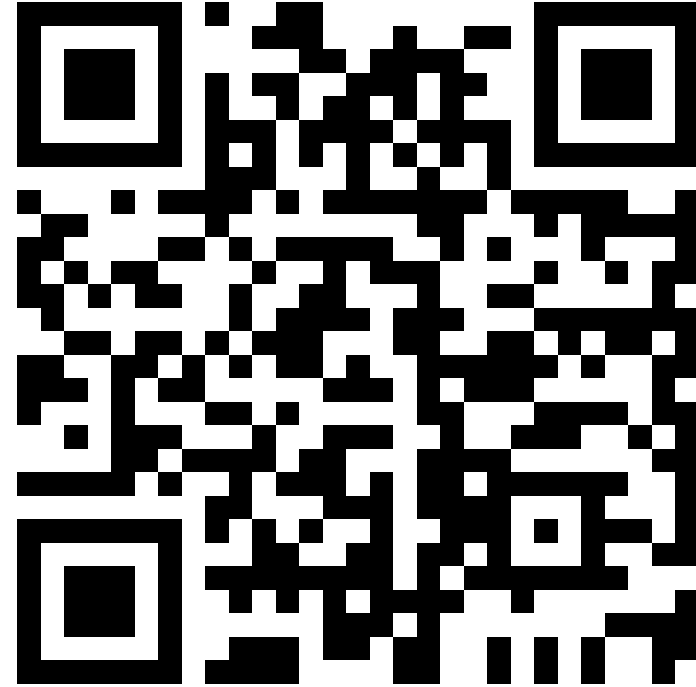




# Learn More!



SceneMotifCoder



HSM