KARA

Newsletter – Issues 003 & 004

Welcome to the KARA newsletter. These newsletters will keep you informed about the research and development that the KARA project team are undertaking.

003. Agent Swarm - Expanding the scope of chat bots

In this edition, we introduce you to the concept of an 'Agent Swarm' - Individual language models (focused on specific expertise) co-ordinating with each other as a team!

004. Enhancing blockouts with 2.5D Modelling

In this edition, we explore the powerful combination of Generative AI and procedural workflows. We'll introduce the Machine Learning Operators (MLOPs) plugin for Houdini: Focusing on how we've utilised them in a procedural network to craft debris models by converting 2D images into 2.5D meshes.

IMPORTANT: Inclusion of tools in this newsletter does not imply their clearance for use.

Project Goal
Examine the potential of Gen AI in game development through applied R&D.

What is KARA?

In 2023, Project AVA took on the challenge of utilising Generative AI tools to assist in the design and development of a videogame from scratch. This applied R&D approach gave us the best insights into the true potential of AI for video game development.

The AVA team went to GDC in 2024 to report on their findings (watch the video).

KARA is the continuation of this effort. Using Electric Square's *Detonation Racing* as a case study, the team's plan is to remaster the game using pipelines infused with Generative AI tools.

The goal of KARA is to further enhance our expertise in Generative AI for game development. This includes a focus on how GAI tools can boost 3D art pipelines.

Inclusion of tools in this newsletter does not imply their clearance for use

ISSUE: 003

TOOL SPOTLIGHT

Agent Swarm

Expanding the scope of chat bots.

Introduction

Agent Swarms are a way to utilise LLMs (Chat-bots) such as ChatGPT to break down large, complex tasks into smaller, narrower tasks and complete them. They are one of the most promising frontiers for using AI for code generation and other complex tasks. It is essentially using lots of LLM instances, each with a very narrow, specific task, to all work collaboratively in a swarm.

TOOL SPOTLIGHT

Agent Swarm - Expanding the scope of chat bots

Vision

How Agent Swarms fit into KARA?

USER <-> LLM

Existing GAI coding tools are fairly limited in scope as we've already seen, and due to the off-the-shelf nature; any R&D findings are limited.

USER <-> AGENT SWARM

Creating our own Agent Swarm implementation has a lot of potential both in terms of direct wins for Project KARA and useful tools which could be used across the entire group, giving us a meaningful competitive edge. They are one of the best hopes for significant AI code-assistance and also offer the possibility of a wide range of uses in other areas of game development and integration into other pipelines across disciplines.

TOOL SPOTLIGHT

Agent Swarm - Expanding the scope of chat bots

Investigation

We decided to investigate Agent Swarms to see if they showed any potential in terms of tackling the problems we had seen with existing chat-bots when tackling larger, more complex tasks. We began by taking an open-source framework available on the internet to kickstart basic capability testing using OpenAI as the back-end. This testing covered 3 basic stages, an initial security audit, investigation into the structure of the system and how it's built, and finally some simple capability tests to see if it can work as advertised.

The security audit was not exhaustive; however, it offered us enough confidence that the framework was built with honest intentions and gave us enough of a basis to conduct testing using an isolated laptop without any confidential content on it, on a secure network. The investigation into the structure gave us a solid view of the way it had been built, the issues they had faced, and revealed many areas where we could see the opportunity to make improvements if we make our own systems.

Finally, the experiments we conducted were simple in nature and designed primarily as a proof of concept for the claimed functionality of the Agency-Swarm framework. Allowing us to see each part of the system functioning and show that there was potential worth investigating further.

The Tasks Were:

Hello User

Write a console application to ask the user their name and then say hello to them.

Tic Tac Toe

Creating a simple console application to play Tic Tac Toe against the user.

User age calculator

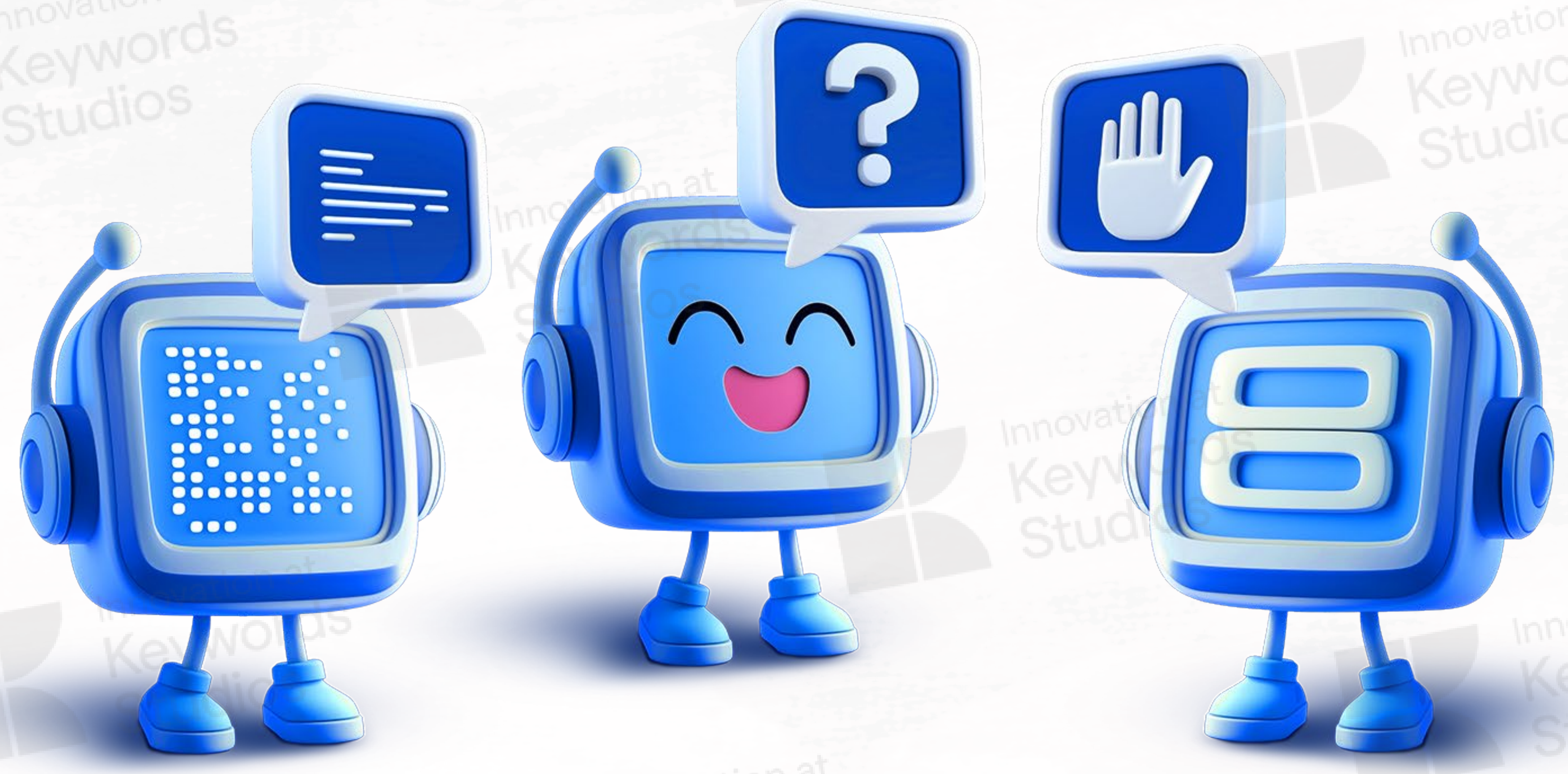
Ask the user for their birthday and then tell them their age.

