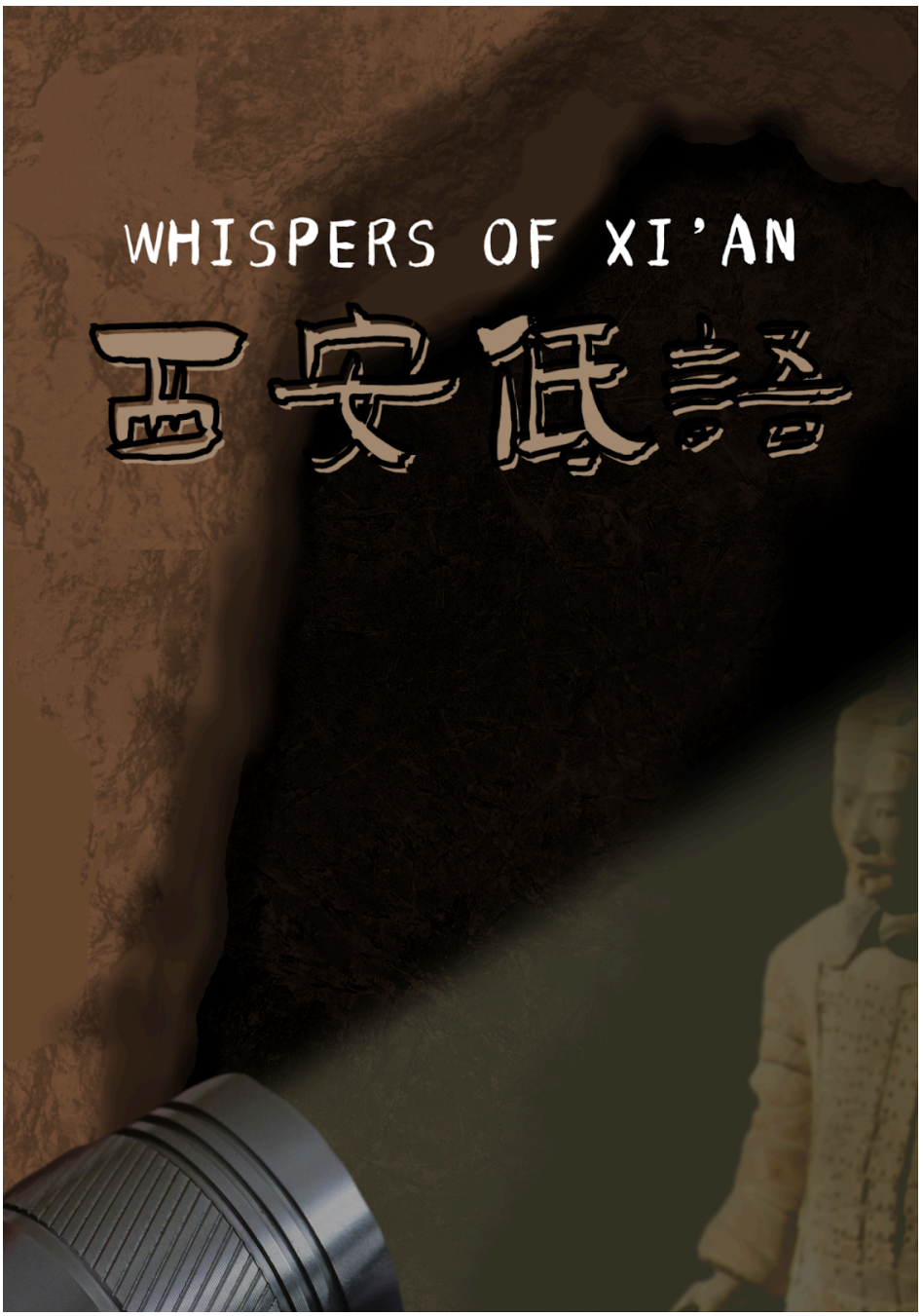


Whispers Of Xi'an

Document version number 1.0.0

Written by Lara Lowndes, Matthew Morrissey, Tom Hannam, Vilius Vaicekauskas

Date of Publishing 19/12/25

Version number 1.0.0

Table of Contents

Game Engine

The game is developed using Unreal Engine 5.5.4, using both C++ and Blueprint. Using Unreal Engine and the Unreal Development Kit (UDK) provides the user with a robust game development environment and tools such as:

- Animation
- Navmesh
- UI
- Unreal Node Based Scripting
- C++ Development
- Audio Systems
- Physics Systems
- Interfaces

Coding and Naming Conventions

Whispers of Xi'an follows the coding and naming standard as suggested in the Unreal Engine documentation (Unreal Engine, 2025). PascalCase is used across the project to maintain clarity and consistency. Naming conventions are followed throughout the project with determined prefixes indicating item types as suggested in *Recommended Asset Naming Conventions* by Epic Games (Unreal Engine, 2025).

- U for UObject.
- A for Actors.
- I for Interfaces from C++.
- BPI for Interfaces from Blueprints.
- ABP for Animation Blueprints.
- BP for Blueprints
- SM for Static Mesh
- SMK for Skeletal Mesh

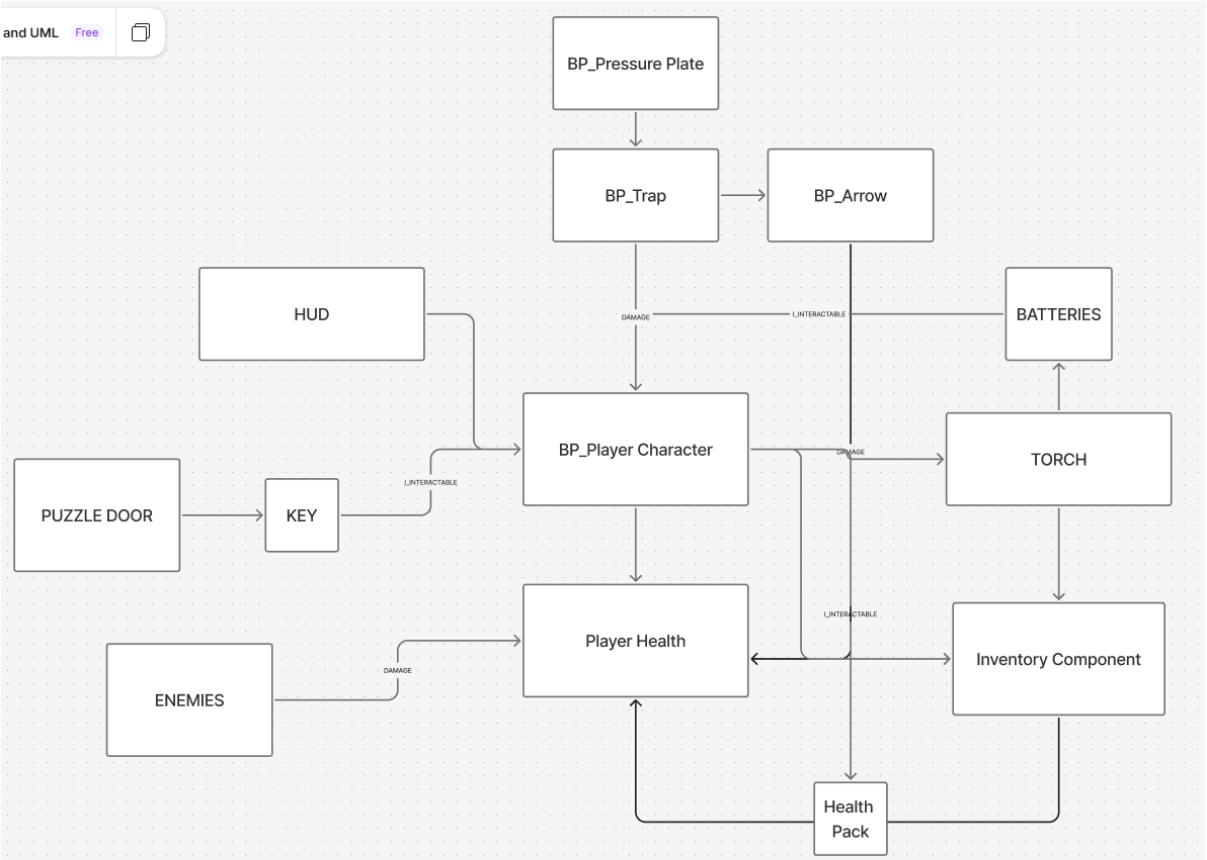
Using the Unreal Engine recommended naming conventions ensures that all content, whether created ourselves or sourced externally, remains consistently organised. As this is the standard across Unreal projects it ensures when integrating third-party assets from platforms such as the FAB Store or Unreal Marketplace, the naming convention remains consistent and helps with seamless integration and improves the efficiency of the development pipeline.

Minimum Specifications

Component	Minimum
CPU	Quad-core (Intel i5-7500/ Ryzen 3 1200)
GPU	NVIDIA GTX 1060 or AMD RX 580 (DX12 Capable)
RAM	8GB
Storage	10GB

OS	Windows 10 64-bit

Player Class Diagram



(Figma, n.d)

List of Software & Tools

- **Unreal Engine 5.5.4**
  - **What is it:** A 3D creation engine developed by Epic Games, used for game development, simulation, visualisation, and interactive media.
  - **What it is used for:** Acts as the primary development environment for Whispers of Xi'an. All development workflows and pipelines, Blueprints, C++ integration, levels, lighting, materials, animations, and packaging are implemented within UE 5.5.4.
- **Canva (Pro)**
  - **What is it:** A web-based graphic design tool used for creating visual assets, mock-ups, and layout concepts.
  - **What it is used for:** Used for designing UI mockups, concept screens and menu layouts.
- **Figma**
  - **What is it:** Figma is a design platform used for UML diagrams and wireframing through a browser-based environment.

