Remnants – Game Prototype Documentation

Link to Games Resource folder:

https://goldsmithscollege-my.sharepoint.com/:f:/g/personal/njeya001_campus_goldsmiths_ac_uk/EoGPvuGHBiZMtB4zCV68fWMBYJI4rWTlS_cWBk_cT7GvQg?e=JzzWeO

PLAYER CONTROLS ON LAST PAGE

1. Abstract

Inspired by the architecture and mythology of the Indian subcontinent, Remnants is a third-person action role-playing game prototype created with Unreal Engine 5.5. The game, which took four months to develop, has systems for monitoring quests, character development, combat, and enemy behaviour. In a realm of ancient ruins, the prototype emphasises explorative gameplay and cultural themes through the environment. Realistic terrain design, fluid animations, and responsive AI are some of the main achievements. In addition to detailing the development process, key obstacles, and player comments, this document outlines the project's future objectives, which include meaningful cultural representation, more intricate gaming elements, and richer storyline.

2. Introduction

2.1. Project Overview

Remnants represents an early but focused exploration into technical game development within the action RPG genre. The current prototype showcases a compact village environment populated by basic NPCs, laying the groundwork for the game’s core systems and interactive mechanics. Though limited in scope, this build successfully demonstrates the underlying architecture designed to support more complex features in future iterations. Developed using Unreal Engine 5.5, the project highlights both technical competence and creative vision. As a foundational stage of a larger third-year university project, Remnants establishes a scalable, adaptable framework for future narrative, gameplay, and AI advancements.

2.2. Research Context

The development of Remnants sits at the intersection of several research domains within game studies:

- System Architecture Approach: The prototype uses a flexible blueprint-based structure that allows different gameplay systems—such as combat, stats, and AI—to be developed, tested, and updated separately. This design method reflects contemporary best practices in Unreal Engine development, making future expansions more manageable and efficient.

- Player Agency and Environmental Storytelling: Remnants explores narrative engagement through exploration, allowing players to uncover the story organically via environmental cues, ruins, and world design—minimizing reliance on direct exposition or dialogue.

- Character Control and Animation Systems: The prototype focuses on achieving a balance between smooth, high-quality animations and responsive player input, especially during combat and traversal, ensuring gameplay remains both immersive and mechanically satisfying.

- Cultural Representation in Game Design: Future development will explore a diverse range of cultural influences—including but not limited to South Asian, Middle Eastern, and East Asian aesthetics and mythologies—to create a richly layered game world. By moving beyond traditional Western fantasy tropes, Remnants aims to offer a unique visual identity and narrative depth that reflects global storytelling traditions.

2.3. Development Philosophy

Over the four-month development period, several key ideas shaped the direction of Remnants:

- Independent Systems: Each part of the game was built to work on its own while still being able to connect with others. This made testing, fixing bugs, and adding new features easier.

- Visual Authenticity: The look and feel of the environment was carefully chosen to reflect certain cultural styles while still being useful for gameplay.

- Player-Centric Design: All gameplay features were made to feel smooth and easy to control, giving players clear feedback during actions.

- Scalable Framework: The game systems were built to grow over time. Instead of using fixed rules, the project used flexible, data-based setups to make future updates easier.

This document explains how these ideas were used in building the environment, characters, combat, AI, and user interface. It ends with a summary of player feedback and a plan for what's coming next as Remnants moves into its future stages.

3. Project Timeline

The development of Remnants followed a structured timeline over the course of approximately four months. This section outlines the production phases in detail, highlighting key achievements and challenges.

3.1. Month 1–2: Environment Development

The first two months were dedicated entirely to crafting the environmental foundations of the game world. The goal during this stage was to create a visually immersive and thematically consistent landscape based on the idea of ancient ruins and the remnants of lost civilizations—drawing inspiration from the Indian subcontinent's historical and cultural backdrop.

3.1.1. Environment Creation Tools and Workflow:

- Gaea (Free Version): Used for procedural terrain generation. The heightmaps and texture maps were exported from Gaea and subsequently imported into Unreal Engine 5.5.