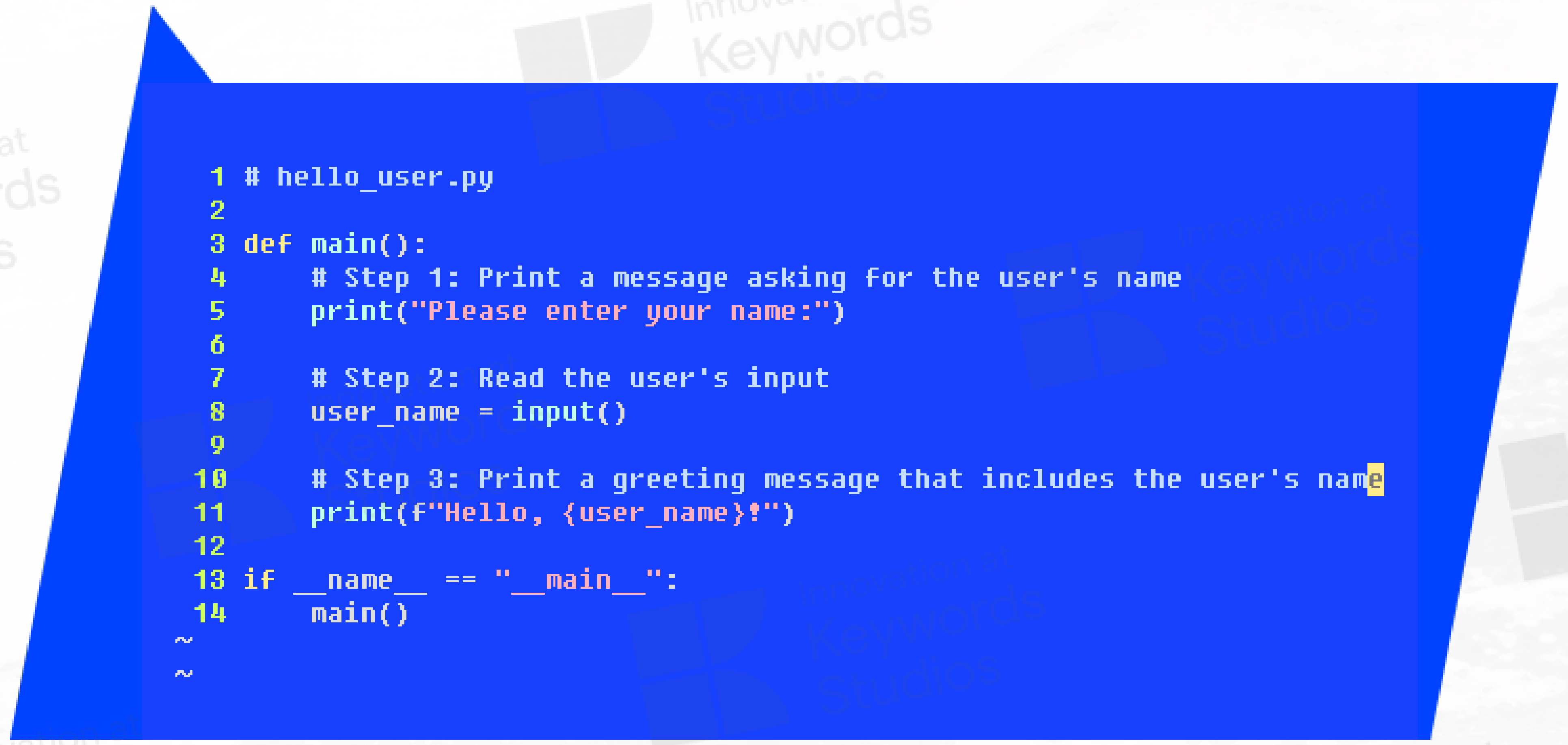
TOOL SPOTLIGHT

Agent Swarm - Expanding the scope of chat bots

Results: Hello User

Write a console application to ask the user their name and then say hello to them.





```
# hello_user.py

def main():
    # Step 1: Print a message asking for the user's name
    print("Please enter your name:")

    # Step 2: Read the user's input
    user_name = input()

    # Step 3: Print a greeting message that includes the user's name
    print(f"Hello, {user_name}!")

if __name__ == "__main__":
    main()
~
~
```

TASK COMPLETED

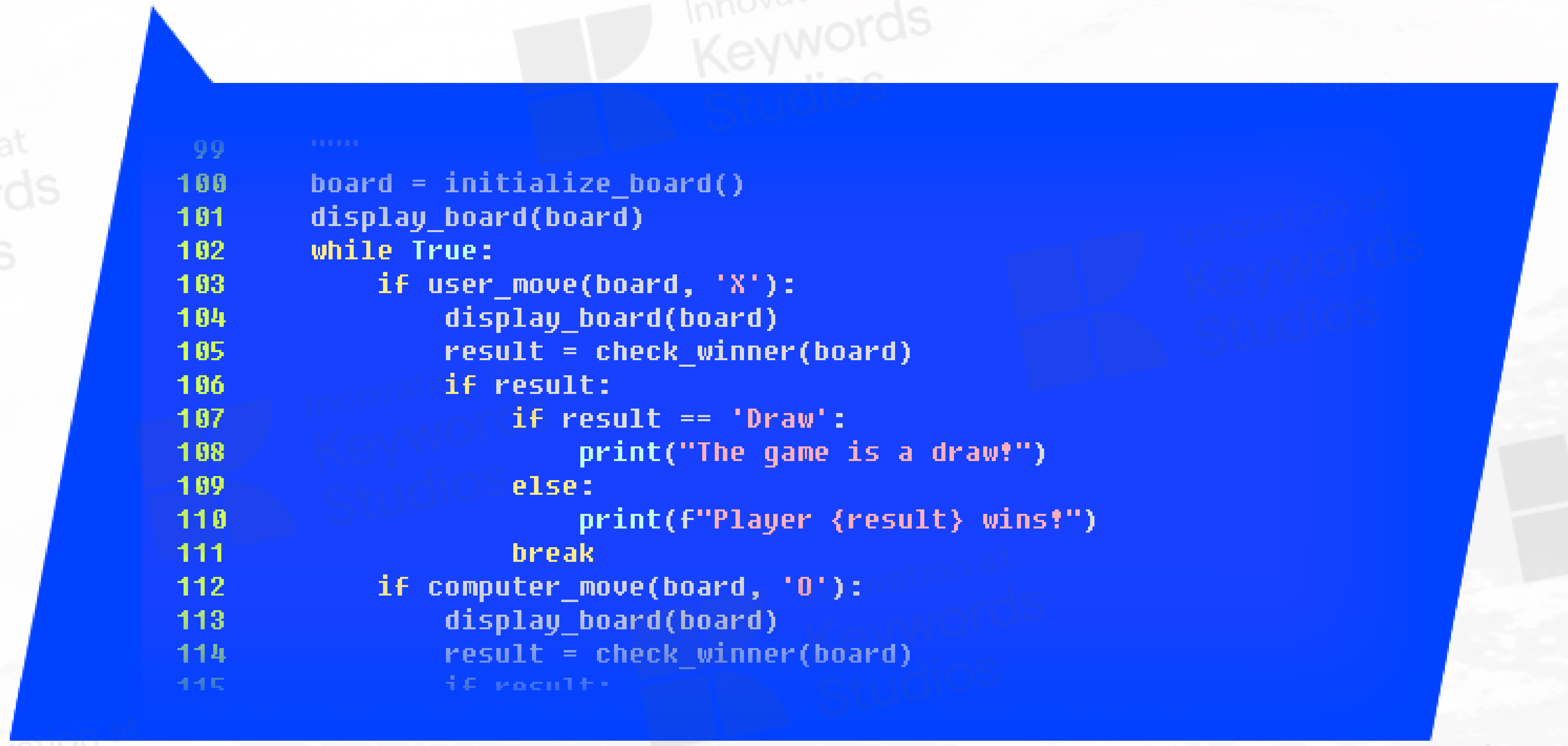
Inclusion of tools in this newsletter does not imply their clearance for use

TOOL SPOTLIGHT

Agent Swarm - Expanding the scope of chat bots

Results: Tic Tac Toe

Create a simple console application to play Tic Tac Toe against the user.



```
    """
    board = initialize_board()
    display_board(board)
    while True:
        if user_move(board, 'X'):
            display_board(board)
            result = check_winner(board)
            if result:
                if result == 'Draw':
                    print("The game is a draw!")
                else:
                    print(f"Player {result} wins!")
                break
        if computer_move(board, 'O'):
            display_board(board)
            result = check_winner(board)
            if result:
```