    - **What is it used for:** Figma is used to create UI, menu layout and HUD and level design mock-ups and wireframes and flow diagrams.
- **Machinations**
    - **What is it:** Machinations is a browser based platform for designing and simulating complex game systems that has Monte Carlo simulations available to test your designs.
    - **What is it used for:** Machinations is used to model and balance our internal economies, particularly battery drain and health management.
- **Google Suite**
    - **What is it:** A collection of cloud-based productivity and collaboration tools from Google, including Google Docs, Google Sheets and Google Slides.
    - **What it is used for:** Used throughout development for documentation, project notes, research collation, and storing shared references. Using the Google Suite enables real-time collaboration and version history for all documentation.
- **Visual Studio 2022**
    - **What is it:** An Integrated Development Environment (IDE) used for writing, compiling, and debugging C++ code, with built-in integration for Unreal Engine development.
    - **What is it used for:** Serves as the main environment for writing the project's C++ gameplay systems and for compiling the project.
- **GitHub Desktop 3.5.4**
    - **What is it:** A GUI for Git version control that simplifies commit management, branching, and repository control.
    - **What is it used for:** Used for source control, reviewing file changes, resolving merge conflicts, and synchronising local development progress with the repository.
- **GitBash**
    - **What is it:** A command-line interface that provides access to Git commands enabling more advanced control over version and file management**.**
    - **What is it used for:** Used for resolving merge conflicts on a file by file basis, branch merging, and for enabling Git LFS.
- **Discord**
    - **What is it:** A free communication platform that supports text, voice, screen sharing, and file exchange.
    - **What it is used for:** Used for the main way of team communication, collaboration, and quick troubleshooting.
- **FAB Store**
    - **What is it:** The FAB Store, previously known as the Unreal Marketplace, is Epic Games' official asset platform. Hosting thousands of assets including 3D models, animations, materials and sound packs that can be licensed under the Unreal Engine EULA.
    - **What it is used for:** FAB Store assets are used to populate the game world and particularly the demo with environmental props and 3D models to aid with rapid prototyping using premade assets.
- **Mixamo**

    - **What is it:** Mixamo is Adobe's 3D model and animation platform that has a large selection of rigged models and animations that can be downloaded for free.
    - **What is it used for:** Mixamo is used to supply the base model and core animations for the main player character.Mixamo assets are licensed under Adobe's royalty-free terms, which permit use in both commercial and non-commercial projects.
- **Trello**
    - **What is it:** Trello is a project management tool used for organising, planning, and tracking projects and tasks using agile development.
    - **What is it used for:** Trello has been used to organise the project into tasks and subtasks to help breakdown and distribute the workload amongst the team.

Schedule & Development Plan

Development Plan —

Outline of the production plan from pre-production to delivery.

Milestones	Date	Deliverable	Approval
Pre-Production Phase	28/12/2025	● Start of Game Design Document (GDD) ● Start of Technical Design Document (TDD) ● Start of Production Plan and task breakdown	● Tutor feedback and team sign off
Milestone 1	1/12/2025	● Core player mechanics implemented ● Basic movement and camera works ● Placeholders for ui ● Basic test level created to test functionality of any new added features	● Internal Team Review & Sign-off
Milestone 2	6/12/2025	● Began implementing new widgets such as inventories, battery drain etc ● Added arm/ flashlight animations ● Expanded gameplay	● Internal Team Review & Sign-off

		systems ● Core gameplay loops are playable ● First internal playtest, looking for bugs and more things to add to alpha ● Completed first section of level	
Alpha	8/12/2025	● Prototype #1 ● All core systems implemented ● Known bugs and performance issues are documented ● Polishing GDD ● Adding to TDD ● Extension to current level, adding more mechanics such as traps	● Internal Team Review & Sign-off ● External playtester reviews and comments
Beta	14/12/2025	● Improve build from alpha using feedback from playtesters ● Prototype #2 ● Polished menus, pause menus, options menus ● Most critical bugs resolved, any unattended to are documented to fix in final build ● Temporary audio cues are in place ● Final level build completed ● Focused QA Testing and bug triage ● Performance testing and stability checks	● Internal Team Review & Sign-off ● External playtester reviews and comments
Final	18/12/2025	● Improve build from 2nd playtest during the beta build	● Internal Team Review &

		● Final playable build ● No known critical bugs ● Completed documentation ● Non-essential features de-prioritied to ensure a stable final build	Sign-off
Pitch and Play	19/12/2025	● 2.5 Minutes Code Demo from all team members ● Individual team reports ● Final touches to TDD ● Edit all 2.5M code demos into one video and upload to youtube (unlisted)	● Internal Team Review & Sign-off ● Assessmen t board for upload

Playtesting Plan & Testing

Testing Schedule

Milestone	Testing Activities
Pre-Production	Base unit testing. Item pick up, basic player movement, health system.
Core Systems Development	Expanded unit testing and some integration testing between systems.
Blockout Phase	Basic level design to test game flow and interaction between puzzles, traps and torch mechanics.
Enemy Interaction	Adding enemy AI systems and testing interaction between environment and player.
Pre Alpha	Core mechanics and key systems are in place and testing is about interaction between systems.
Alpha	Base level is built and systems are in-place for an Alpha playtest.
Vertical Slice	All core and non-core systems are in place and

	the overall game feel is being tested and tweaked.
Beta	Improvements and feedback from Alpha test and vertical slice have been added.
Final	Final implementation of mechanics and final level design test for game flow.

Testing Goals

1. Verify core gameplay systems (Camera, Interaction, Doors, Traps, Damage, Inventory) are behaving as intended.
2. Ensure the game is running smoothly on a variety of hardware.
3. Identify and Resolve issues that impact the player experience.

Test Types

- Unit Tests
    - Small scale validation of individual elements or components.
- Integration Tests
    - Test Interactions between systems such as door keys and doors.
- System Tests
    - Test larger gameplay elements and levels.
- Performance Tests
    - Measure frame rates, memory usage and load times.
- User Playtesting
    - Gather feedback from players to iterate upon.

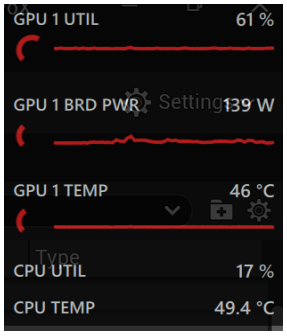
Tests

Test	Test Type	Result/Feedback	Actions
Door and Key Interaction - Testing the key systems works with the doors.	Integration Test	SUCCESS - Correct keys assigned to puzzle doors are unlocking them and not unlocking incorrect doors.	Implement across all puzzle doors.
Pressure Plate/ Trap Activation - Testing if the event dispatchers are working between pressure plate activation and trap firing.	Integration Test	SUCCESS - The correct event dispatchers are being called when a player activates the pressure plate and the crossbow trap is firing the projectile.	Implement across all pressure plate traps.
Health System - Check if damage is being caused from traps and	Systems Test	SUCCESS/FAIL - Traps are causing health damage, and packs	It’s an issue with the ConsumeItem function on the

if the health packs are restoring health.		are restoring health but you can keep restoring your health even after you have used all available packs.	InventoryComp, the system was working previously with batteries when it was handled directly so will try and implement it that way instead. Result: SUCCESS - Extra packs are no longer being consumed.
Battery Reload - Check if decoupled code is working to reload batteries from the items in inventory.	Systems Test	FAIL - Initially this system was working perfectly but after decoupling the reload code and implementing a ConsumeItem function it no longer is working as intended and batteries aren't reloading if the player reaches 0 battery and infinite batteries can be consumed.	Reintegrate the ConsumeItem code directly on BP_Torch again as it was before. Result: SUCCESS.
Torch On/Off - Check if Torch On/Off using input system.	Unit Test	SUCCESS - Using tab is setting visibility correctly to appear like the torch is switching on/off.	Integrate with battery drain so when the torch is off battery drain stops.
Torch Battery Drain - Testing if the battery is draining correctly and that the reload is resetting the battery %. Additionally, turning the Torch On/Off stops/starts battery drain.	Integration Test	SUCCESS - Batteries are draining correctly at 5% p/s and a successful reload is resetting the battery to full. Battery drain stops/starts depending on torches on/off state.	None.
UI Interaction - Check integration between health system and battery drain with the UI to provide player feedback.	Integration Test	FAIL - UI wasn’t bound correctly. SUCCESS - After fixing binding issues UI is updating based on player health and battery amount.	Improve graphics for UI.

Alpha Test Build	User Playtest	Reported Feedback: “The game feels scary but the controls are confusing and some of the keybinds are too close together and I kept hitting the wrong keys. I’d prefer using space to reload instead of R and the battery drain felt too fast”	Look into changing keybinds from individual keys to a simple one key to “use” system that works with a hot-bar style inventory. Decrease the battery drain rate from 5 p/s to 2.5 p/s.
Alpha Test Build	User Playtest	Reported Feedback: “I couldn’t get rock throw to work, and the current enemies felt like they came up on you too fast.”	Known bug: ¼ Rocks cannot be picked up without multiple attempts of pick up. Look into enemy balance.
Rock Pick-up	Unit Test	SUCCESS/FAIL - Rock Pick-up successful but at some angles takes one too many attempts.	Increased sphere collision for pick-up significantly. With larger sphere colliders still some angles pick up is difficult.
Rock Throw	Unit Test	SUCCESS/FAIL - Majority of Rocks throw / physics work correctly. Some clip under the floor after being thrown.	Increased Floor Thickness improves success rate significantly.
Rock throw Play Audio at Location	Unit Test	SUCCESS -	
AI Sight	Unit Test	SUCCESS - AI characters would spot the player.	None
AI Swan Quack	Unit Test	SUCCESS - When a player is seen the npc quack sound is played at location.	None

AI Hearing with Rock Sound Hit + Swan Quack	Integration Test	SUCCESS - AI characters would hear the sound, store the location and investigate.	None
AI Sight With multiple AI NPC's	Integration Test	FAIL - Enemy NPC's would spot each other and continue as if the Player character was not in sight.	Resolved by implementing more player checks.