- Unreal Engine Landscape System: The terrain generated in Gaea was sculpted and textured using UE5's MWLandscapeAutoMaterial and custom material layers to provide realistic transitions between rock, soil, grass, and sand.

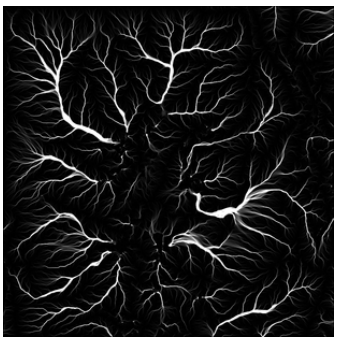
- Quixel Megascans Integration: To enhance visual fidelity and realism, a variety of scanned assets were used—such as broken stone pillars, eroded walls, and natural foliage—to populate the landscape.

- Custom Lighting: A dusk ambience was manually configured using UE5's directional light, sky atmosphere, and volumetric clouds. Additional fog effects were applied to create depth and mood.

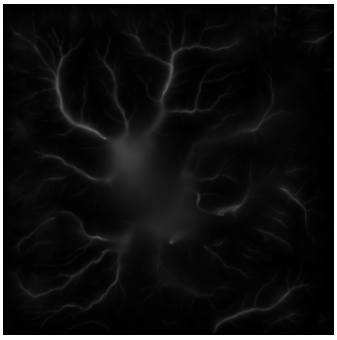
Texture Maps from Gaea:

Lake_Shore

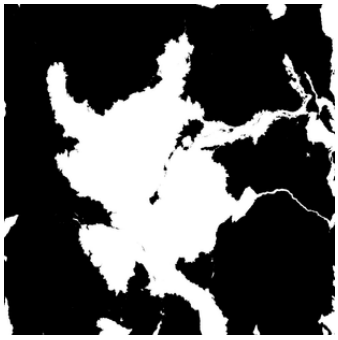
Lake_Water



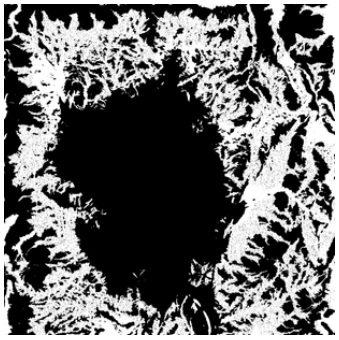
Erosion_Flow



Erosion_Wear



Trees_DeadZones



Snow_Snow

3.1.2. Environmental Areas Developed:

- Ruined Village: Representing the decay of civilization, this area showcases abandoned structures with culturally specific architectural elements, suggesting a once-thriving settlement.

- Shrine Area: Built as a mystical location using Megascans altar props and moss-covered stone structures. This area serves as both an aesthetic showcase and a narrative focal point.

- Cliffside Plateau: A high-altitude area meant for long-range visibility and sunset exposure. This dramatic viewpoint provides players with a sense of scale and perspective, reinforcing the game's themes of discovery and exploration.

3.1.3. Early Day/Night Cycle Implementation:

A day-night cycle was prototyped using Unreal Engine's experimental DaySequence plugin. Although functional, it revealed several technical challenges:

- The moon rotation did not align naturally with the sun path, causing lighting issues.
- Atmospheric fog behaved inconsistently, requiring further tweaking.
- Integration of celestial elements (stars, clouds) still requires post-processing tuning.

3.2. Month 3–4: Character & AI Implementation

With the environmental groundwork complete, the final two months of the initial phase focused on character movement, combat systems, AI behaviors, and the attempt to integrate a custom player character.

3.2.1. Character Design and Integration

Initially, the plan involved importing a custom-designed character model. However, Unreal Engine's default equipment system placed inventory sockets at the root of the skeleton. This configuration caused severe bugs when interacting with pickup items—especially during socket-switching animations for weapons and armor. Due to Unreal's rigid skeleton hierarchy, and without access to professional rigging tools, this part of the integration was abandoned for the current prototype.