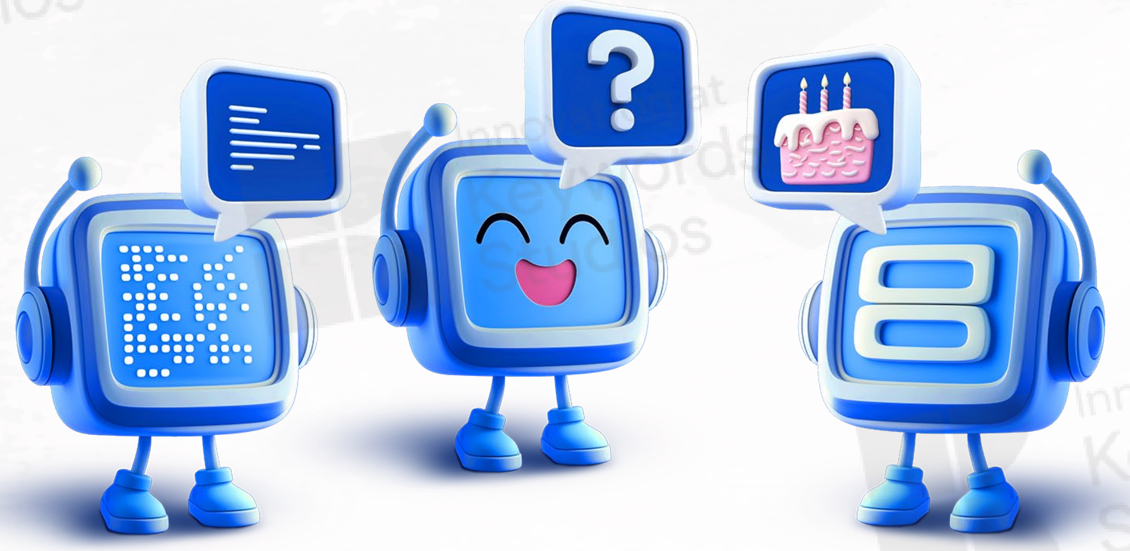
TASK COMPLETED

TOOL SPOTLIGHT

Agent Swarm - Expanding the scope of chat bots

Results: User age calculator

Ask the user for their birthday and then tell them their age.

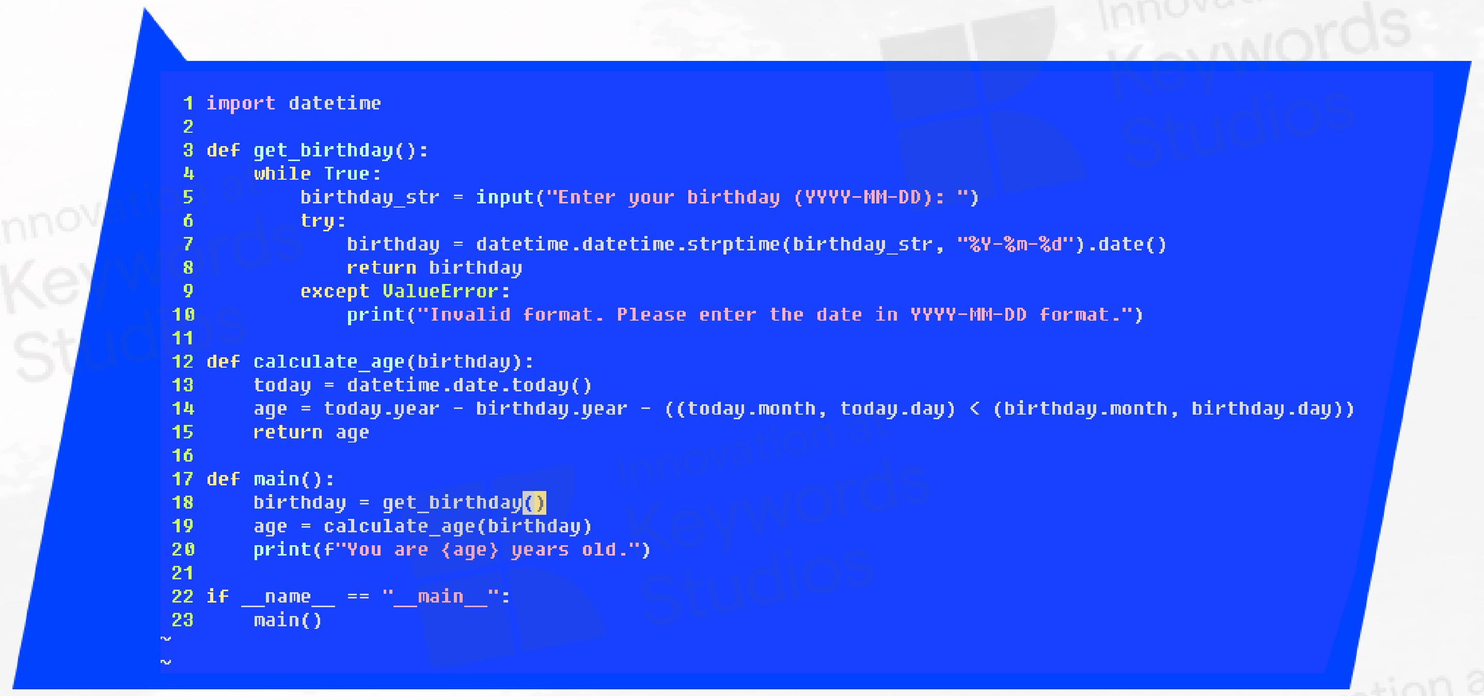


The initial result of the experiment only asked the user for their birth year and did not take into account the day and month:

TASK COMPLETED

Modification required

We used a new instance of the development agency to test introducing an existing application and asking it to understand and modify it to take account of the day and month.



```
import datetime

def get_birthday():
    while True:
        birthday_str = input("Enter your birthday (YYYY-MM-DD): ")
        try:
            birthday = datetime.datetime.strptime(birthday_str, "%Y-%m-%d").date()
            return birthday
        except ValueError:
            print("Invalid format. Please enter the date in YYYY-MM-DD format.")

def calculate_age(birthday):
    today = datetime.date.today()
    age = today.year - birthday.year - ((today.month, today.day) < (birthday.month, birthday.day))
    return age

def main():
    birthday = get_birthday()
    age = calculate_age(birthday)
    print(f"You are {age} years old.")

if __name__ == "__main__":
    main()
~
~
```

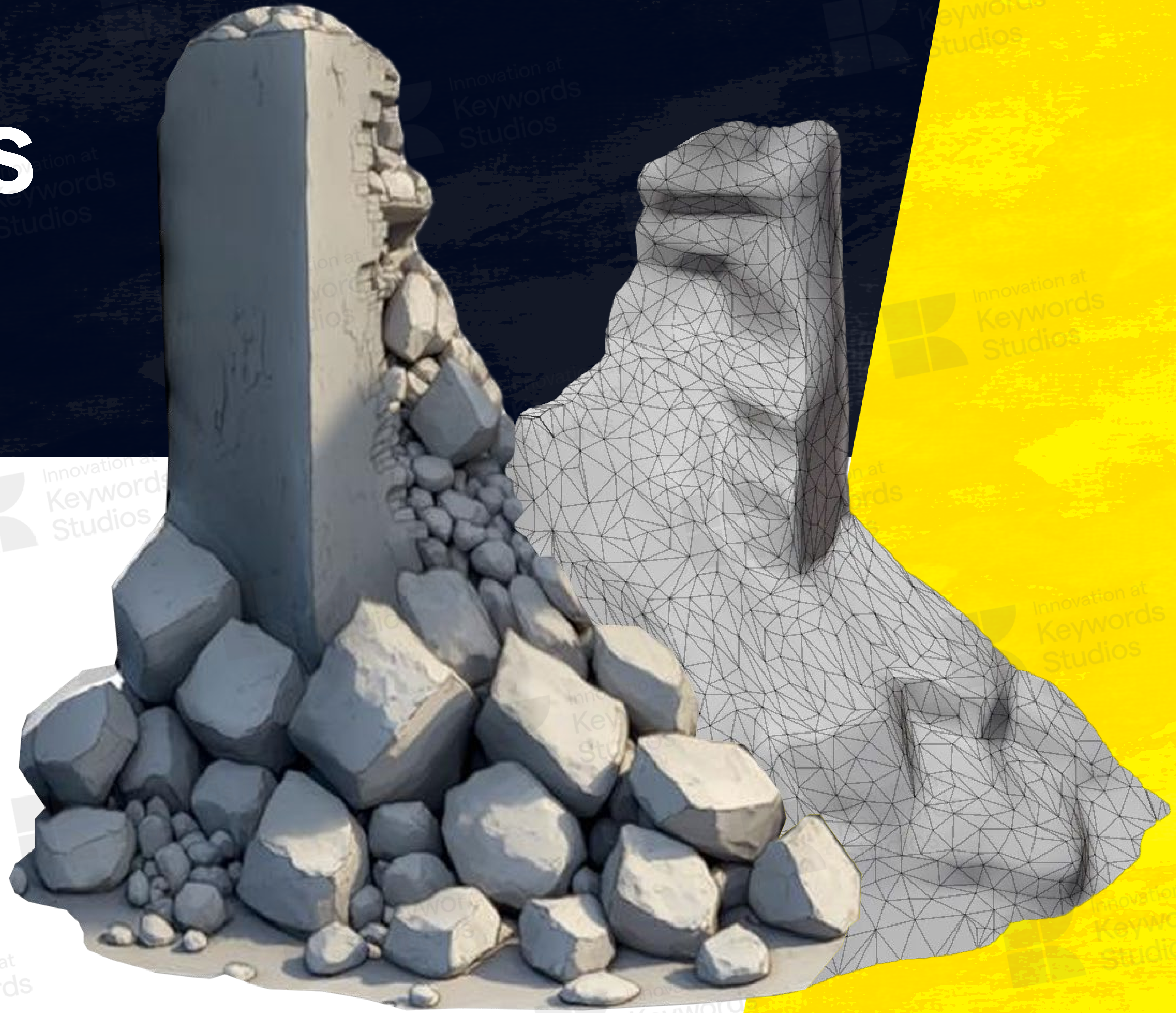
TASK COMPLETED

ISSUE: 004

PIPELINE SPOTLIGHT

Enhancing blockouts with 2.5D Modelling

The concept of "2.5D" denotes a geometry which derives from the extension of a 2D image in a third dimension, designed to function optimally only from a specific, limited perspective.