Performance test

Using AMD Software: Adrenalin I was able to run a basic performance test to see how much CPU and GPU power the game was using at a capped 60 FPS. The GPU power was a 61% utility which means that the graphics of the game was quite high which is typically standard with creating games using unreal engine but board power was only at 139 Watts average. The GPU I used to test the game was a Radeon 7900XTX which has a typical board pwr of 355 Watts which suggests that all of the GPU Processors weren't running as fast as they could've been. The low temperature of the graphics card also heavily implies this.

While the GPU utilisation was quite high the CPU utilisation was only 17% this is likely due to the FPS limit is set to 60 before running the test as the majority of monitors only support up to 60hz and performance increase is exponentially less after reaching 60 fps on a 60hz monitor. The CPU Temp was also low like the GPU meaning the game has a very low reliance on CPU which makes a lot of sense as there aren't many background tasks going on within the game other than the default unreal engine game logic and the NPC AI.

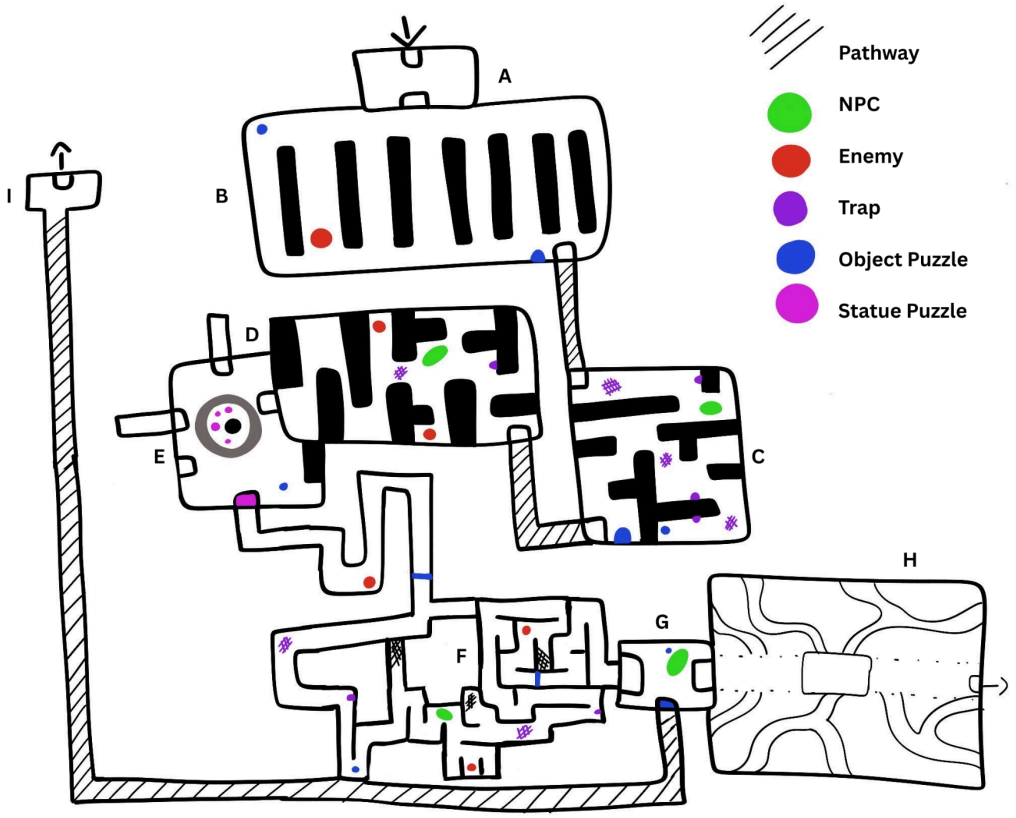
Levels

Level	Item Type	Item	Amount
Opening Area	Key Player Item	Torch	1
	Consumable	Health Pack	1
	Consumable	Batteries	6
	Prop	Clay Pot	4
	Prop	Pile of Crates	1
	Prop	Lamps	3
	Light	Point Light	3

	Prop	Debris	1
Level 1	Enemy	Soldier Enemy	1
	Consumable	Health Pack	1
	Consumable	Batteries	4
	Prop	Terracotta Soldiers	314
	Key Item	Key	1
	Puzzle Door	Door	1
	Prop	Burnt Bricks	2
	Prop	Pots	4
Level 2	Key Item	Key	1
	Puzzle Door	Door	1
	Consumable	Health Pack	1
	Consumable	Batteries	4
	NPC	Dr. Wen	1
	Trap	Pressure Plate	2
	Trap	Crossbow Trap	2
	Prop	Pile of Crates	1
	Prop	Debris	1
	Prop	Burnt Debris	2
	Prop	Terracotta Soldiers	45
	Prop	Stone Rubble Pile	1
Level 3	Prop	Kneeling Archers	8
	Prop	Terracotta Soldiers	45
	Prop	Kneeling Archers	8
	Prop	Stone Rubble Pile	1
	Prop	Kneeling Archers	8
	Prop	Kneeling Archers	8
	NPC	Okoro	1

	Prop	Debris	1
	Prop	Burnt Debris	2
	Consumable	Health Pack	1
	Consumable	Batteries	4
	Enemy	Swan	1
	Enemy	Crane	1
	Key Item	Key	1
	Puzzle Door	Door	1
Level 4	Prop	Kneeling Archers	8
	Consumable	Health Pack	1
	Consumable	Batteries	3
	Trap	Pressure Plate	1
	Trap	Crossbow Trap	1
	Enemy	Soldier	2
	Prop	Debris	1
	Prop	Burnt Debris	2
	Prop	Terracotta Soldiers	30
Level 5	Prop	Terracotta Soldiers	45
	Consumable	Health Pack	1
	Consumable	Batteries	4
	Prop	Clay Pot	4
	Prop	Pile of Crates	1
	Puzzle	Statue	4
	Prop	Debris	1
	Prop	Burnt Debris	2
Level 6	Consumable	Health Pack	1
	Consumable	Batteries	4
	Prop	Debris	5

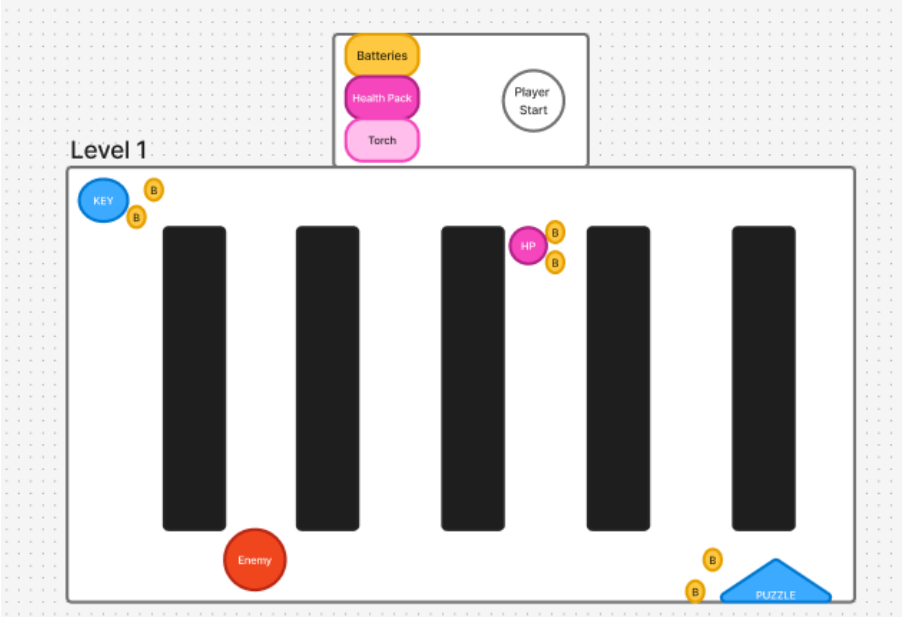
	Prop	Burnt Debris	2
	Enemy	Swan	2
	Enemy	Crane	2
	Key Item	Key	2
	Puzzle Door	Door	2
	Trap	Pressure Plate	2
	Trap	Crossbow Trap	2
Level 7	Consumable	Health Pack	1
	Consumable	Batteries	6
	Prop	Kneeling Archers	8
	Prop	Tomb Door	1
	Key Item	Key	1
	Puzzle Door	Door	1
Level 8	Prop	Debris	10
	Prop	Burnt Debris	20
	Prop	Clay Pot	8



