IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 12, May 2025
Impact Factor: 7.67

AI-Assisted 2D to 3D Level Generation in Unreal Engine 5

Mr. Nishant Naresh Gawade and Prof. Omprakash Mandge
MET Institute of Computer Science, Mumbai, India
mca23_1416ics@met.edu, omprakashm_ics@met.edu

Abstract: *Modern level design in game development demands considerable time and effort, especially when translating concept art or reference images into 3D environments. While Unreal Engine 5 provides tools for procedural generation, it lacks the ability to semantically interpret visual content, often leading to unsuitable asset placement. This research proposes an AI-enhanced framework to automate the conversion of 2D images into corresponding 3D environments. The process initiates with an object detection model like YOLO or a custom-trained convolutional neural network (CNN), which identifies objects in the 2D image and records their labels, sizes, and positions in a structured JSON format. Unreal Engine 5 then uses a custom C++ Actor to read this data, map it to appropriate 3D assets, and place them accurately in the game world. Experiments with varied test images—ranging from fruit arrangements to architectural layouts—demonstrated effective object detection and accurate 3D representation. The approach reduced manual workload, supported asset reuse, and retained visual consistency. This blend of computer vision and procedural design offers an innovative and scalable tool for game creators.*

Keywords: AI in Game Design, Unreal Engine 5, Object Detection, Procedural Generation, 2D-3D Translation, Computer Vision

I. INTRODUCTION

Translating 2D references into immersive 3D game environments is a common but labor intensive process for game developers. While game engines like Unreal Engine 5 offer
procedural tools for environment creation, they lack the intelligence to understand contextual elements in a reference image. As a result, designers often spend time manually arranging
assets to reflect the source. Incorporating AI into the level design pipeline introduces a way to reduce manual work and streamline content generation. Specifically, object detection models such as YOLOv8 can analyze images to extract meaningful features like object types and positions. This spatial data, when mapped into the game engine, allows for a structured recreation of the visual layout with minimal human input.

II. METHODOLOGY

The proposed workflow consists of two key stages:

1. Object Detection Using AI

A YOLOv8 model is used to process 2D images and identify key objects such as fruits, trees, or structures. Bounding boxes are generated for each detected object. These boxes are then processed using a Python script to calculate the center coordinates, which are normalized relative to the image resolution. This ensures consistency regardless of input size. The image space is divided into a 10x10 grid
where each cell corresponds to a fixed Unreal Engine unit (typically 100–200 units). Each detected object's position, along with its label and mapped grid location, is saved into a JSON file.

2. Integration with Unreal Engine

In Unreal Engine 5, a C++ class named AAutoSpawner is responsible for loading and parsing the JSON file. Each entry is read and matched to a predefined asset in an internal map that links object labels to 3D models. The engine then programmatically spawns each object at its respective world coordinate. This preserves the layout of the original image, allowing designers to quickly generate coherent and interactive scenes.

III. RESULTS AND DISCUSSION

To validate the system, a test image containing a bowl, banana, apple, and orange was used. The YOLOv8 model successfully identified all four objects. Their normalized coordinates were mapped to grid cells and then converted to world positions in Unreal Engine

Table 1: Object Detection and Spatial Mapping Output

Detected Object	Normalized Image Position (x, y)	Grid Position (x, y)	World Position (x, y)	Assigned Asset
Bowl	(0.502, 0.609)	(5, 6)	(500, 600)	SM_Bowl
Banana	(0.551, 0.315)	(5, 3)	(500, 300)	SM_Banana
Apple	(0.357, 0.437)	(3, 4)	(300, 400)	SM_Apple
Orange	(0.669, 0.473)	(6, 4)	(600, 400)	SM_Orange