Introduction

GAI, though innovative, can lack control and produce random outcomes. Procedural workflows, while efficient and structured, can sometimes be rigid and produce repetitive patterns.

When merged, GAI's creativity and procedural workflows can complement each other, overcoming each approach's limitations.

In our research, we explored this synergy, starting with a 2.5D generation technique to improve old Detonation Racing debris models.

Inclusion of tools in this newsletter does not imply their clearance for use

PIPELINE SPOTLIGHT

Enhancing blockouts with 2.5D Modelling

Considerations

The blockout phase is crucial for visualizing the final design of a scene early in development.
It allows the whole team to quickly assess the composition, scale, mood and gameplay flow, allowing fast iteration and ensuring everyone is aligned before committing to detailed models and textures.

STANDARD PIPELINE

Standard pipeline

1
Reference Gathering

Artists start by gathering reference materials, including concept art, basic sketches, and any visual guides relevant to the scene layout. These references help determine the scale, form, and approximate position of key elements. The focus at this stage is on identifying the essential shapes and spatial relationships, ensuring that the layout will provide a solid foundation for further development.

2
Base Model Creation

Using basic geometric shapes such as cubes, cylinders, and spheres, artists construct rough versions of each object in the scene. These placeholder models capture the essential proportions and positions without diving into fine details.
The objective is to quickly establish a readable layout that will allow for easy adjustments and ensure that the overall composition feels balanced and functional within the intended environment.

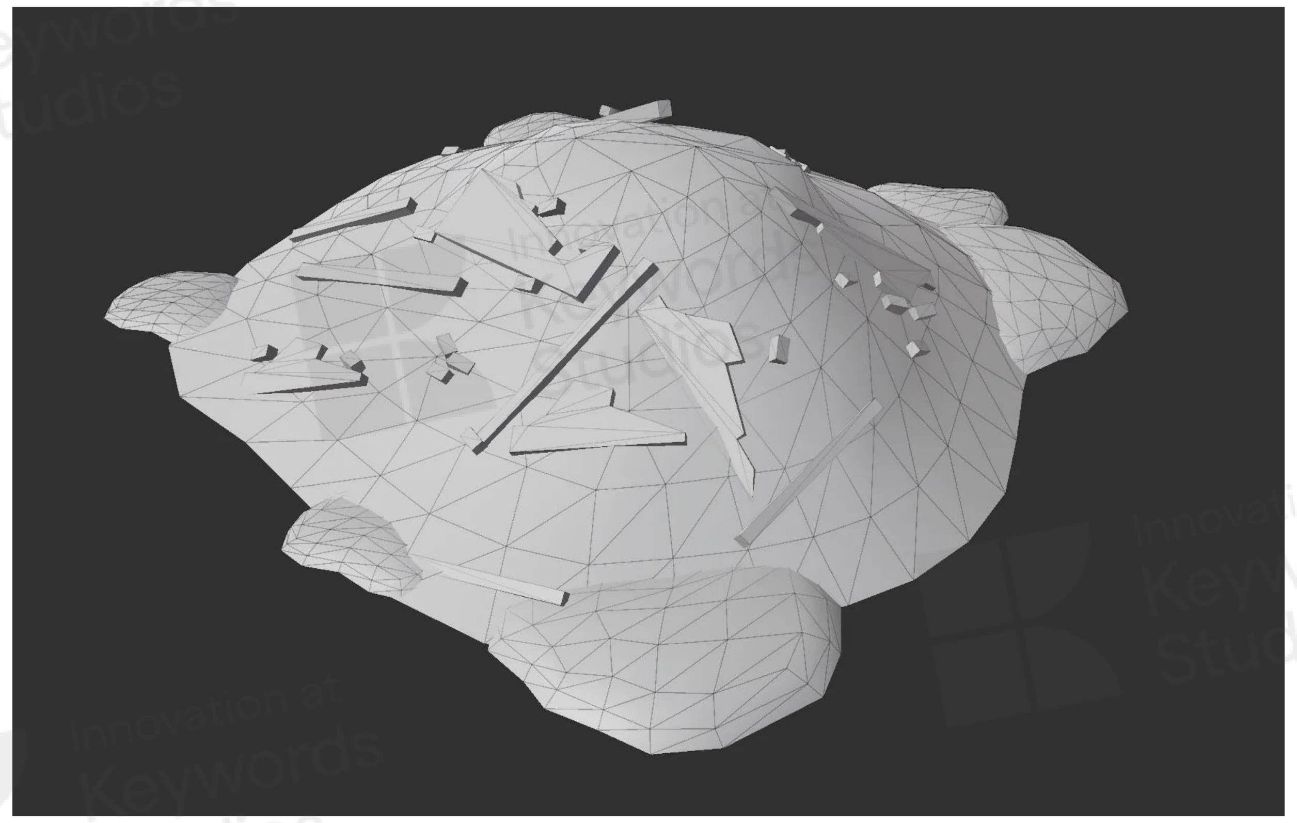
3
Adding Texture and Colour

Once the base models are in place, artists apply simple colours and textures to the blockout assets to help distinguish different elements and materials. This step is not focused on realism but rather on providing visual clarity. This ensures that each placeholder object can be easily identified in the game world. The use of basic colours or patterns aids in visual organisation and helps other team members understand the layout.

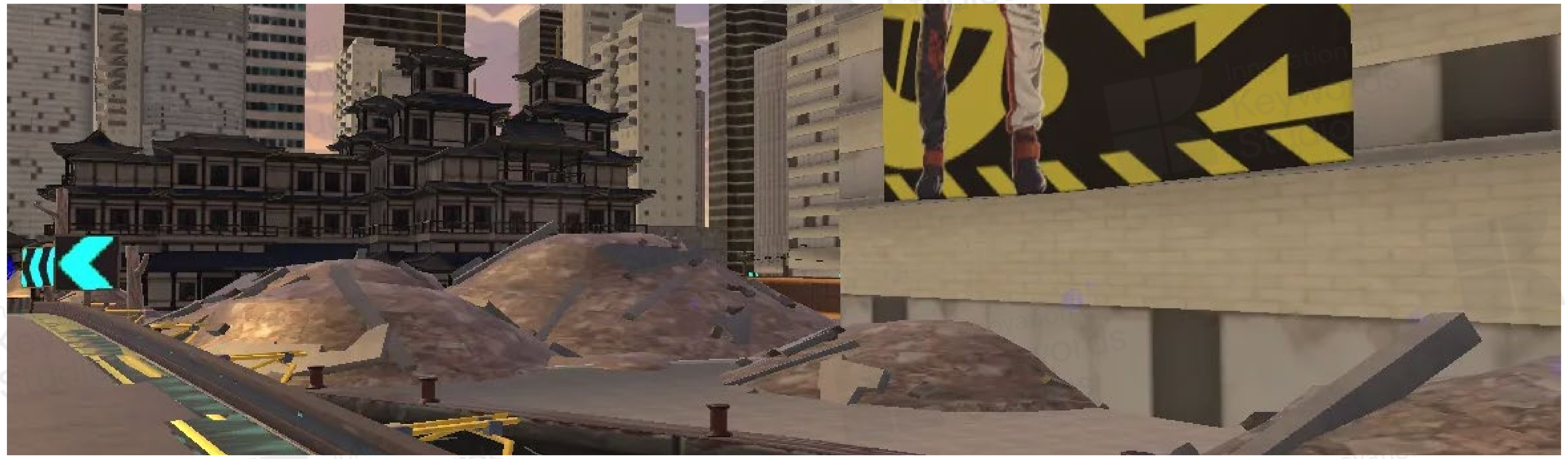
4
Engine Integration

The completed blockout models are imported into the game engine for initial testing. This allows artists and designers to evaluate the layout and gameplay flow within the actual game environment. Adjustments are made to refine the spatial relationships and ensure that the placeholders support the intended interactions and visual hierarchy. This integration phase ensures that the blockout provides a reliable framework for future, detailed modeling stages.