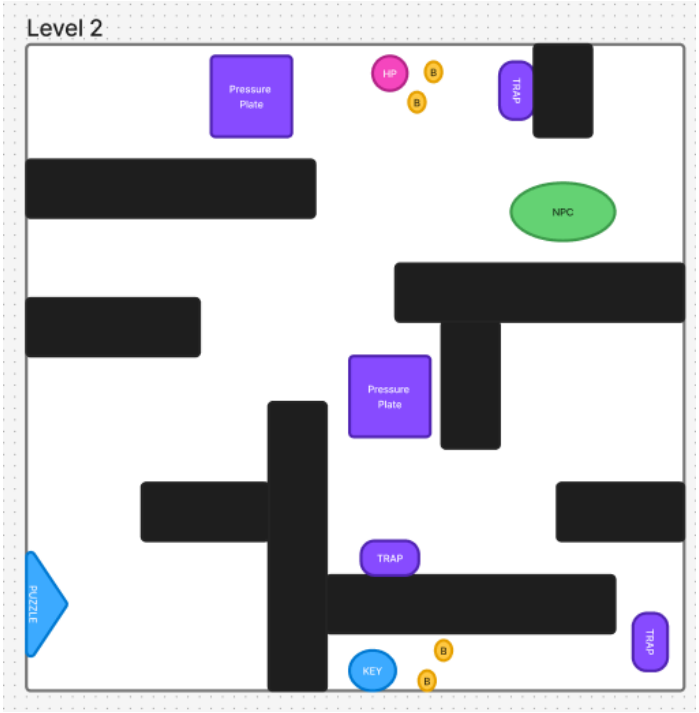
Wox World Map (Lara Lowndes, Author, 2025)

Level Diagrams

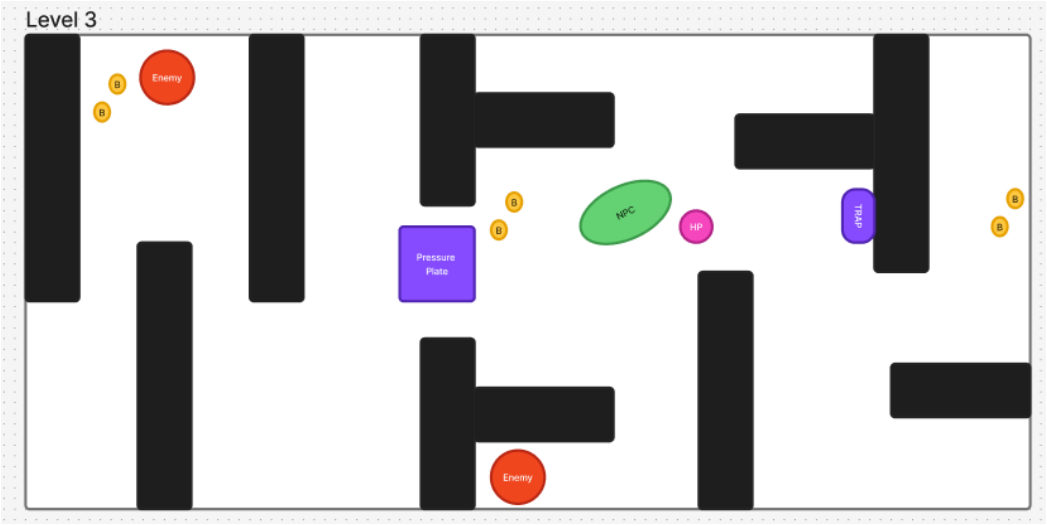
These layout diagrams illustrate the first 3 levels of Whispers of Xi'an, they demonstrate the placement of consumables, puzzles, traps, npcs and enemies within a given level.



(Figma, n.d)



(Figma, n.d)



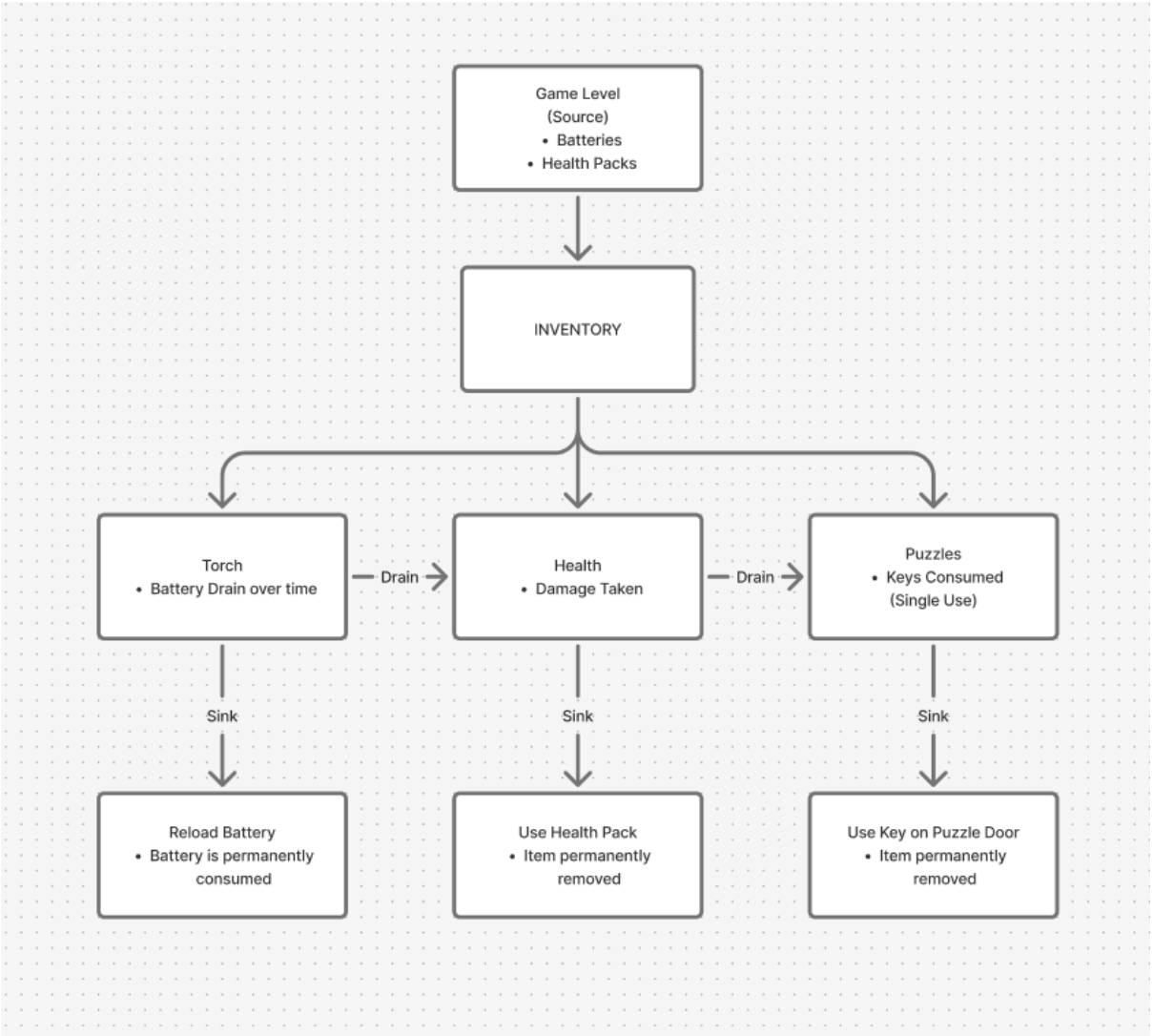
(Figma, n.d)

Internal Economy

Whispers of Xi'an uses a lightweight internal economy built around the management of consumable items, primarily batteries for the torch and health packs for player survival. These resources are intentionally limited within the game world, requiring players to make strategic decisions about when to consume or save them.

Battery drain on the torch is continuous and time-based, meaning that constant use directly reduces the player's visibility and increases tension. When battery levels fall below 5%, the torch begins to flicker, providing a warning that encourages players to look for new batteries, reload if they have the available or conserve remaining power until more batteries are found. Health packs operate similarly, offering recovery from trap, environmental or enemy damage.

Sources: Are places around the game world where players can come across both Health Packs and Batteries.
Drains: The torch battery system functions as a continuous drain as battery power constantly decreases as the torch is used, this creates a natural pressure for players to conserve batteries or to risk searching for more. Health is an additional drain as it decreases in response to damage events.
Sinks: Sink are permanent removal points from the economy where resources are not reintroduced into the loop. Reloading the torch acts as a sink as once that battery is consumed it is not returned to the world environment, similarly health packs work in the same way.



(Figma, n.d)

User Interface

UX / UI Design

The user interface has been designed to be clear, minimal, and supportive of gameplay, ensuring that players can easily access important information without breaking immersion. UI elements are displayed only when necessary to reduce on-screen clutter and maintain player focus.

The HUD presents essential gameplay information such as player status, interaction prompts, and system feedback in a consistent and readable format. Information hierarchy is prioritised so that critical elements are immediately visible, while secondary information is less visually dominant.

Menus follow a consistent layout and navigation structure across the game, allowing players to intuitively move between screens such as the main menu, pause menu and options menus. High-contrast text and icons are used to ensure readability across different lighting conditions.

From a UX perspective, the interface prioritises ease of use and accessibility. UI interactions provide clear visual feedback and confirm player input, reducing confusion and improving usability. The overall design aims to support both new and experienced players by minimising cognitive load during games.

UI Elements

The following ui elements are included within the game;

- **Main menu -**allows the player to start the game, access options and exit the game.
- **Hud (heads up display) -** displays essential gameplay information such as player status and system indicators during gameplay
- **interaction/ pickup prompts -** context-sensitive prompts that appear when the player can interact with objects or environments
- **Pause menu -** enables the player to pause the game, resume gameplay or access the settings
- **Options menu -** allows player to adjust settings such as graphics, resolution scale and v-sync

Each ui element is designed to be visually consistent and easily recognisable to ensure smooth navigation and player understanding.

Wireframes / Mockups

Wireframes and mockups were created to plan the layout and structure of the user interaction before full implementation. These visual references were used to determine the positioning, scale and hierarchy of ui elements.