Instead, the project reverted to the default character skeleton, refined through Control Rig and MetaHuman elements. Movement blueprints were expanded to allow for:

- Directional running and sprinting
- Vaulting, climbing, and dodging
- Weapon-based locomotion transitions

3.2.2. Character Blueprint System

BP_ThirdPersonCharacter serves as the main player class, equipped with modular components that handle different aspects of gameplay.

Key components include:

- *BPC_playerStat*: Handles health, stamina, mana, level, attributes, and status effects
- *BPC_AttackSystem*: Manages offensive actions, damage calculations, and attack chains
- *BPC_EquipmentSystem*: Controls gear, stats, and visual representations

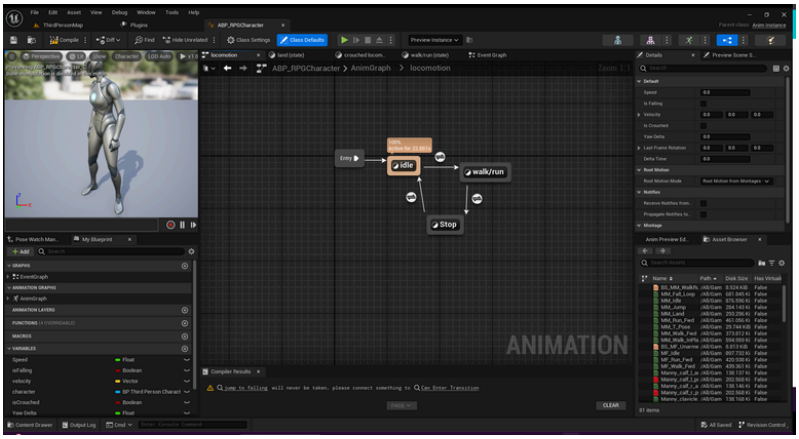
3.2.3. Combat Blueprint System

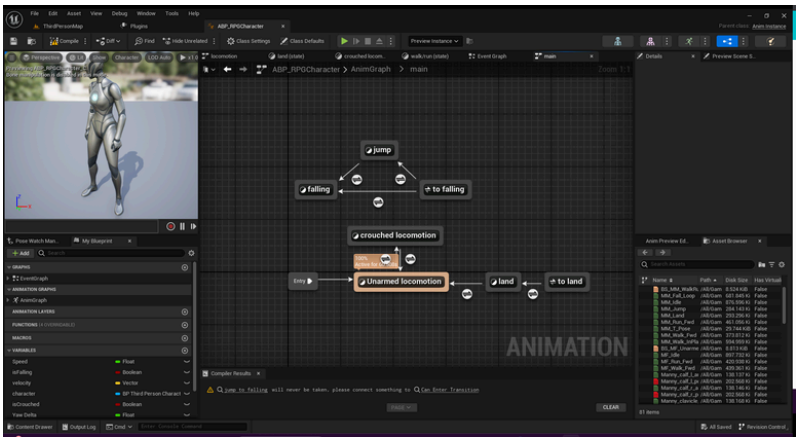
The combat system was built using modular Blueprints with damage trace channels:

- Animation notifies for sword attacks and arrow firing
- Projectile systems for ranged combat
- Test enemies with hit detection and reaction logic (error in current prototype)

Animation integration was fine-tuned using:

- Control Rig for dynamic head tracking
- IK foot placement for terrain adaptation
- Blend spaces for varied movement animations

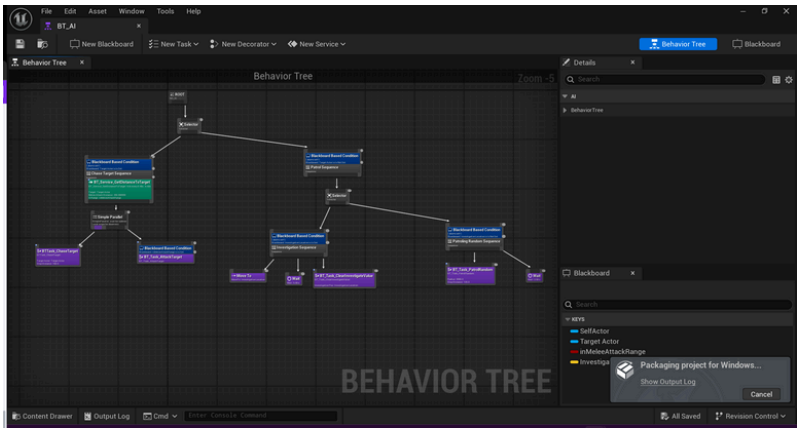




3.2.4. AI Characters

AI enemies were introduced using Behavior Trees and Blackboard logic:

- Generic AI controller for perception and response
- Behavior tree for patrolling, chasing, and combat
- Blackboard for storing decision variables
- AI-specific animation blueprint



Types of AI included:

- Standard melee fighters
- Non-combatant NPCs

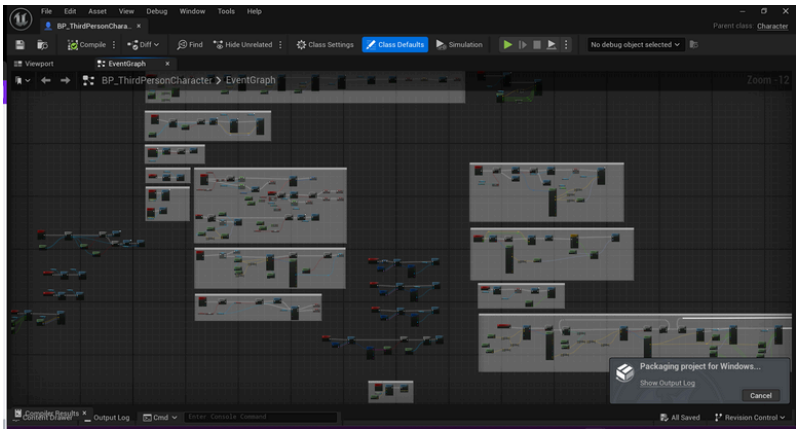
4. Deep Dive: Blueprint Architecture

The modular and component-based design was central to the Remnants prototype. Each major game system—combat, movement, animation, and inventory—was implemented as a standalone component or blueprint class that could be debugged, reused, or swapped independently.

4.1. Central Character Class: BP_ThirdPersonCharacter

This is the primary actor class for the player, housing several essential functions:

- Spawning and possession
- Equipment socket overrides
- Animation Blueprint assignment
- Player input routing



Key Internal Variables:

- Combat mode state tracking
- Currently equipped weapon type
- Movement speed modifiers

Functions:

- Weapon switching system
- Damage handling
- Movement action triggers

4.2. Blueprint Components

4.2.1. BPC_playerStat

Handles all stats related to the character:

- Level progression and experience tracking
- Resource management (health and stamina)
- Status effect system

The component includes several interconnected systems:

- Attribute Calculation System: Dynamically computes derived stats based on core attributes
- Level-Up Manager: Handles experience gain and attribute point distribution
- Resource Management: Controls regeneration rates with configurable curves
- Status Effect Framework: Applies temporary effects with variable durations

4.2.2 BPC_AttackSystem

Manages the combat mechanics:

- Combo chain sequencing
- Damage calculations with critical hits
- Attack range and hitbox detection
- Animation-combat synchronization

Key subsystems include:

- Combo Chain Controller: Tracks attack timing for combo continuations
- Damage Calculation Pipeline: Processes damage through various modifiers
- Hitbox Generation: Creates collision volumes during attacks
- Weapon Type Integration: Adapts behavior based on equipped weapon

4.2.3. BPC_EquipmentSystem

Responsible for inventory and equipment:

- Gear slot management
- Skeletal mesh updates for visual representation
- Stat modifications from equipment
- Item durability tracking

The system includes:

- Socket Management: Controls attachment points for equipment
- Stat Modification: Applies equipment bonuses to player stats
- Item Database Integration: Connects to centralized item definitions
- Durability System: Tracks wear and performance degradation

4.3. Blueprint Interfaces

Interfaces were used to standardize communication between systems:

- Quest interaction protocols
- AI communication systems
- Stealth mechanics

4.4. Reusable Blueprints

- Weapon base class with damage profile and animation settings
- Projectile system for arrows and throwable objects
- Quest giver NPC with dialog integration
- Training dummy for combat testing

5. Animation Systems and Combat Framework

5.1. Animation Blueprint Architecture

The animation system represents one of the most complex aspects of Remnants, controlling all character movements and combat actions.