Examples of output from a typical pipeline:



Original assets used in Detonation Racing:



Inclusion of tools in this newsletter does not imply their clearance for use

PIPELINE SPOTLIGHT

Enhancing blockouts with 2.5D Modelling

Tools used:

Houdini | MLOPs

Houdini is a leading tool in the 3D and VFX industry, known for its powerful procedural workflows. **MLOPs** (Machine Learning Operators) enhance this by enabling AI-driven data processing for even greater creative flexibility.

AI INFUSED PIPELINE

AI Infused Pipeline

1

Generate or choose input image

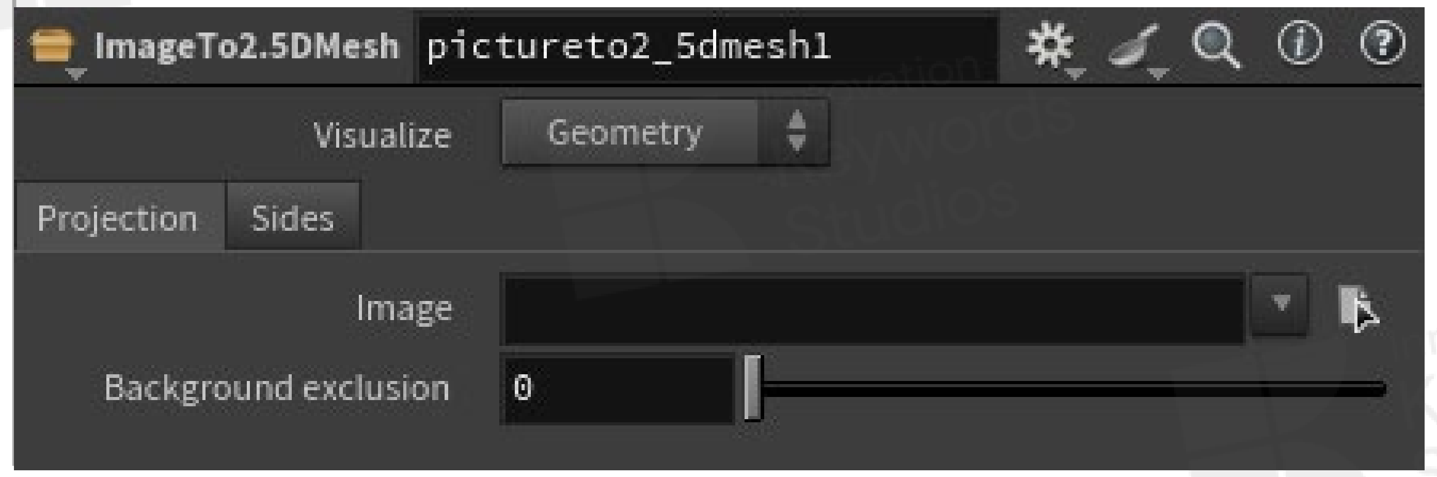
Using machine learning algorithms, it is possible to generate an input image right in Houdini through MLOPs.

Alternatively, you can manually select an image. Make sure your chosen image provides sufficient depth information for depth map generation.

2

2.5D mesh generation

Input the image path into the tool's HDA (Houdini Digital Asset) within Houdini or right within Unity using the Houdini Engine plugin.



ImageTo2.5DMesh: pictureto2_5dmesh1

Visualize: Geometry

Projection | Sides

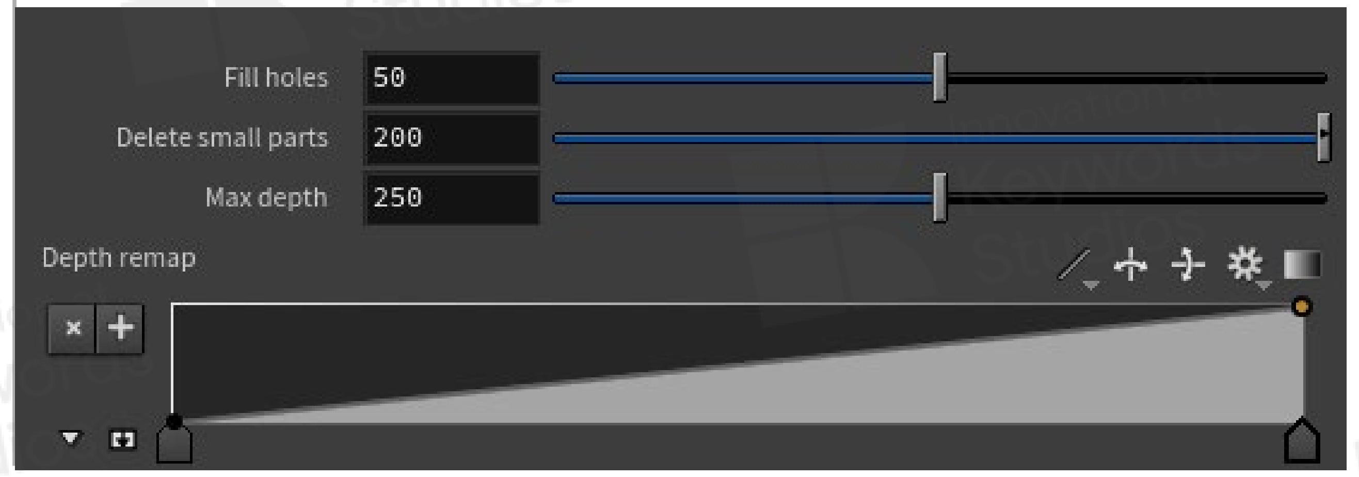
Image:

Background exclusion: 0

3

Adjustments

Set the parameters for displacement based on your specific requirements for background removal, geometry displacement and sides refinement.



