iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

WILD ESCAPE: A Unity-Based Rule-Driven Wildlife Survival Simulation for Human-Wildlife Conflict Awareness

K. Manikanta[1], K. Suma[2], Y. Jagadeesh Kumar[3]
Sri Sivani College of Engineering. Andhra Pradesh, India

Abstract- Human-wildlife conflict is often represented in digital games through enemy-oriented animal behavior, while the ecological reason behind conflict is reduced to background text. WILD ESCAPE proposes a Unity-based wildlife survival simulation in which the animal is represented as a displaced survivor rather than a natural aggressor. The prototype uses a 3D environment, player-controlled animal movement, human threat detection, health feedback, combat/contact responses, level selection and cinematic camera handoff. The implementation is based on rule-driven game AI rather than machine learning: detection radius, target tags, attack range, NavMeshAgent movement, Animator clip control and collision/trigger events define the behavior loop. The ZenG Ultra scene and MMW scripts demonstrate a practical prototype with animal control, human chase behavior, menu flow and checkpoint-based gameplay start. The paper presents the system design, methodology, implementation modules and observed results, and identifies future directions such as smarter NPC behavior, multi-animal selection and possible data-driven analytics.

Index Terms- Game AI, Human-Wildlife Conflict, NavMeshAgent, Rule-Based Simulation, Unity 3D, Wildlife Awareness

I. INTRODUCTION

Human-wildlife conflict is a serious ecological and social problem, but many interactive systems simplify it by treating wild animals only as enemies. In reality, animal aggression often emerges when habitat loss, noise, fear, injury, territorial pressure or sudden contact with human settlements creates a survival threat. WILD ESCAPE is designed around this interpretation. The player controls a wild animal that must survive through forest, village and city-like spaces while avoiding or responding to human pressure.

The project was implemented as a Unity 3D prototype using C# scripts, Unity UI, NavMeshAgent navigation, Animator-based feedback, physics events and Cinemachine cameras. Its research value is not in training an ML model, but in showing how rule-driven game AI can communicate conservation awareness through player experience.

II. RELATED WORK

The IUCN guidelines on human-wildlife conflict and coexistence describe conflict as a complex interaction between human activities, wildlife needs and landscape-level pressures [1]. Digital simulation and game-based learning literature also supports the idea that interactive systems can make abstract environmental topics easier to understand by turning them into active decisions and feedback [2].

Unity provides practical tools for this type of prototype. NavMeshAgent supports movement over walkable surfaces [3], Animator components connect states with visual feedback [4], and Cinemachine supports third-person and cinematic camera workflows [5]. C# acts as the implementation language for modular gameplay logic [6].

III. PROBLEM STATEMENT AND OBJECTIVES

The problem addressed by this work is the limited representation of animal displacement and survival pressure in simple wildlife games. Existing game patterns often place the human as the hero and the animal as the enemy. This reduces the educational value of the interaction and does not highlight why conflict begins.

The objectives are: to design a player-controlled wildlife survival loop; to model human threats through rule-based AI; to integrate mobile-style controls, health feedback and attack/contact responses; to provide a menu and level-selection flow; and to document the system as a reproducible Unity prototype.

IV. PROPOSED SYSTEM

The proposed system represents the animal as the main playable character. The player selects the animal and level, views a short cinematic introduction and then receives control through joystick and action buttons. The animal can walk, run, jump and attack, while health decreases during hostile contact. Human NPCs detect the animal within a radius, move toward the target and apply attack or contact damage.

From the machine learning point of view, no external dataset, training set or learned model is used. The intelligence is rule-based game AI. Behavior is generated through scripted conditions such as target tags, target layers, detection radius, attack range, cooldown time, collision events and animation names. This is appropriate for a prototype because the behavior is transparent, easy to tune and lightweight for real-time gameplay.

V. SYSTEM ARCHITECTURE

The system is organized into scene, player, human AI, UI, camera and data modules. The ZenG Ultra scene contains approximately 289 GameObjects and 134 MonoBehaviour components. Terrains and building objects define the playable world. UI panels provide animal selection, level selection and in-game controls. The camera layer uses an intro camera and a FreeLook gameplay camera for handoff after the cinematic sequence.

TABLE I. IMPLEMENTATION MODULES

Module	Implementation role
AnimalController.cs	Camera-relative joystick movement, attack, jump, health, target matching, contact damage and VFX.
HumanClass.cs	Human detection radius, chase, attack timing, health slider and death state.
humanFightClass.cs	Alternate fighter NPC with random attack animation and collision damage.
MenuFlowController.cs	Animal selection, level selection and play-panel transition.
GameplayBootstrap.cs	Checkpoint spawn, Cinemachine intro camera, gameplay UI reveal and control unlock.