5.1.1. State Machine Design

- Locomotion State: Handles walking, running, and sprinting with directional blending
- Combat State: Controls attack animations, blocks, and parries
- Special Movement State: Manages climbing, vaulting, and dodging
- Hit Reaction State: Processes damage responses of varying intensities

5.1.2. Animation Layering System

The blueprint implements sophisticated layering to allow simultaneous upper and lower body animations:

- Lower body maintains locomotion while upper body performs actions
- Additive layers for head tracking and hand IK adjustments
- Dynamic adjustment of blend weights based on player state

5.1.3. Procedural Animation Elements

- IK Foot Placement: Adapts foot positions to uneven terrain
- Dynamic Hand Positioning: Adjusts grip on weapons based on weapon type
- Procedural Leaning: Adds natural body weight shifts during direction changes

5.2 Combat Animation Framework

The combat system relies on precise animation triggers and notifies:

5.2.1. Animation Notifies

- Sword hitbox creation during swings
- Camera shake effects for impact feedback

5.2.2. Attack Chains

- Light attack sequences (3-4 hits per chain)
- Heavy attack sequences (2-3 hits with longer recovery)
- Mixed chains combining light and heavy attacks
- Finisher animations with enhanced damage

5.2.3. Combat Mobility

- Dodge rolls with invincibility frames
- Combat strafing with weapon-ready stance
- Backstep and lunge mechanics for spacing control

6. Quest and Progression Systems (Not fully implemented in the prototype)

6.1. Quest Architecture

The quest system was developed to support narrative progression and player guidance:

6.1.1. Quest Component

The central quest will management system handles:

- Active and completed quest tracking
- Objective completion logic
- Reward distribution
- Quest state persistence

6.1.2. Data Structures

- Quest Definition Structure: Core parameters including description, difficulty, prerequisites, and rewards
- Objective Structure: Individual goal tracking with quantities and conditions

6.1.3. Quest Distribution System

- NPC quest givers with interaction capabilities
- Dialog system integration
- Quest availability based on player progress
- Visual indicators for available quests

6.1.4. Quest UI Framework

- Main quest tracking interface with progress indicators
- Dialog system for NPC interactions
- Objective markers with distance tracking
- Reward preview system

7. UI Systems and Player Feedback

7.1. User Interface Architecture

The UI was designed to be minimalist yet informative, providing essential feedback without overwhelming the screen:

7.1.1. HUD Components

- Primary gameplay interface with health and resource bars
- Enemy health indicators
- Minimap with objective markers
- Combat state information

7.1.2. Interaction System

- Contextual prompts for stealth and investigation
- Environment interaction indicators
- Quest-related prompts

7.1.3. Menu Framework

- Game navigation interface
- Equipment and inventory management
- Settings and configuration options

7.2. Visual Feedback Systems

Beyond UI elements, several systems provide player feedback:

7.2.1. Combat Feedback

- Hit effects and directional indicators
- Enemy reaction animations
- Weapon impact effects (to be fully implemented)
- Screen effects for player damage (to be fully implemented)

7.2.2. Interaction Feedback

- Object highlighting
- Contextual button prompts
- Camera adjustments during interactions
- Environmental effects

8. AI and NPC Systems

8.1. AI Architecture

The enemy AI system was built around behavior trees and blackboard data:

8.1.1. Core AI Blueprint

The main enemy blueprint manages:

- Perception system for detecting players
- Navigation and movement control
- Combat action execution
- Animation state management

8.1.2. AI Controller

The controller component handles:

- Behavior tree execution
- Decision variable storage
- Player detection logic
- State transition management

8.1.3. Behavior Tree Structure

The behavior tree organizes AI decision-making through:

- Hierarchical task organization
- Priority-based decision making
- Conditional branches for different situations
- Continuous evaluation services

8.1.4. AI Animation Control

The AI animation blueprint manages:

- Movement animation blending
- Attack animation selection
- Reaction animations
- Idle variations

8.2. AI Types and Behaviors

Different enemy types exhibit specialized behaviors:

8.2.1. Combat Enemies

- Melee attackers with approach and engagement patterns
- Special enemies with unique abilities and attack patterns

8.2.2. Non-Combat NPCs

- Civilian routines with daily schedules
- Reactive behaviors to player presence
- Ambient animation sets for environmental storytelling

9. Technical Challenges and Solutions

9.1. Equipment System Challenges

The equipment system posed several technical challenges:

- Socket Alignment: Equipment attachment points frequently misaligned during animation transitions
- Solution: Custom socket transform adjustment based on animation state
- Weapon Switching Lag: Weapon switching is not functional in this prototype

9.2. Animation Blending Issues

Several animation challenges were addressed:

- Transition Popping: Abrupt changes between animation states
- Solution: Implementation of custom blend curves and transition poses
- IK Foot Placement: Feet occasionally clipping through terrain
- Solution: Ray-casting adjustment with smoothing parameters

9.3. AI Pathfinding Optimization

AI movement required several refinements:

- Navmesh Generation: Complex environment caused navigation holes
- Solution: Manual navmesh adjustments and dynamic navmesh rebuilding
- Group Behavior: AI units frequently collided or blocked each other
- Solution: Implementation of formation-based movement and spacing awareness

9.4. Performance Optimization

To maintain acceptable frame rates on target hardware:

- Level of detail systems were implemented for distant objects
- Asset streaming was configured to load environmental elements dynamically
- Instanced static meshes were used for repetitive environmental elements
- Occlusion culling was fine-tuned to minimize rendering overhead

10. Project Exhibition and Feedback

Remnants was showcased at a university exhibition, offering an invaluable opportunity for peer and instructor feedback. The prototype was packaged and fully playable, allowing testers to explore the environment, engage in basic combat, and interact with the game world.

10.1. Feedback Highlights:

- Visual Quality: Players responded positively to the realism and detail of the environment, particularly the lighting and use of ruins. The architectural elements were especially praised for their uniqueness in the gaming landscape.

- Combat Feel: The chaining of light and heavy attacks was praised, though responsiveness could be improved. Some players noted that the weight of attacks felt satisfying, but dodge timing needed adjustment for better combat flow.

- Environment Navigation: Vaulting and climbing added to player agency, but some transitions felt stiff due to animation blending issues. The traversal mechanics were seen as a strength, though certain climbing animations needed refinement.

- UI Design: Minimalist and intuitive, but quest tracking and interaction indicators needed more clarity. Players appreciated the uncluttered screen but occasionally missed important prompts during intense gameplay.

- Conceptual Systems and Future Features: While not yet fully developed, several planned systems such as branching dialogue, faction dynamics, and morality-based quest outcomes received interest from testers during early presentations. Players expressed enthusiasm for deeper narrative integration and more meaningful player choices, suggesting that these features could significantly enhance engagement. Feedback emphasized the importance of clear consequences and culturally grounded storytelling, highlighting a strong appetite for gameplay systems that go beyond combat to explore the game's world and themes more deeply.

This feedback will help inform future updates and adjustments to the core gameplay loop and interface.

11. Future Plans and Year 3 Vision (2026)

Remnants is not a final product but a foundation. The prototype sets up a scalable framework for the third-year final project. As development continues into 2026, the next phase will focus on the following:

11.1. Narrative Development

The story will center around ancient ruins scattered across a fictional landscape, deeply inspired by architecture, legends, and philosophies from different cultures including but not limited to South Asian, Middle Eastern, and East Asian aesthetics. The goal is to evoke a sense of loss, mystery, and rediscovery through environmental storytelling, inscriptions, and lore items.