Fill holes: 50

Delete small parts: 200

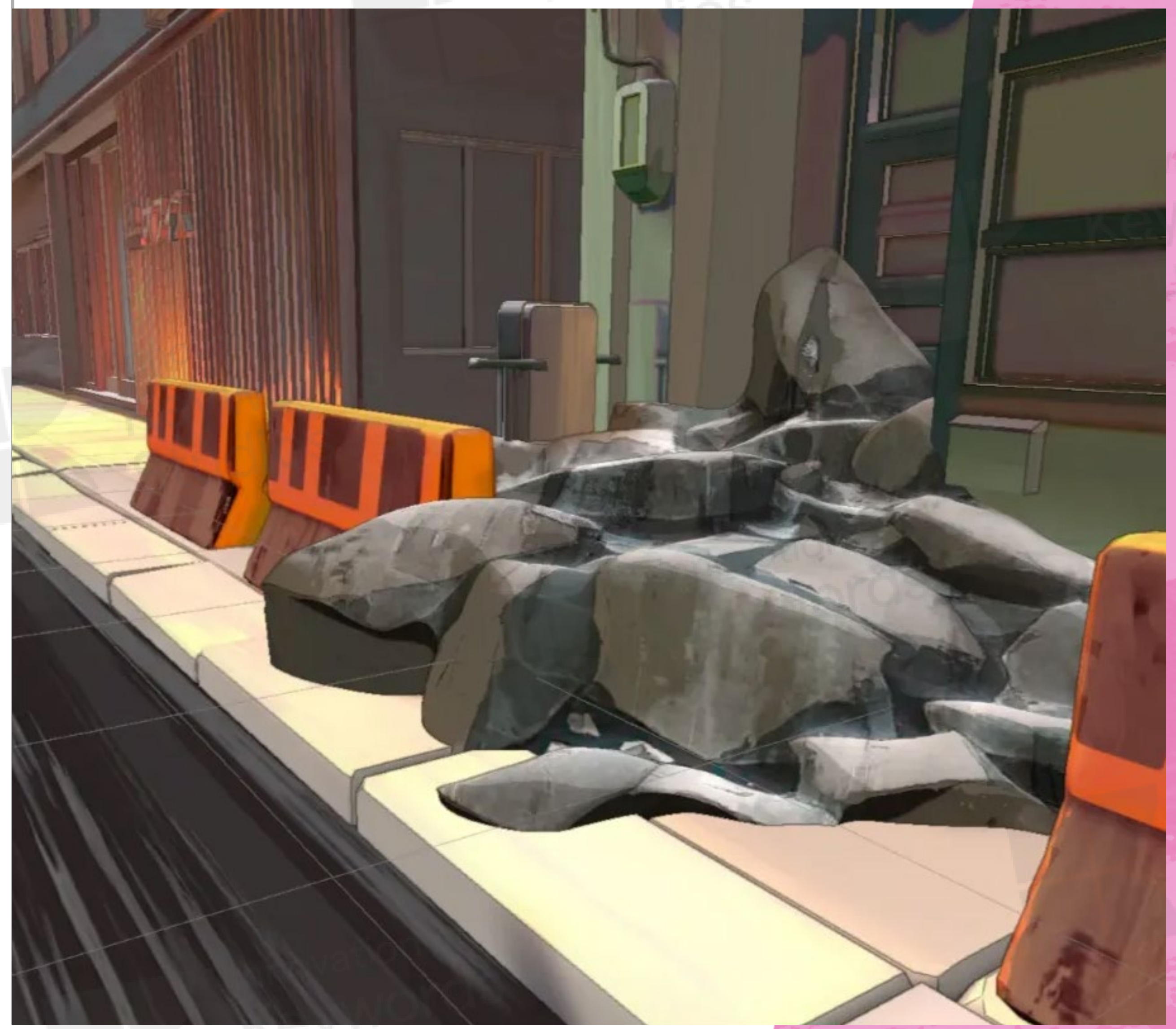
Max depth: 250

Depth remap

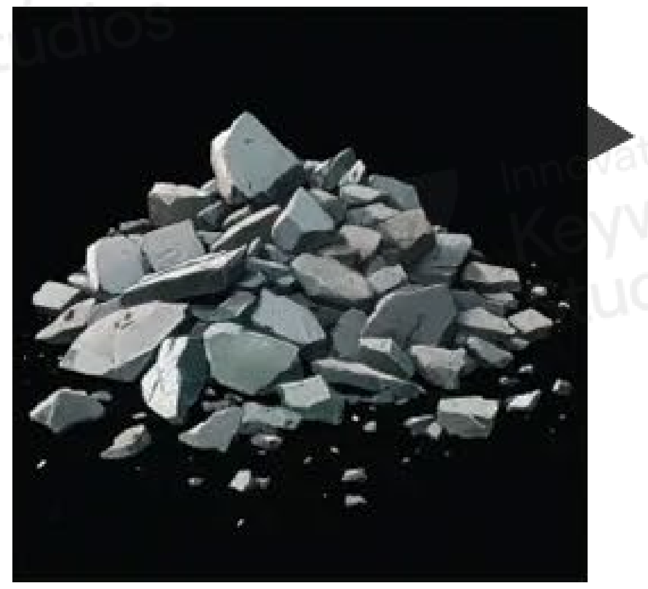
4

Integration

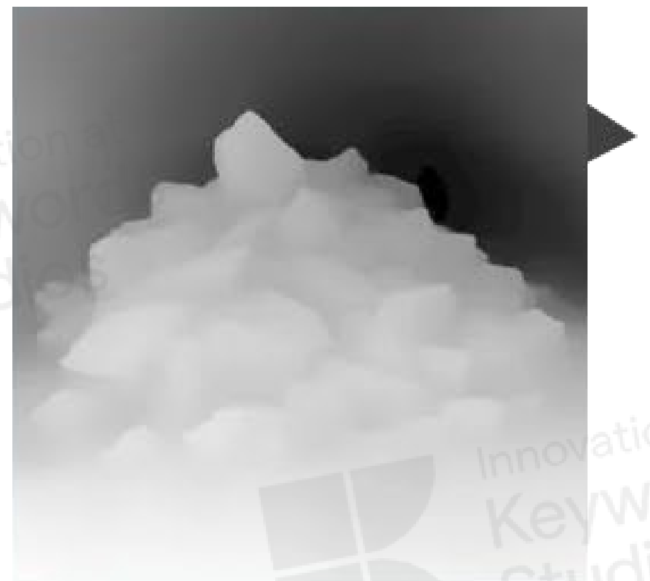
Export the model to engine. This step is not required if using the Houdini Engine plugin, as the result will immediately be imported.



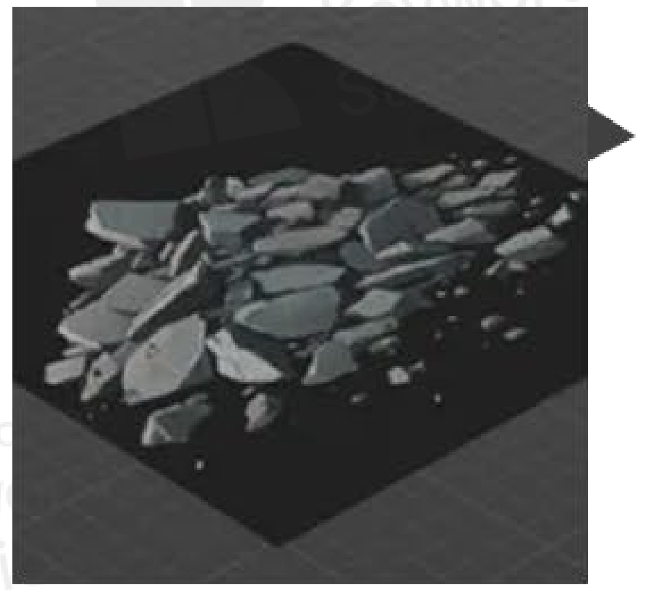
Example workflow from the AI Infused Pipeline:



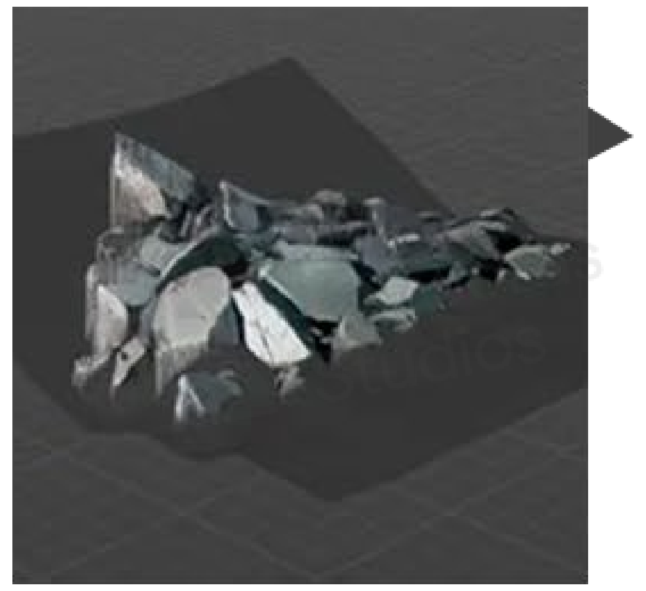
Input Image



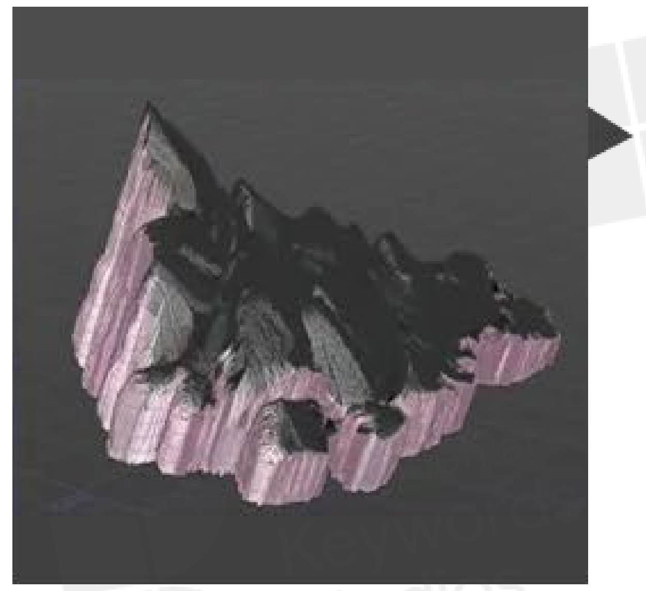
Generated Depth Map



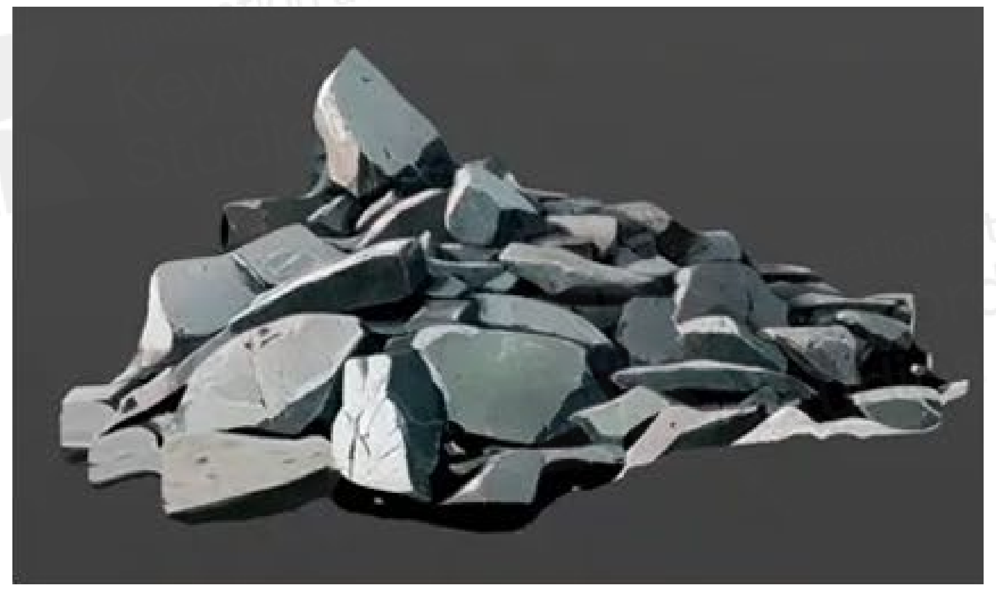
Applied to mesh



Offset Grid by depth



Cleanup and polish



Textured 2.5D Mesh

Inclusion of tools in this newsletter does not imply their clearance for use

PIPELINE SPOTLIGHT

Enhancing blockouts with 2.5D Modelling

Tools used: Houdini MLOPs

BEFORE
AFTER
CURTIS
CURTIS

Time to create debris and populate:	8hrs (mostly on texture creation)
Texture resolution:	512x512 (x1 - applied to all meshes)
Tri count:	1484 (x6 instances)
Unique mesh count (Debris Only):	x1

Time to create debris :	2hrs (Includes image gen and iterations)
Texture resolution:	512x512 (x3 - applied to all meshes)
Tri count:	2949 (x4), 3360 (x1), 13790 (x1).
Unique mesh count (Debris Only):	x3

Inclusion of tools in this newsletter does not imply their clearance for use

TEST CASE

2.5D Modelling Experiment

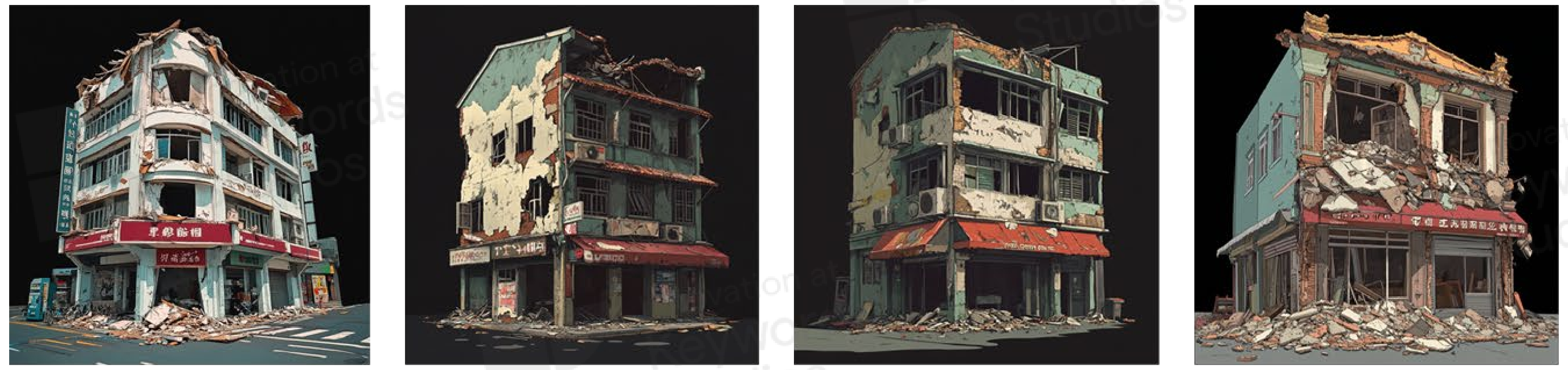
Generating a row of damaged buildings in 2hrs

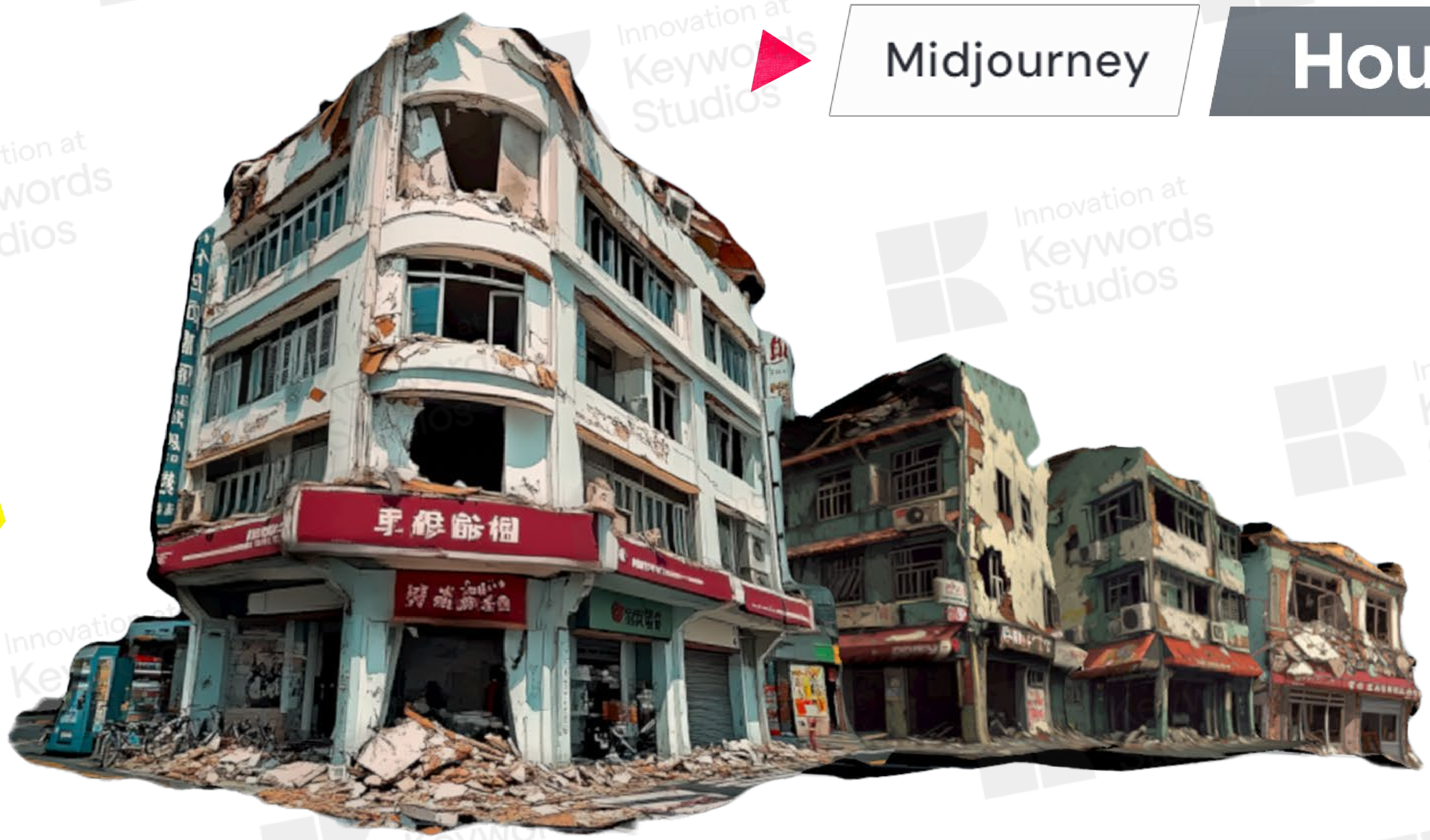
Tools used:

Midjourney

Houdini

MLOPs

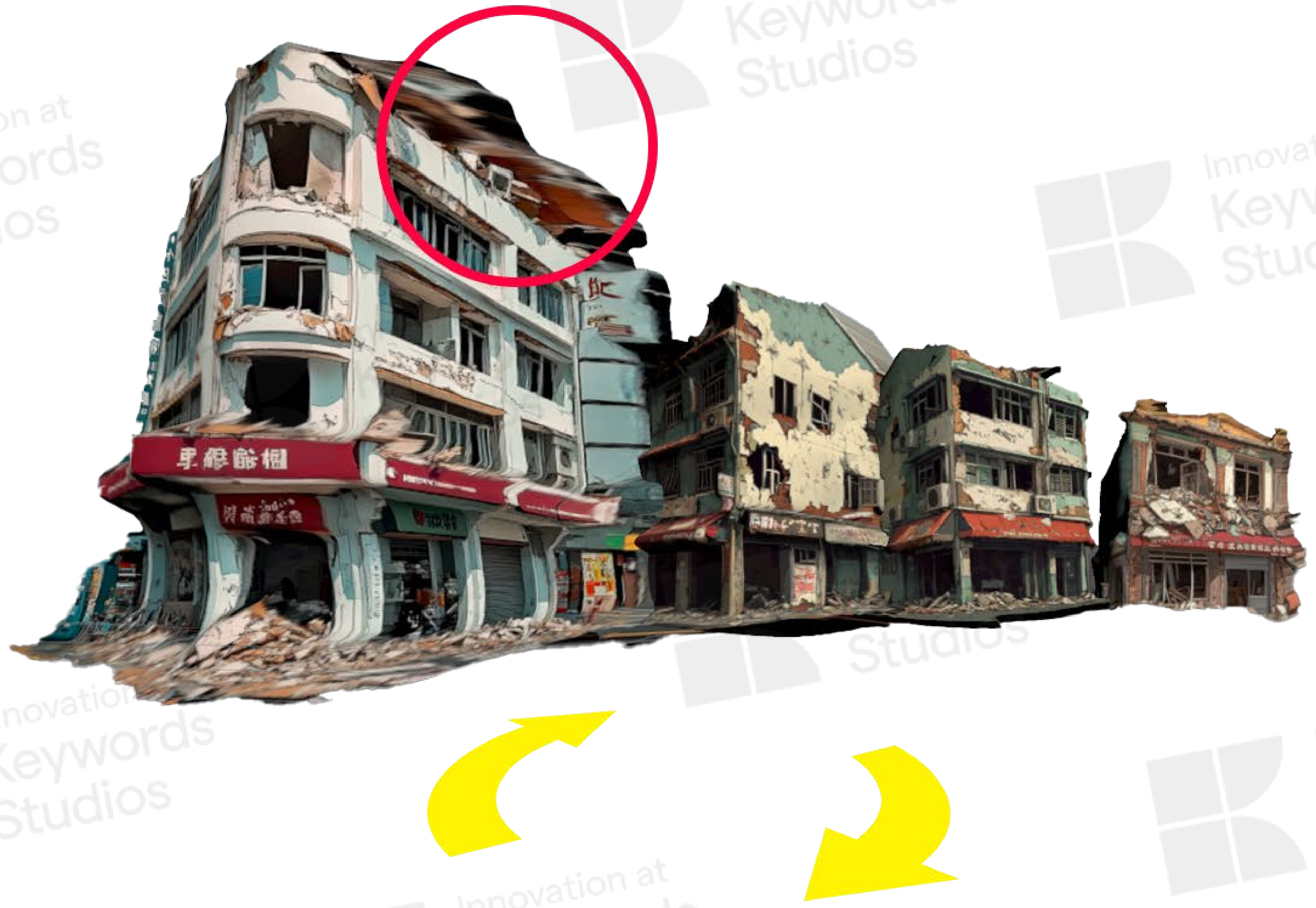


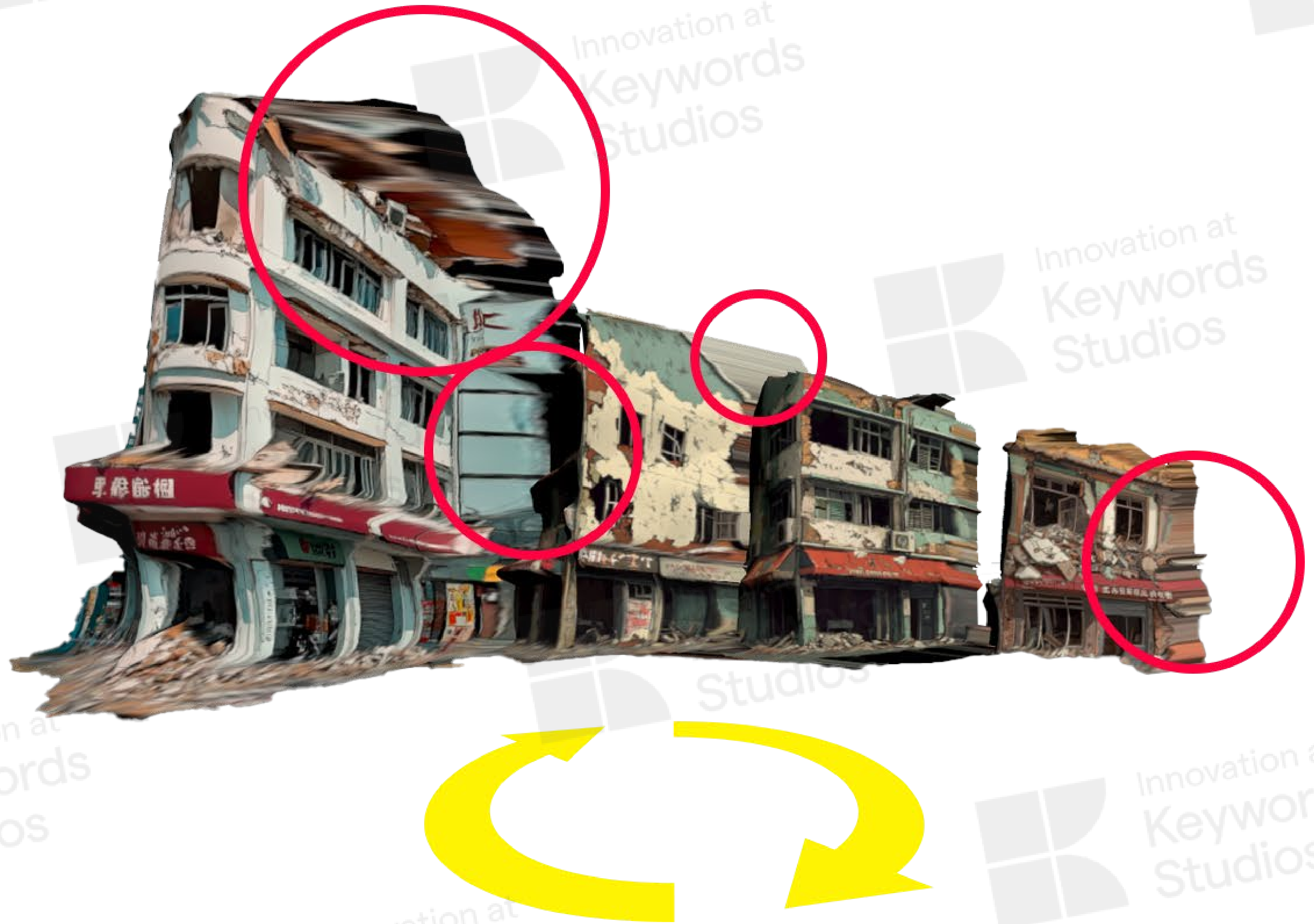


This quick experiment starts using an image generation model. We then used the images with the MLOPs projection technique to generate the 2.5D buildings. They look very convincing when arranged next to each other.

The buildings look good from this point of view and can tolerate a small amount of rotation, creating a very convincing 3D effect. However as you orbit or move past the models too much, the illusion can collapse.

As advised this technique is best used for objects seen from a limited POV or in the background. Because of this the results will be more convincing if the source imagery is generated at the POV you intend to see the model.





Inclusion of tools in this newsletter does not imply their clearance for use

TEST CASE

2.5D Modelling Experiment

Tools used:

Midjourney

Houdini

MLOPs

With the 2.5D buildings complete it occurred to us that we could quickly change their art direction using the retexture feature in Midjourney. This is a huge added benefit of using this technique during the blockout phase.

Thank you.

Featured in the next Supercharged:

- Exploring GAI Assisted Graphic Design
- Lighting Concept Art

Inclusion of tools in this newsletter does not imply their clearance for use