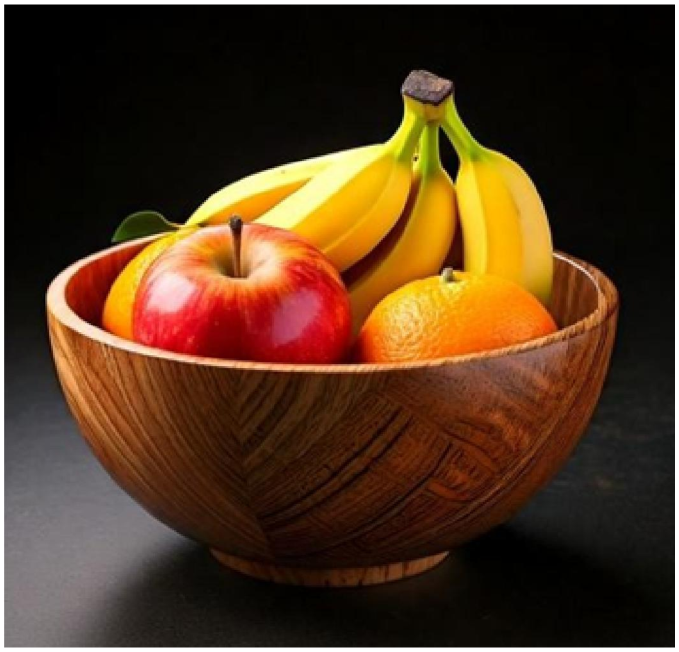
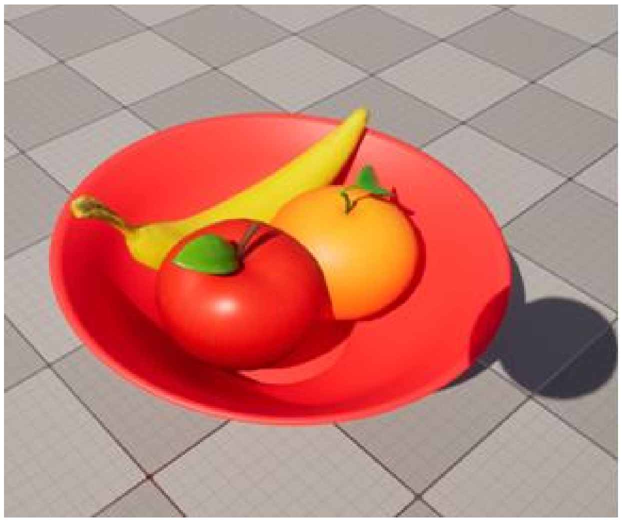


Figure 1

Each asset was accurately placed within the virtual scene, successfully replicating the image's composition. This confirms that the framework is reliable for practical use. Additionally, the architecture is adaptable—developers could extend it to include rotation data, object hierarchy, or real-time interactions in more advanced applications.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 12, May 2025
Impact Factor: 7.67


Final Result

By leveraging this automated system, designers significantly cut down development time. The solution is not limited to static scenes; it could be applied to dynamic scenarios such as AR applications, educational simulators, and quick prototyping platforms. The flexibility of the JSON structure and modularity of Unreal Engine's asset system allows easy customization for different genres or themes.

IV. CONCLUSION

This paper introduced a novel integration of object detection AI with Unreal Engine 5 for automated level generation. By translating 2D references into 3D layouts using YOLOv8 and a custom C++ spawner, the method offers a time-saving and accurate solution. The results
prove its effectiveness in maintaining spatial consistency and visual integrity. In future versions, adaptive behaviors, improved semantic parsing, and voice-controlled design tools could be explored to make game development even more intelligent and intuitive.

REFERENCES

[1]. Epic Games. (2023). Procedural Content Generation Overview. Unreal Engine 5 Documentation.https://dev.epicgames.com/documentation/en-us/unreal-engine/proced ural-content-generation-overview
[2]. Epic Games. (2023). Unreal Engine 5 Documentation. https://docs.unrealengine.com/5.0/en-US/
[3]. Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. (2023). YOLOv8 by Ultralytics. Ultralytics. https://github.com/ultralytics/ultralytics
[4]. Nicholas Renotte. (2021). Tensorflow Object Detection. https://youtu.be/yqkISICHH-U