Figure 1 (Main Menu)

Main Menu Mockup (Lara Lowndes, Author, 2025)

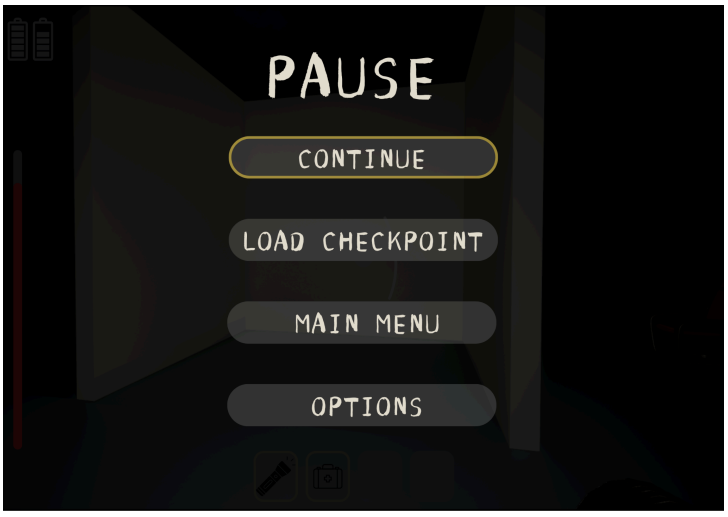
Displays the layout of the main-menu. This includes navigation buttons and how it will visually look.

Figure 2 (Hud Mockup)

Hud Mockup (Lara Lowndes, Author, 2025)

Placement of gameplay information on screen to avoid obstructing player visibility while remaining easily accessible

Figure 3 (Pause Menu Mockup)



Pause Menu Mockup (Lara Lowndes, Author, 2025)

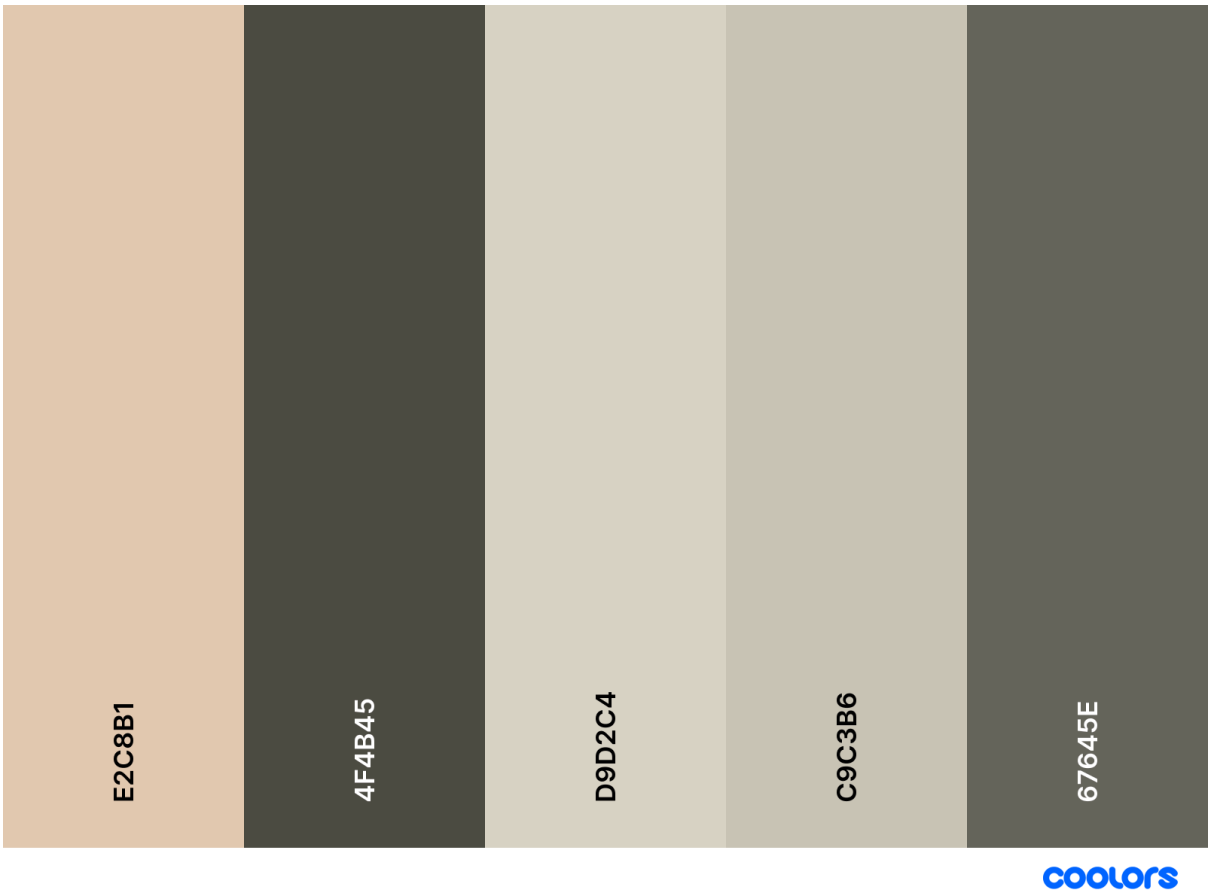
Shows the structure of the pause menu and how players will navigate between gameplay and settings and main menu.

Figure 4 (Death Screen Mockup)

Death Screen Mockup (Lara Lowndes, Author, 2025)

This screen informs the player of failure and provides clear options to reload from the last checkpoint or return to the main menu. The layout is kept minimal to ensure readability and allow the player to quickly continue or exit the game.

Colour Scheme



(Coolors, 2025)

The UI and menu colour scheme was chosen to evoke the look and feel of a sandy, journal-like palette, aligning with the game's archaeological theme. There is a focus on naturalistic tones, such as

muted browns, warm neutrals, and soft stone colours. These were chosen to replicate the aesthetic of an archaeologist's leather-bound field journal.

Required Assets

Asset	Type	Author	Link/Source
E Icon	PNG	pinterest	
Page Turning Sound	Audio	freesound.org	
Holiday_Homework	Font	dafont	
Roboto	Font	Default font in UE	
crosshair_ui	texture	Our Team	
Terracotta soilder	texture	Google search	
Manuscript background	texture	Pinterest	
Paper rustle	sound		

Asset List

Asset Specifications

File Type	Format
3D Model	FBX
Textures	PNG (max 4096×4096)
Audio	MP3 or WAV
UI	PNG
Font	TTF

WoX makes use of third-party assets sourced from the Unreal FAB Store/Unreal Marketplace as well as Adobe Mixamo. Assets obtained through the FAB/Marketplace are provided under Epic's standard Unreal Engine End User License Agreement (EULA), which permits developers to use, modify, and distribute these assets within any Unreal Engine project, including commercial releases, as long as the assets are not redistributed in a standalone or extractable form. Mixamo animations are licensed under Adobe's royalty-free animation use license, which grants the right to download, modify, and use the animations in both non-commercial and commercial projects. These stores were chosen as they both provide high quality assets free to use with an appropriate licence.

Asset List

Asset	Type	Author	Link/Source
Starter Content	Asset Pack	Unreal Engine	In-Engine
Third Person Starter Pack	Asset Pack	Unreal Engine	In-Engine
First Person Starter Pack	Asset Pack	Unreal Engine	In-Engine
Megan	3D Model	Mixamo/Adobe	https://www.mixamo.com/#/?page=1&query=megan&type=Character
Action Adventure Pack	Animations	Mixamo/Adobe	https://www.mixamo.com/#/?page=1&query=&type=Motion%2CMotionPack
Battery Low Poly	3D Asset - FBX	Ele.G (FAB)	https://www.fab.com/listings/2a0b6f92-18cc-4347-9cd9-79341201fc76
Simple First-Person Flashlight Animations V1	Asset Pack	Treety (FAB)	https://www.fab.com/listings/1b3c04f5-3a87-44a0-9f72-d27a56c02646
Pile Of Crates Basic	3D Model	Rosbergen Designs	https://fab.com/s/bfcff7c6c345
Terracotta Warrior	3D Model	Pixol3d	https://fab.com/s/7fdeca820ab7
Construction Lights	3D Model Pack	Salt Lake Manor (FAB)	https://fab.com/s/8c195b462104
Ancient Clay Vase ceramic antique pottery Realistic Game Ready 3D model	3D Model Pack	Gvozdy (FAB)	https://fab.com/s/18dbc92b96b1
Pressure Plate	MP3 File	proolsen (Freesound)	https://pixabay.com/sound-effects/pressure-plate-45634/
Stone Rubble Pile Med Quality	3D Model	Quixel Megascans	https://www.fab.com/listings/046f4f4f-ffa2-4d63-ae22-4524ce10e529

Burnt Debris Pack	3D Model Pack	Quixel Megascans	https://fab.com/s/ec2594815d04
Burnt Bricks Pack	3D Model Pack	Quixel Megascans	https://fab.com/s/bb2dff0a7cdb
Old Clay Pitcher	3D Model	Quixel Megascans	https://fab.com/s/f091ed50eec2
Chinese Kneeling Archer Qin Dynasty Raw Scan	3D Model	JordanFry3D	https://fab.com/s/f04993d3eee2
CC0 - Wooden Arrow	3D Model	Plaggy	https://fab.com/s/aebee62d16a1
Recurve Bow	3D Model	The Van	https://fab.com/s/7d5d89692f47
Rock_Hit	MP3 File	Pixabay	https://pixabay.com/sound-effects/kicking-a-stone-74151/

Packed Level Actors

Whispers of Xi'an utilises packed level actors to consolidate large non-interactive environmental structures such as the walls and floors of the game level. This system allows for a non-destructive workflow where multiple static meshes can be combined into a single asset. Using packed level actors means we can reuse the same large sections of floor more easily and much quicker (Unreal Engine, 2025) without having to manually place or duplicate each floor tile, this also helps maintain a clean and organised outliner as packed level actors are one object instead of several.