The main gameplay code is concentrated in AnimalController.cs (1198 lines), HumanClass.cs (747 lines), humanFightClass.cs (406 lines), DPadController.cs, MenuFlowController.cs, GameplayBootstrap.cs and GameData.cs. This separation keeps movement, AI, UI and startup flow understandable during testing and future extension.

VI. METHODOLOGY

The development method followed an iterative Unity workflow. First, the wildlife awareness concept was converted into playable requirements. Second, animal assets, level art, UI sprites, terrain objects and prefabs were prepared. Third, gameplay scripts were implemented and assigned through Inspector references. Fourth, detection, contact damage, health bar response and animation playback were tested in the scene. Finally, the project was documented as both a technical system and an educational simulation.

A. Rule-Based AI Design

The animal script uses joystick input, camera-relative direction calculation, NavMeshAgent movement, animation clip playback and health updates. Human scripts use Physics.OverlapSphere to locate valid targets, then rotate, chase and attack based on distance thresholds. Collision and trigger callbacks provide damage fallback when physical contact occurs.

B. No Dataset Usage

The project does not require a dataset because it does not train a classifier, detector, reinforcement learner or prediction model. The observations used for evaluation are functional test outcomes: movement works, target detection occurs, damage is applied, health UI updates and the camera handoff completes.

VII. IMPLEMENTATION

AnimalController implements the playable animal. It reads joystick axes, applies a deadzone, resolves movement relative to the camera and moves the NavMeshAgent. It also handles attack, jump, run state, health slider updates, death animation and contact VFX. The target-matching logic accepts explicit targets, tags, name prefixes or layer names, allowing the same script to be reused across animal prefabs.

HumanClass and humanFightClass implement threat behavior. HumanClass searches for the nearest target in a detection radius, plays idle/running/attack/death clips and applies damage after an attack delay. humanFightClass provides an alternate fighter pattern with random attack clips and collision damage. MenuFlowController stores the selected level, while GameplayBootstrap teleports the lion to the selected checkpoint, plays the intro camera and unlocks gameplay controls.

VIII. RESULTS AND DISCUSSION

The implemented prototype satisfies the major functional goals. The menu flow moves from animal selection to level selection and then into gameplay. The selected level index is preserved through GameData. The animal is teleported safely by disabling and re-enabling the NavMeshAgent before control is unlocked. Human NPCs detect the playable animal and apply attack/contact damage. The health slider reflects damage and death routines stop movement and play death feedback.

The result demonstrates that a lightweight rule-based system can communicate the project message without the complexity of ML training. The player experiences vulnerability, pressure and navigation difficulty from the animal perspective. This makes the conservation message more direct than a static paragraph, while keeping the system understandable for academic review and future implementation work.

IX. LIMITATIONS

The current prototype is limited by manually tuned AI values and Inspector references. Only a selected animal path is fully connected in the gameplay bootstrap, while additional animal images and assets require a broader selection registry. The human AI is radius-based and does not yet include advanced patrol, memory, group coordination or path prediction. Performance testing on low-end Android devices is also a future requirement.

X. CONCLUSION AND FUTURE SCOPE

WILD ESCAPE presents a Unity-based wildlife survival simulation that uses rule-driven game AI to explain human-wildlife conflict from the animal perspective. The prototype integrates player movement, human threat detection, health feedback, collision damage, UI controls, level selection and Cinemachine-based camera flow. The work shows that simple transparent AI rules can be effective for awareness-oriented gameplay.