Key narrative elements to be expanded:

- Ancient civilization backstory with cultural specificity
- Faction conflicts rooted in philosophical differences
- Player-driven narrative choices affecting the game world
- Environmental storytelling through ruins, artifacts, and inscriptions

11.2. Gameplay Expansion

- Quest branching systems with moral choices
- Unique factions with ideological conflicts
- Skill trees for player specialization
- Dynamic event systems and side-quests

Combat mechanics will be refined with:

- Expanded weapon variety with unique movesets
- Advanced combo system with stance switching
- Parry and counter mechanics
- Elemental effects and status interactions

11.3. Cinematics and Sound

- Cutscenes using Unreal's Sequencer
- Ambient soundscapes and adaptive music tracks
- Voice-over work for immersive storytelling
- Dynamic audio mixing based on environment and tension

11.4. AI and Factions

- Advanced AI routines for different faction behavior
- Group combat dynamics and ambush logic
- Dialogue trees for NPCs with faction alignment
- Reputation system affecting world interactions

11.5. Performance and Deployment

- Optimization for mid- and high-tier PCs
- Controller support

12. Conclusion and Research Outcomes

The Remnants prototype represents the successful implementation of a modular game framework capable of supporting a narrative-driven action RPG. The component-based architecture ensures that future development can build upon this foundation without significant refactoring.

Key research outcomes include:

- Modular Blueprint Design: The effectiveness of component-based design was demonstrated through the ease of system iteration and extension.
- Cultural Integration: The prototype successfully incorporated architectural elements, proving the creation of distinctive and engaging game environments.
- Animation System Complexity: The research highlighted the challenges of creating responsive character animation systems, particularly the balance between animation fidelity and gameplay responsiveness.
- AI Behavior Framework: The behavior tree implementation demonstrated a scalable approach to enemy variety, suggesting that further faction-based AI expansion is feasible.

The Remnants prototype stands as both a technical achievement and a foundation for future development, embodying the research goal of creating culturally distinctive game worlds with engaging gameplay systems.

13. Technical Implementation Details

13.1. Unreal Engine 5.5 Features Utilized

The project leverages several advanced features introduced in Unreal Engine 5.5:

- Enhanced Input System: Employed for responsive character controls with contextual action mapping.
- MetaHuman Integration: Incorporated for realistic character models and facial animations. (To be fully implemented in the next model)

- Plugin Ecosystem: Several plugins enhanced development capabilities:

    - ModelingToolsEditorMode for advanced environmental modeling
    - MotionWarping for smooth animation transitions
    - Landmass for procedural terrain generation
    - DaySequence for time-of-day management
    - LiveLinkControlRig for enhanced animation control
    - AppleARKitFaceSupport for facial animation capabilities

13.2. Project Structure and Organization

The project's file structure was organized to facilitate collaborative development and system independence:

- Content Organization:
    - Character-related assets contained in dedicated directories
    - Environmental elements organized by area
    - Systems maintained in functional categories
    - UI elements centralized for consistent styling

- Blueprint Hierarchy:
    - Parent classes established for major game elements
    - Child classes extended functionality without duplicating core logic
    - Interfaces ensured consistent communication between systems

This organizational approach will support the project's continued development as it evolves toward its full implementation in the third-year curriculum.

Player Controls

The movement mechanics include:

- W, A, S, D: Standard movement
- Spacebar: Jump
- E: Sprint
- C: Crouch (adjusts capsule height and triggers crouch animation)
- V: Vault (contextual vaulting over low obstacles using motion warping)
- K: Climb (WIP feature; current implementation supports climbing flat vertical surfaces)
- F: Dodge Roll (triggered on input; enables rapid evasion)
- G: Interact (collect items, open chests, and initiate dialogue with NPCs)
- P: Throw stones to distract patrolling NPCs
- Right Mouse Button to use when RMB option shows up on screen
- Left Mouse Button to attack using the weapon equipped

These actions are tied to animation states managed in the character's Animation Blueprint. Transitions are handled via a State Machine, and blendspaces enable smooth transitions between idle, walk, run, and combat states.