Game Mechanics

Player Movement & Animations

Player Movement - Core

Player movement functionality is a core mechanic and is implemented using a C++ base class, from which the character Blueprint is created, ensuring the best performance and control. The movement system is based on the design principles and implementation presented in Stephen Ulibarri's *Unreal Engine 5 C++ The Ultimate Game Developer Course (Ulibarri, n.d.)*, which informed the base structure and behavior of the movement mechanics. A basic stealth system was implemented where if the player is crouching the enemy detection threshold is lessened enabling the player to move through the environment more easily. Additionally, when in a crouched position the player can safely traverse pressure plate traps without triggering them.

Base Walk Speed: 600 units per second (Unreal Default)
Crouch Eye Height: 50.0 cm

Crouched Half Height: 50.0 cm
Max Walk Speed While Crouched 200.0 cm/s

Player Camera - Core

The game uses a first-person style camera, in order to achieve this the player camera is parented directly to the player character's head socket on the character skeleton, with a local positional offset of (x = 2.8, y = −4.0, z = 18.0) units. This offset slightly adjusts the base viewpoint to reduce visible clipping with the character mesh.

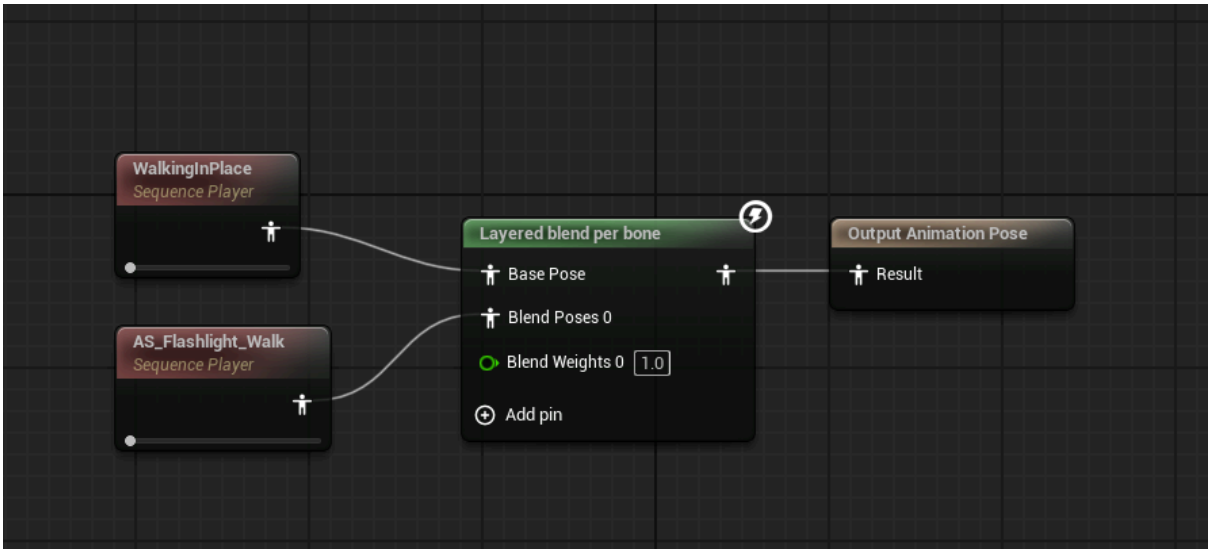
The camera field of view (FOV) is set to 90°, with a first-person scale of 0.6 applied to fine-tune the perceived distance between the camera and nearby objects. Helping create a more natural sense of depth and player presence in confined spaces.

To help reduce clipping of both the player mesh and geometry camera angles, rotational constraints are enforced. Yaw is clamped to a range of −90° to +90°, limiting horizontal rotation relative to the player. Pitch is similarly restricted to −60° to +60°, preventing the player from looking excessively up or down, which could otherwise lead to geometry clipping or disorienting viewpoints.

Additionally a body follow speed has been added and is set to 5.0f, controlling how quickly the player's body rotates to align with the camera's facing direction. This creates a smooth, lagged follow behaviour rather than an immediate snap. All key camera variables (including offsets, rotational limits, and body follow speed) are exposed to Blueprints via BlueprintReadWrite from the existing C++ implementation. This design choice allows rapid iteration, tuning, and playtesting of camera behaviour without requiring constant recompilation.

Player Animations - Non Core

Two separate animation asset packages were required for the implementation of player animations in Whispers of Xi'an. These consisted of the Action Adventure Pack sourced from Mixamo and the Simple First-Person Flashlight Animations V1 package obtained from the FAB store, developed by the user Treety. The use of both packages was required due to the design choice of using a full-body first-person perspective, in which the complete player character model is used rather than only the arms. As the Simple First-Person Flashlight Animations V1 package provides animations limited to the arms, these animations were retargeted onto the base full-body character model within Unreal Engine to ensure compatibility with the chosen perspective. Then, an animation bone blend was applied within the animation blueprint to integrate the lower-body and torso animations from the Action Adventure Pack with the arm and flashlight animations from the Simple First-Person package, producing a full-body animation system.



Unreal Engine 5.5 Wox Project screenshot (Lara Lowndes, 2025)

Animations downloaded from Mixamo used for any traversing animations required the ‘In Place’ checkbox to be ticked to ensure clean animations when imported into the project as ‘Loop Animation’ is used where needed.

Item Pick-Ups, Interfaces & Interaction

Item Pick-Ups & Interfaces - Core

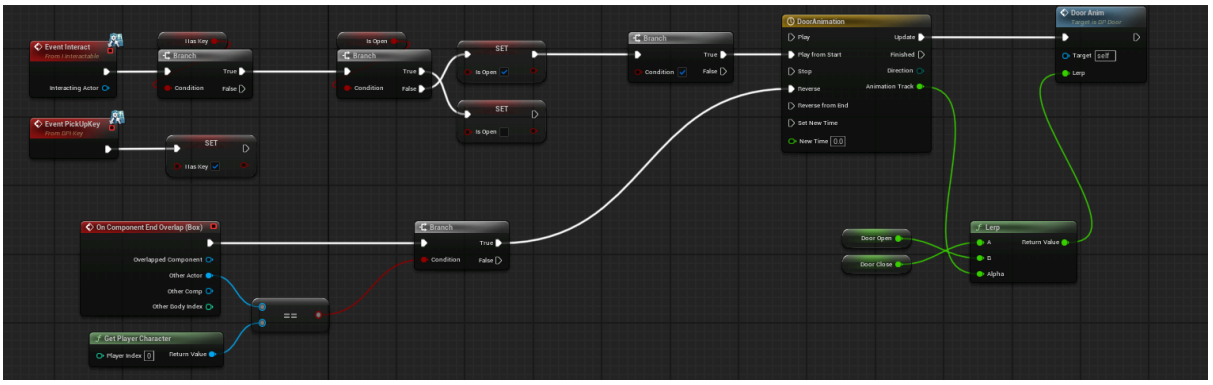
WoX implements an interface-driven interaction system designed for modularity and ease of use. All interactable actors implement a custom I_Interactable interface, which defines the interaction behaviour and allows for re-use of code across multiple object types for example, doors, items, pick-ups. Interaction is initiated through a forward-facing line trace originating from the player camera, with a base interaction distance of 250.0f. This range was used to help with accuracy and ease of use, allowing the player to interact with nearby objects without requiring overly precise positioning. Interaction events only register when the target actor has collision enabled and successfully implements the I_Interactable interface, preventing unintended interactions.

The player triggers interactions through the dedicated ‘E’ key, which is set in the Input Mapping Context. When pressed, the trace checks for a valid interactable target and, if found, enables interaction on that object whether that is picking up an item or opening a door. This approach decouples player input from object-specific logic, ensuring that new interactable objects can be introduced without modifying the core player code.

Door Puzzles - Core

The puzzle door system builds on the base interaction system by introducing conditional access based on collectible key items. Puzzle doors require the player to obtain a specific BP_Key object placed within the level. Each key implements a dedicated BPI_Key interface, allowing it to communicate cleanly with its matching door that can be assigned in the details panel.

To ensure smooth door motion, a LERP-based interpolation was used to adjust the door's relative location over time, creating a fluid movement that improves gameplay feel. Once the player has passed through, a collision mesh triggers the door to close again. A similar system has been implemented for traps linking specific pressure plates with each trap.



Unreal Engine 5.5 Wox Project screenshot (Lara Lowndes, 2025)

Traps - Non-Core

The pressure plate trap system uses a similar system to the Door Puzzles but relies primarily on collision-based triggers and event dispatching. Pressure plates use a simple BeginOverlap / EndOverlap workflow to detect when the player steps onto or off the plate. Because the physical movement of the plate is minimal, an audio cue is used to reinforce player feedback and convey successful activation. Each pressure plate is linked to a corresponding trap actor through event dispatchers, broadcasting PlateActivated and PlateDeactivated events. When the plate is triggered, the trap responds by spawning a projectile and initiating its firing behaviour which has a delay of 1 seconds.

The projectile itself is implemented as an BP_Arrow, a blueprint class with a ProjectileMovementComponent. This component is set up with an initial speed of 2500 units, a maximum speed of 2500 units, and a gravity scale of 1.0, giving the projectile a fast, linear trajectory. Collision is handled through a box collider attached to the projectile and OnHit the projectile performs a type cast to the player character which calls the damage health function to apply damage to the player.

Acid or 'Pools of Mercury' traps work using overlap events but instead of calling event dispatchers they work by directly interacting with the player health system causing a 'poison' effect where player health ticks down by 10 over 3 seconds using delta seconds until the player leaves the collision zone.

Health System - Core

The health and damage framework is built using Unreal Engine's base damage system, enabling a modular approach to applying damage across a wide range of gameplay actors. The player character uses the Event AnyDamage, which is called whenever a valid damage event occurs. A simple health validation routine checks whether the player is still alive before subtracting the incoming damage. As

it is using Unreal Engine's built in Damage function any actor within the game world can use this by calling this function helping create an easy to use modular system.

Torch - Core

The torch is a core mechanic and has several systems needed for full gameplay functionality. The torch can be easily toggled on and off using SetVisibility on the light component and is attached to the player via AttachComponentToComponent, using a dedicated socket at the right hand added to the player's skeletal mesh. While the visible torch mesh is attached directly to the hand, the actual light component is parented to a torch pivot point positioned slightly offset near the hand in the position where the light from the torch would emanate. Separating the torch light component from the mesh improves feel and visual feedback of the torch and helps maintain positioning when transitioning between different movement states such as walking and crouching.

To aid in visual player feedback a flicker effect is applied to the Light Function Material when battery level reaches below or equal to 5% giving a visual warning to the player. This works by creating a new material and modifying the Emissive Colour value, this material takes time scaled by 0.5 and passes it through a sine and a frac function. This creates a variation in intensity of the emissive colour effectively simulating a flickering light source.

The torch is powered by a battery system that continuously drains over time, each tick of game time the current battery is drained by 5% (Battery Drain Amount * DeltaTime) and is clamped between the max battery life (100) and the minimum (0). When the battery reaches 0, the light component's visibility is automatically turned off simulating the torch turning off. Battery drain is integrated with the torch on/off state so that battery drain is stopped when the torch is in its off state.

The battery reload system is directly integrated with the inventory and item pick up system where when the player attempts to reload their batteries if there are some available in the inventory the player can directly reload and reset the torch battery life. *See Inventory Interaction.*

Inventory Interaction - Core

Consumable items can be collected in the game world such as Batteries and Health packs are stored within the player's inventory, allowing for controlled resource distribution and the creation of a simple internal economy. Consumable items are identified using a Name variable that must correspond to the specific BP_Item blueprint placed in the environment. When an item is used, the relevant gameplay actor, for example the torch when reloading batteries, checks the inventory to see if the item type is available and checks the current quantity. If enough items are in the inventory (More than 0), the item is consumed t and the inventory component decreases its stored count. This system enables players to directly activate consumables through input actions, for example Batteries can be used to reload the torch via the 'R' key defined in the Input Mapping Context.

Matthew -

Rocks - Core

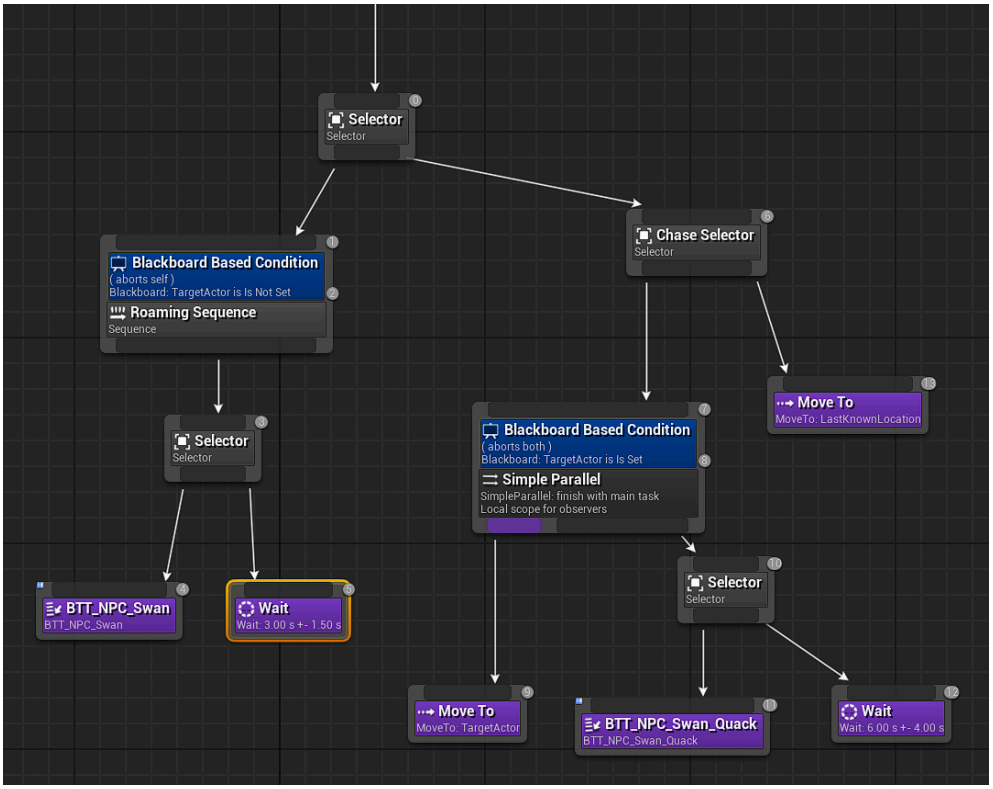
Rocks are an item found scattered around the game world and are key items to help lure the crane enemy. The items are collected by the player and can be kept for vital moments within the game using them as a scarce resource gives the player an additional layer of risk assessment determining whether it would be useful to use their rocks now or keep for a more troublesome task. The player is able to pick up as many rocks as they can find, which also gives the player the opportunity to explore and discover more areas of the map to go find more rocks. The player will pick up these rocks with the input actions E - interact key and then throw the rock with the left mouse button / click. The rocks are a key gameplay element when you come across the bronze crane and are vital to surviving and traversing across the home pit of the crane.

Enemy (Swan / Crane) NPC - Core

Swan

The goal of the bronze swan enemy is to wander randomly until finding the player, and then upon locating the player attempting at keeping up / following the player to the best of its ability. The swans are relatively easy for the player to evade as the swans are slower than the player and they don't harm the player directly. But they are common to come across meaning it is a vital gameplay mechanic for the player to face. When the player is spotted by a swan the swan will occasionally call out when the player is still in vision of the swan. This call will be heard by the bronze crane who will start to make its way over to the last call or audio cue heard within range of the crane.

Behaviour Tree (Swan)

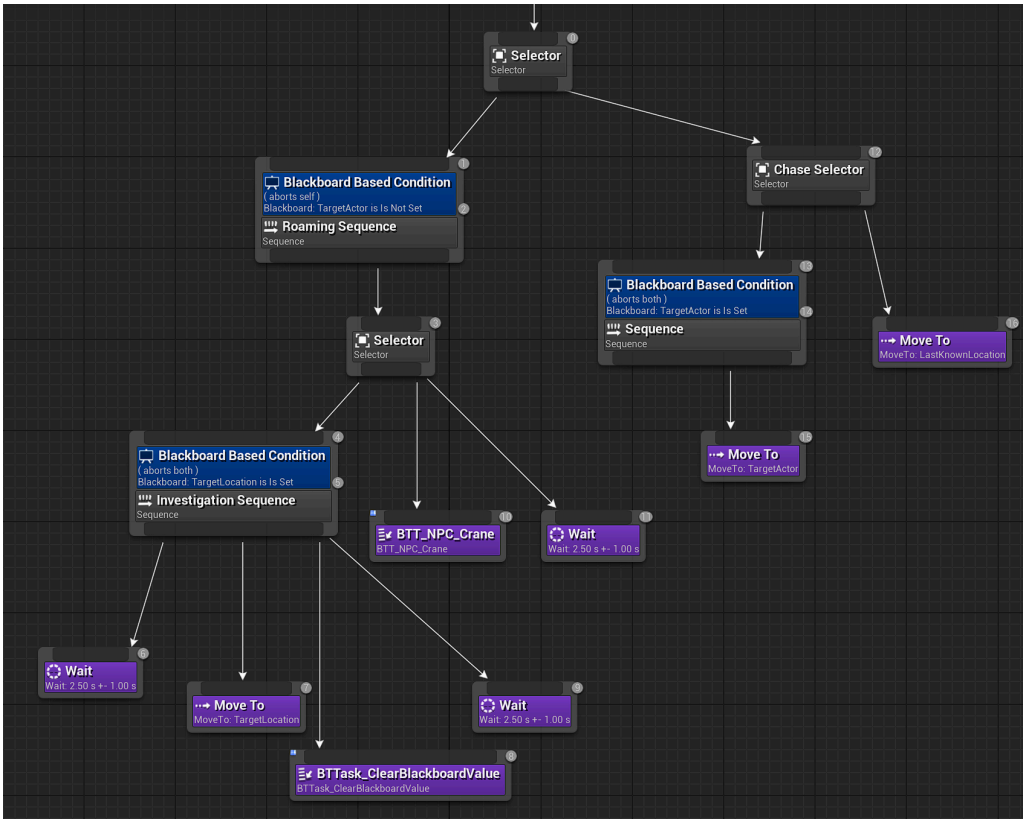


There are a lot of similarities between the behaviour trees of the swan and crane ai but the key difference is the Simple Parallel which allows for the BTT task ‘BTT_NPC_Swan_Quack’ to be played in intervals of 2-10 seconds while (‘Move To’ - Target Actor) is active. As well on that right side is the (‘Move To’ - LastKnownLocation) which is received by checking the last place the actor was located and going there is played until it is stopped once it arrives and TargetActor is removed. When the AI has no input for TargetActor the AI runs the BTT_NPC_SWAN BTT task which is a simple roaming implementation of finding a location anywhere from up to a 2000 distance from the player and a minimum of a 500 distance from the player.

Crane

The bronze crane listens out for any sound made by either the player or the bronze swans which is what makes

Behaviour Tree (Crane)



As i said the behaviour tree’s are similar but

Vilius -

Tom -

Version Control Logs

Commits are to the ‘main’ branch only.

User	Commit	Date

Lara Lowndes	Initial Commit	07/11/25
Lara Lowndes	Added Basic Player Movement and Third Party Animations from UE Marketplace (FAB)	08/11/25
Lara Lowndes	Re-targeted free torch animations onto full body model	11/11/25
Lara Lowndes	Camera Adjustments, Ani Adjustments, Torch Light	12/11/25
Lara Lowndes	Added new player model and rebuilt animations and added crouch.	12/11/25
Lara Lowndes	Small torch tweaks, created Actor class for Torch	13/11/25
Lara Lowndes	Changed Torch to SKM, added a broken pick up system.	14/11/25
Lara Lowndes	Fixed Torch Pick up and broken animations!	14/11/25
Vilius Vaicekauskas	MainMenu+OptionsMenu	02/12/25
Lara Lowndes	Added interactable interface	02/12/25
Matthew Morrisey	Basic Ai System for NPC's	03/12/25
Lara Lowndes	Small Updates to torch	04/12/25
Matthew Morrisey	Added raycast / trail for enemy npc ai	04/12/25
Matthew Morrisey	Fixed Ai	06/12/25
Matthew Morrisey	Started a throw Rock Mechanic - Animation Broke	07/12/25
Matthew Morrisey	Throw Animation Implemented	07/12/25
Matthew Morrisey	Pick Up Rock and Rock Counter to ensure player can only throw rock when available	07/12/25
Matthew Morrisey	Rock Throw Audio and Asset spawning + AI Luring Behavior Tree Added	09/12/25
Lara Lowndes	Door mechanics test	12/12/25
Lara Lowndes	Branch has been fixed!	12/12/25
Matthew Morrisey	Ai Setting change	12/12/25
Vilius Vaicekauskas	Blueprint inventory	12/12/25
Lara Lowndes	add umap to lfs	12/12/25
Lara Lowndes	Added Rock assets	13/12/25

Lara Lowndes	Fixed BP_Rock blueprint	14/12/25
Lara Lowndes	Added greybox lvl. Added Doors with puzzle interaction.	14/12/25
Lara Lowndes	Fixed Look Inversion	16/12/25
Lara Lowndes	Added Pressure Plate and crossbow traps	16/12/25
Vilius Vaicekauskas	Inventory Fix	16/12/25
Lara Lowndes	Added torch toggle on/off	16/12/25
Lara Lowndes	Adjusted Torch Source, Added prompt for players, made it so crouch doesn't trigger pressure plate	16/12/25
Lara Lowndes	Messing about with inventory stuff	16/12/25
Matthew Morrisey	Rock throw fixes while merging to main	16/12/25
Matthew Morrisey	Added rocks to greybox	16/12/25
Matthew Morrisey	Changed game instance	16/12/25
Lara Lowndes	Added Prop Items from Fab, Added Torch Battery Drain and Reload	17/12/25
Lara Lowndes	Mesh Updates to Traps	17/12/25
Lara Lowndes	Fixing missing bits after merge	17/12/25
Lara Lowndes	Trying to fix AI	17/12/12
Lara Lowndes	Building Demo lvl	18/12/12
Matthew Morrisey	fixed rock colliders and Ai	18/12/12
Matthew Morrisey	Specified AI NPC Classes and got BTT_Swan_Quack Working	18/12/12
Lara Lowndes	Started on base level for demo, changed inventory interact to 'e', started working on health system fixes	18/12/12
Lara Lowndes	Changed battery drain rate	18/12/12
Matthew Morrisey	NPC Quack Lure Working with interval	18/12/12
Vilius Vaicekauskas	MainMenu Cursor+player Fix	18/12/12
Lara Lowndes	Health system, UI updates, demo map updates	18/12/12
Vilius Vaicekauskas	All Menus updated + style update	18/12/12

Vilius Vaicekauskas	added interaction/ pickup popups	19/12/12
Tom Hannam	implementation of "weeping angel mechanic" from personal project	19/12/12
Tom Hannam	imported skeletal mesh model and assigned/tested movemoent mechanics	19/12/12
Tom Hannam	implimentation of jumpscare mechanic with tweaks and fixes	19/12/12
Tom Hannam	installed player damage blueprint provided.	19/12/12
Tom Hannam	added a play sound mechanic for jumpscare.	19/12/12
Tom Hannam	created a simple crouch stealth detection mechanic.	19/12/12

Team Member Contributions

Team Member	Contribution
Lara Lowndes	Player Movement
Lara Lowndes	Player Animations
Lara Lowndes	Item Interaction Interface and Implementation
Lara Lowndes	Door Puzzles
Lara Lowndes	Traps
Lara Lowndes	Health Systems
Lara Lowndes	Torch Systems
Lara Lowndes	Inventory Interaction
Lara Lowndes	Greybox Level
Matthew Morrissey	Swan NPC
Matthew Morrissey	Crane NPC
Matthew Morrissey	AI Perception (Sight and Sound) For NPC (Crane, Swan)
Matthew Morrissey	Throw Rock AI Lure

Appendix

References

Ali Elzoheiry (2023) AI perception tutorial [Video]. YouTube. Available at: https://www.youtube.com/watch?v=UIAazOgww_I (Accessed: 15 December 2025).

Anthony2361 (2024) AI Perception: “On Perception Updated” will never trigger. Unreal Engine Forums. Available at: https://forums.unrealengine.com/t/ai-perception-on-perception-updated-will-never-trigger/1704088 (Accessed: 17 December 2025).

Coolors (n.d.) Generated Palette. Available at: https://coolors.co/e2c8b1-4f4b45-d9d2c4-c9c3b6-67645e (Accessed: 25 November 2025).

Epic Games (n.d.) *AI Perception Blueprint API*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/AI/Perception?application_version=5.7 (Accessed: 6 December 2025).

Epic Games (n.d.) *Behavior tree in Unreal Engine: Quick start guide*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-tree-in-unreal-engine---quick-start-guide (Accessed: 6 December 2025).

Epic Games (n.d.) *Coding Standard*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/epic-cplusplus-coding-standard-for-unreal-engine (Accessed: 1 December 2025).

Epic Games (n.d.) *Level Instancing*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/level-instancing-in-unreal-engine (Accessed: 1 December 2025).

Epic Games (n.d.) *Recommended Asset Prefixes*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/recommended-asset-naming-conventions-in-unreal-engine-projects (Accessed: 1 December 2025).

Epic Games (n.d.) *Using a multi-line trace (raycast) by object in Unreal Engine*. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/using-a-multi-line-trace-raycast-by-object-in-unreal-engine?application_version=5.5 (Accessed: 1 December 2025).

Figma (n.d.) *WoX Wireframes and UML*. Available at: https://www.figma.com/board/vyZ1AMkZntBYnnxQrTOnft/WoX-Wireframes-and-UML?node-id=0-1&t=WmcPmpGm4kYzYFew-1 (Accessed: 19 December 2025).

Krensis (2020) *Unable to increment a variable present in another Blueprint in Unreal*. Stack Overflow. Available at:

https://stackoverflow.com/questions/63868260/unable-to-increment-a-variable-present-in-another-blueprint-in-unreal (Accessed: 10 December 2025).

launemax (n.d.) *Kicking a stone* [Sound effect]. Available at: https://pixabay.com/sound-effects/kicking-a-stone-74151/ (Accessed: 9 December 2025).

Lowndes, L.L. (2025) *Death Screen Mockup*, unpublished image.
Lowndes, L.L. (2025) *HUD Mockup*, unpublished image.
Lowndes, L.L. (2025) *Main Menu Mockup*, unpublished image.
Lowndes, L.L. (2025) *Pause Menu Mockup*, unpublished image.
Lowndes, L.L. (2025) *Screenshot of Unreal Engine WOX Project*, unpublished image.
Lowndes, L.L. (2025) *Wox World Map*, unpublished image.

Matt Aspland (2021) *Unreal Engine AI tutorial* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=w0t75DdkazU (Accessed: 8 December 2025).

ProfMoto (2023) *Unreal Engine tutorial playlist* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=xg7SJU-n6U4 (Accessed: 8 December 2025).

philUnreal_1 (2015) *Blueprint: Change the value of a bool variable*. Unreal Engine Forums. Available at: https://forums.unrealengine.com/t/blueprint-change-the-value-of-a-bool-variable/310661 (Accessed: 6 December 2025).

Riley Belleville (2022) *Unreal Engine AI behaviour tutorial* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=neF4dPJraAY (Accessed: 10 December 2025).

Ryan Laley (2022) *Unreal Engine Blueprint tutorial* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=OytuX_swh8M (Accessed: 10 December 2025).

Shaun Codes (2025) *Unreal Engine gameplay tutorial* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=iegalgFX1-Y (Accessed: 5 December 2025).

Ulibarri, S. (n.d.) Unreal Engine 5: The Ultimate Game Developer Course. Udemy. Available at: https://www.udemy.com/course/unreal-engine-5-the-ultimate-game-developer-course/ (Accessed: 8 November 2025).

Vercion Games (2025) *Unreal Engine Blueprint tutorial* [Video]. YouTube. Available at: https://www.youtube.com/watch?v=B4W9lKbGeIc (Accessed: 8 December 2025).