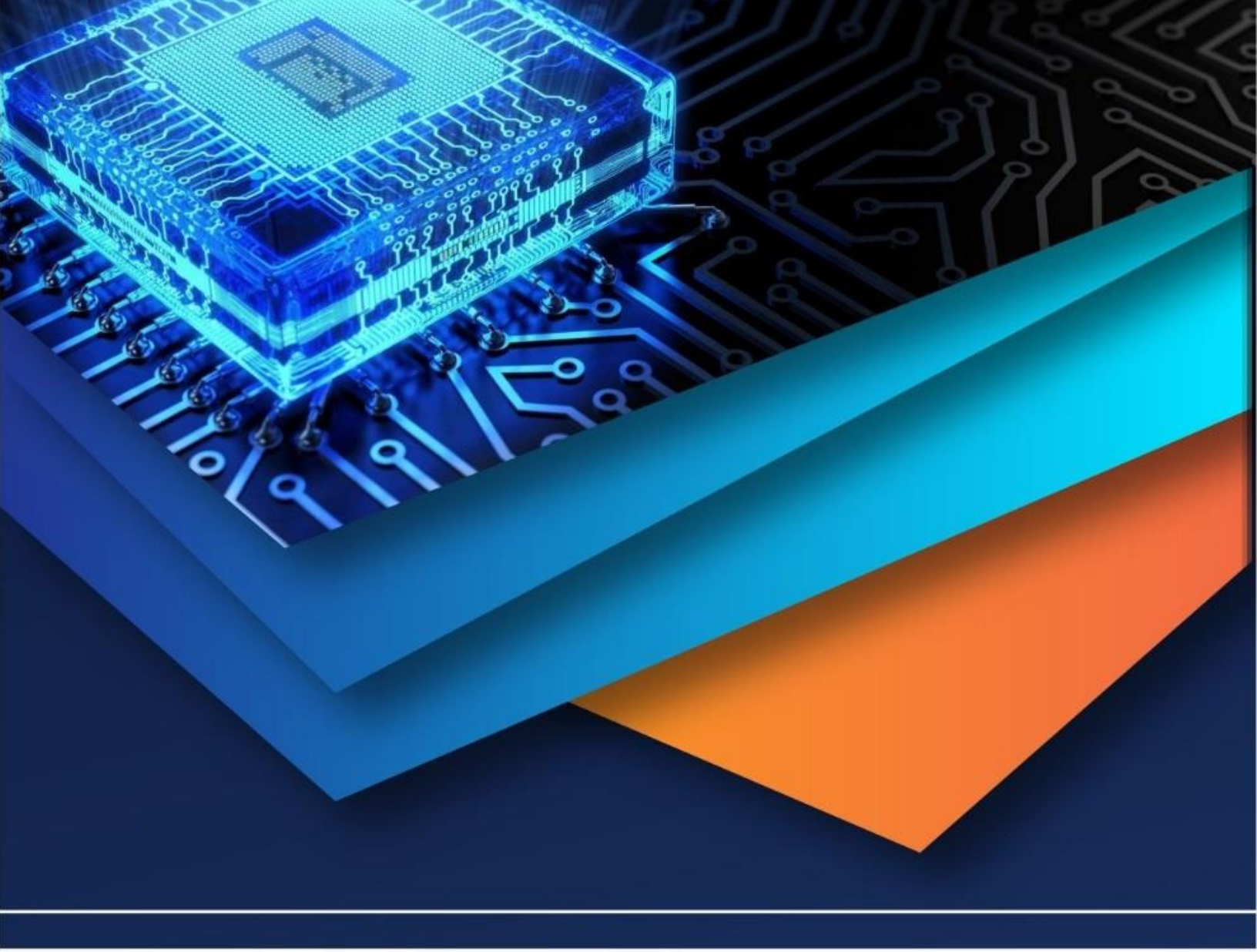
Future work can add multiple playable animals, richer ecosystems, patrol-based human behavior, animal stamina, rescue objectives, mobile optimization and analytics. If an ML extension is required, future gameplay logs such as position, health, collision count, route choice and escape/failure outcome could become a dataset for behavior prediction or adaptive difficulty. That dataset is not part of the present implementation.

XI. ACKNOWLEDGMENT

The author thanks the Department of Computer Science and Engineering, Sri Sivani College of Engineering, for academic support and guidance during the development and documentation of the WILD ESCAPE project.

REFERENCES

[1] IUCN SSC Human-Wildlife Conflict and Coexistence Specialist Group, IUCN SSC Guidelines on Human-Wildlife Conflict and Coexistence, 2023. [Online]. Available: https://www.hwctf.org/guidelines
[2] M. Prensky, Digital Game-Based Learning. New York, NY, USA: McGraw-Hill, 2001.
[3] Unity Technologies, Unity Manual: NavMesh Agent. [Online]. Available: https://docs.unity.cn/2019.1/Documentation/Manual/class-NavMeshAgent.html
[4] Unity Technologies, Unity Manual: Animator Controller. [Online]. Available: https://docs.unity3d.com/Manual/AnimatorControllers.html
[5] Unity Technologies, Cinemachine FreeLook Camera Documentation. [Online]. Available: https://docs.unity.cn/Packages/com.unity. cinemachine@ 2.9/manual/CinemachineFreeLook.html
[6] Microsoft, C# Language Reference. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/
[7] WILD ESCAPE project files, Assets/MMW/ZenG Ultra.unity and Assets/MMW scripts, local Unity project workspace, 2026.
[8] WILD ESCAPE project documentation, WILD_ESCAPE_PROJECT_DOCUMENTATION_TOC_WITH_SUBTOPICS_FINAL.docx, local project report, 2026.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY