



Game Design Document

266

07/05/26

Game Technology and Simulation

2SPILL101

Table of contents

PROJECT: Metal Gear Solid VR Missions Reimagined.....	4
High Concept & Scope.....	4
Story.....	5
Core Concept.....	6
Unique Selling Points (USP):.....	6
Design Pillars:.....	6
Pillar 1: (Movement and feel).....	6
Pillar 2: (Visual profile focus).....	6
Pillar 3: (Stealth at its core).....	6
Universal Mechanics.....	7
Player Mechanics.....	7
Movement:.....	7
Stealth & Cover:.....	7
Actions:.....	7
Progression (Keycard System).....	8
Enemy AI & Detection.....	9
Core Philosophy.....	9
Perception Systems.....	9
AI State Flow.....	9
Interactive Objects overview.....	10
Items that can be picked up.....	11
Items that can be interacted with.....	12
UI & HUD.....	13
Radar/Minimap:.....	13
Personal info stats:.....	14
Visual Cues:.....	14
Menus:.....	15
Game Loop & Scene Management:.....	17
Game over state:.....	17
Game Win state:.....	17
How do the levels start.....	17
Technicians overview.....	18
List of programmer responsibilities.....	18
List of systems.....	18

Camera & Control Scheme.....	19
Gunplay & Interaction.....	20
Aiming:.....	20
Shooting:.....	20
Stealth Mechanics:.....	21
Visual:.....	21
Distraction:.....	21
Takedown:.....	21
Asset & Animation Pipeline.....	22
Constraints set by the faculty.....	22
Programmers:.....	22
Artists:.....	22
Scale and Metrics.....	22
Artist responsibilities.....	23
3D assets list.....	24
Player Avatar model.....	25
Enemy model(s).....	26
Level Design Overview.....	27
Visual profile / Game world:.....	27
Level list:.....	27
Level Layout design.....	28
Audio Design and Overview.....	29
Sound effects list.....	29
Music list.....	30
Scope Management:.....	31
Non-Goals:.....	31
Sources:.....	32

PROJECT: Metal Gear Solid VR Missions Reimagined

Based on the original video game Metal Gear Solid VR Missions from 1999 by Konami.

Engine: Unreal Engine 5

Genre: Tactical Stealth / Puzzle, first-person shooter

Target audience: 13+ primarily male audience (no blood, mild violence)

Team: 3 Programmers and 3 Artists

High Concept & Scope

A modern reimagination of Metal Gear Solid: VR Missions.

Set in first person, utilizing UE5 physics and lighting.

The game focuses on "pure" gameplay: navigating levels, solving spatial puzzles, and neutralizing targets within a simulated environment.

Core Loop:

Infiltrate through the elevator.

Execute Objective. (Reach Goal / Eliminate Targets)

Exfiltrate from the elevator.

Constraint:

Logic must be original (Stealth/Detection), and contain no more than 20% outsourced code.

Story

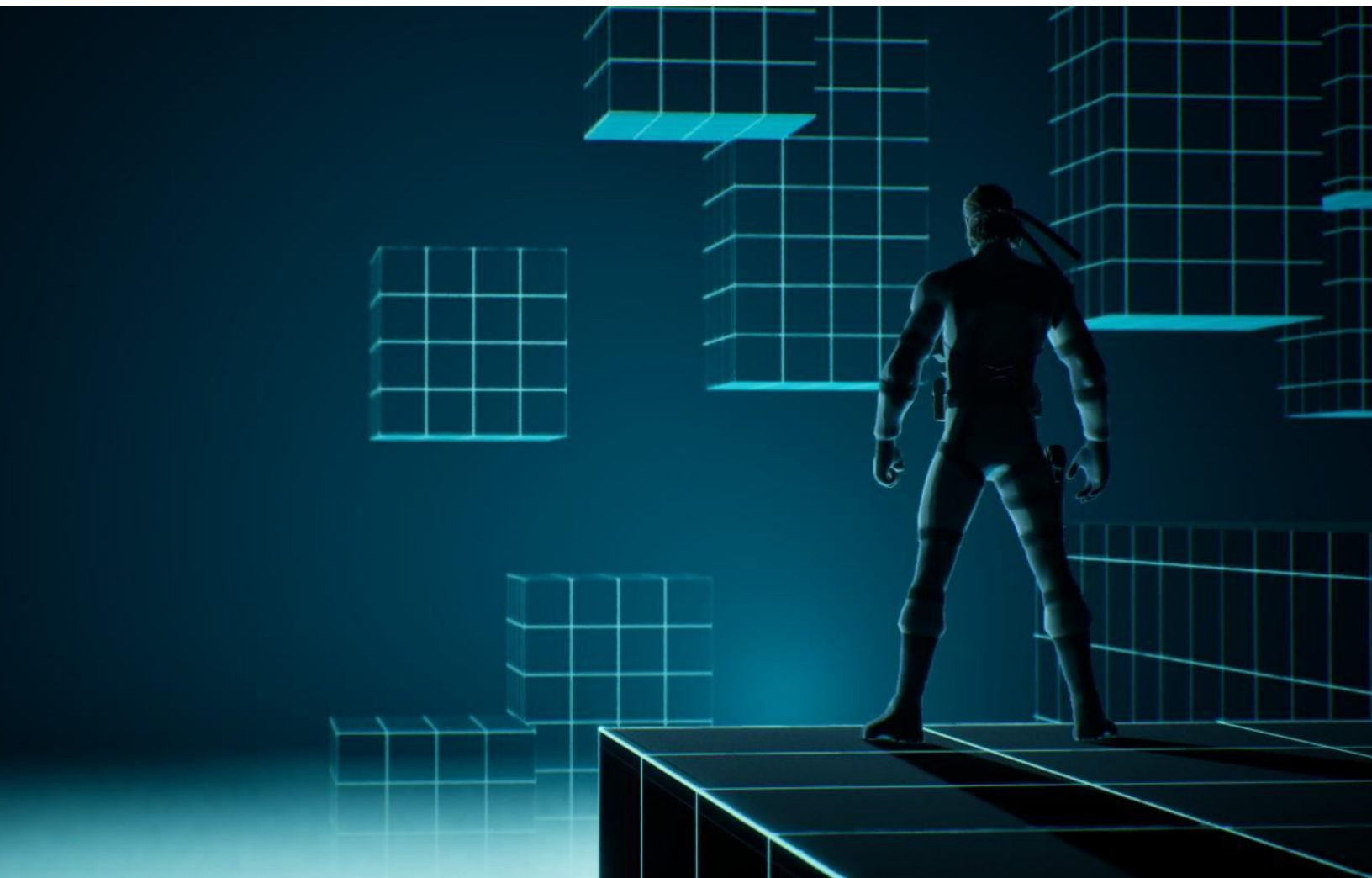
You are the famous Metal Gear Solid Snake, playing training missions in virtual reality (VR) to test your skills.

Each level has its own obstacle you have to go through. Whether it is to sneak past enemies, eliminate them, and steal their keycards, or distract them by throwing “Distraction duckie”.

You have to reach the end without being caught.

The game is not very story-driven; there is no dialogue and no information about the setting. This is intentional to give it an “arcade feel.”

The game will have a few secret hidden easter eggs if you beat all levels.



Core Concept

First-person stealth game utilizing a silenced weapon, a throwing knife, and a “Distraction duckie” used to distract enemies.

Non-full body, just hands for simplicity and less focus on full-body animation.

Gameplay focuses on environmental puzzle-solving, either by avoiding enemy sightlines or by eliminating targets.

Unique Selling Points (USP):

Retro-Futuristic FPS Perspective:

Shifting the classic top-down stealth of MGS to a "snappy" first-person experience, increasing tension through limited vision.

Physics-Based Distractions:

Utilizing Unreal Engine 5's physics to allow players to toss "Distraction Duckies" to manipulate enemy paths manually.

High-Contrast Neon Visuals:

A "Matrix-Blue" aesthetic that provides instant readability for a pure gameplay-focused experience.

Design Pillars:

Pillar 1: (Movement and feel)

Snappy Movement. Controls should feel arcade-like and responsive (The "Doom" standard). This is the opposite of old MGS, as the original game was very clunky with “tank” movement.

Pillar 2: (Visual profile focus)

Visual Readability. Every object's function is clear through color.
(Blue = Safe, Red = Danger, Yellow = Objective).

Pillar 3: (Stealth at its core)

Pure Stealth.

Combat is a fail-state; the goal is to bypass or eliminate targets without triggering an alert.

Universal Mechanics

Player Mechanics

Movement:

- Walk
- Crouch
- Jump (e.g., onto boxes) (This is not currently used for anything)
- Climb ladders

Stealth & Cover:

- Hide in lockers
- Scout safely through one-way windows
- The player can also crouch to activate “X-ray mode”,
 - This allows you to see mission objectives, enemies, and items through walls.

Actions:

- Shoot enemies with a silencer
 - More in-depth interaction description here
- Throw knives
 - More in-depth interaction description here
- Throw “Distraction duckie” to distract enemies
 - More in-depth interaction description here
- Execute stealth takedowns from behind,
- Pick up items (keycards and ammo),
- Interact with environmental objects (doors, scanners, lockers, elevators)

Progression (Keycard System)

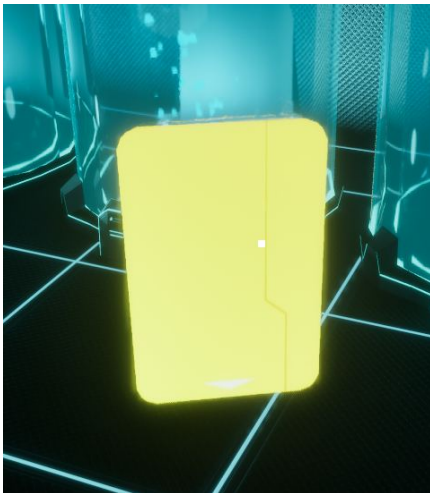
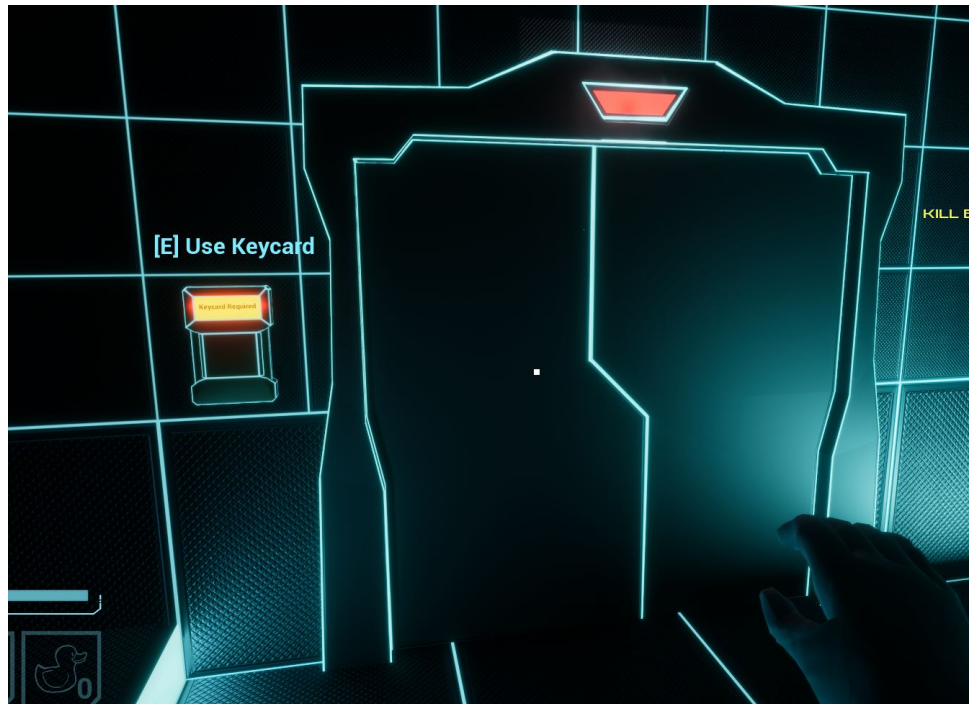
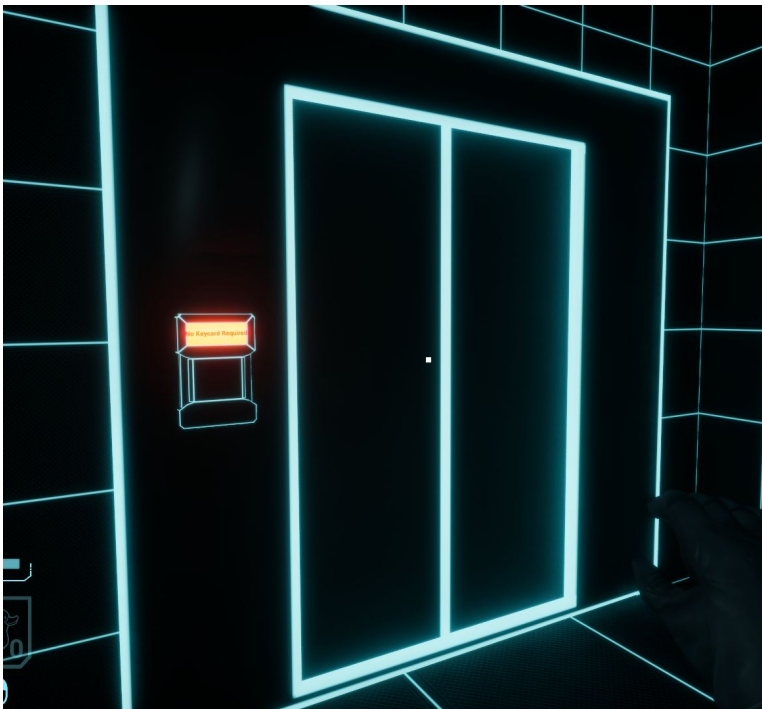
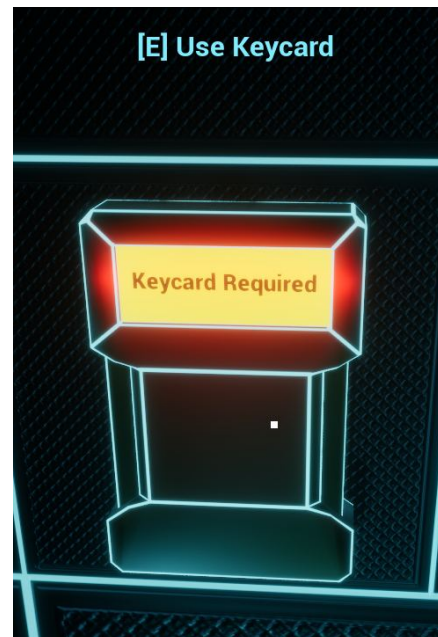
Keycards are brightly colored for high visibility.

Enemies carrying them are marked yellow when scanned in X-ray mode.

Locked doors and their corresponding scanners emit a red light and display a "Please scan keycard" prompt.

Interaction is disabled until the keycard is acquired.

Successfully scanning a keycard turns the scanner and door lights yellow, unlocking the path forward.



Enemy AI & Detection

Core Philosophy

Gameplay revolves around solving spatial puzzles by either avoiding enemy vision cones or strategically eliminating targets.

Perception Systems

- **Vision & Visibility:** Enemies detect the player gradually based on proximity within their vision cones. The game streamlines visibility by tying it to player posture (crouching reduces visibility) rather than calculating dynamic light sources.
- **Audio Detection:** Footsteps and gunshots generate noise events within a defined radius, utilizing Unreal Engine 5's built-in AI Perception System for hearing and noise reporting.

AI State Flow

Enemies transition through four distinct behavioral states driven by the perception systems:

- **Passive (Patrol):**
The default state. Enemies remain at their designated posts or follow a set patrol path.
- **Suspicious (Investigate):**
Triggered by an audio event or the player entering a vision cone. The enemy actively pursues the player's last known location.
This state is communicated to the player via a growing UI detection bar.
If the player breaks line-of-sight and evades, the enemy eventually abandons the search and returns to the Passive state.
- **Alert (Full Detection):**
Triggered if the UI detection bar fills completely.
The screen flashes bright red, and the guard attempts to call for reinforcements.
This grants the player a strict 5-second window to eliminate the enemy before the call goes out.
- **Called In (Game Over):**
Triggered if the 5-second Alert window expires. The guard successfully alerts the base, resulting in an immediate Game Over.

Interactive Objects overview

Interaction System

The game uses an interaction system built primarily in Blueprints, with support for both Blueprint-based and C++-based interactable objects.

Interaction Prompt

Each interactable object includes a collision sphere. When the player enters this sphere:

- A 3D widget component becomes visible
- The widget displays a prompt indicating that the objects can be interacted with

This prompt logic is handled inside the object's Blueprint.

Interaction Execution

When the player presses E, the character performs a sphere trace forward:

- Radius: 20 units
- Distance: 250 units

If the trace hits an actor that implements the Interact interface, the system calls the object's Interact function. Interaction behaviour (opening doors, picking up items, activating objects, etc.) is defined per object.

Blueprint and C++ Compatibility

The Interactable interface exists in both Blueprint and C++. Because of this:

- Blueprint interactables respond to interaction
- C++ interactables also respond to interaction
- Both types work with the same sphere-trace system

Most interactable objects are implemented in Blueprint, but several (such as doors, sliding doors, and vents) are implemented in C++ and function identically within the system.

Designer Workflow

To create a new interactable object:

1. Create a blueprint actor or C++ class
2. Add a collision sphere
3. Add a 3D widget (hidden by default)
4. Implement the Interactable interface
5. Add interaction logic inside the Interact function
6. Place the object in the level

This keeps interactions consistent across all objects and allows both Blueprint and C++ implementations to be used interchangeably.

Items that can be picked up

Throwing knife Ammo

Adds one throwing knife to the player's inventory. Used for silent ranged attacks. Can be picked up again after throwing.

"Distraction Duckie" Ammo

Adds one throwable duck. When thrown, it creates a noise that distracts nearby enemies.

Silenced Pistol Ammo

Adds ammunition for the player's silenced pistol.

Keycards (Various Colors)

Used to unlock specific doors or access points. Each color corresponds to a different access level.

Items that can be interacted with

Doors

Doors can be opened through interaction. The game includes:

- Proximity-based doors, which open when interacted with at close range
- Switch-based doors, which require activating a nearby panel

Door movement is handled through Blueprint or C++ depending on the door type.

Elevators

Elevators function as level transition points. Entering the elevator loads the next level and moves the player to the corresponding starting area of that level.

Security Cameras

Security cameras rotate using a panning motion and display a visible vision cone. If the player enters the vision cone, a detection meter begins to fill. When the meter reaches its maximum value, the player is detected and the game ends.

Hiding Lockers

Lockers allow the player to hide from enemies.

- Interacting with the locker places the player inside
- Player movement is disabled while hiding
- The player can look through the slits with limited camera movement
- Interacting again exits the locker

Ladders

Ladders enable vertical movement.

- Interacting with a ladder switches the player into ladder-climbing mode
- Only vertical movement is allowed while climbing

Camera Control Screens

Interacting with a camera control screen disables all security cameras in the area.

UI & HUD

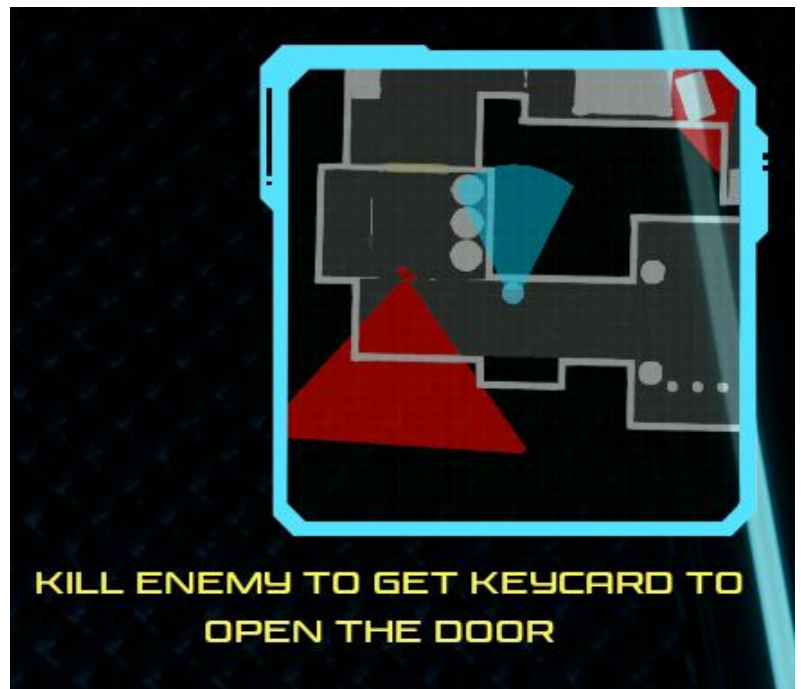
Radar/Minimap:

Displays level geometry

- Walls and Floors in blue
- Uses Scene Capture and blueprint widgets
- The cone is made from static mesh inside the enemy
- Minimap only sees things tagged “minimap”
- All things on minimap without a stencil layer will be colored in as grey objects.
- The standard setup is:
 - Select all floor meshes, Set Minimap and Depth render 5 (Floor), Then select all objectives on the map and add depth 2. If needed tag items with 3 or 4 to get a different color but same effect as the Yellow and red render depths.
 - This ensures the floor gives the minimap a background layer to work off of, and separates all walls and other objects in the minimap, From the floor render depth.
- Stencil Layers
 - 1(Red Enemies)
 - 2(Yellow Items and objectives)
 - 5(Dark Gray Floors)
 - 3+4 (Extra colors for marking and Xray))

Enemy and camera positions

- Enemy with red cones indicating direction
- Player icon with blue cone for indicating direction



Personal info stats:

- Detection bar
- Ammo counter
- Crouch indicator
- Keycard indicator



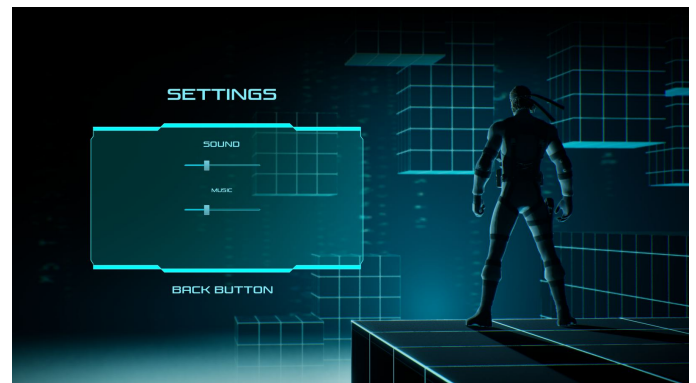
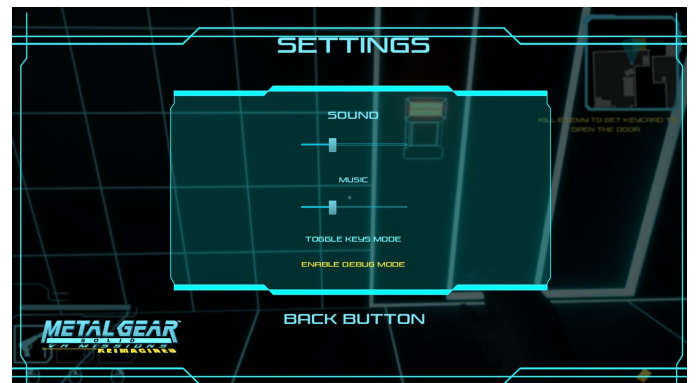
Visual Cues:

- Crouch vignette and visual indicator.
- Camera detection (point light cones).
- Camera shake (for win condition/elevator - illusion of elevator moving to next stage).
- Detection feedback, bar, and eye icon indicating player detection.
- The Crouch indicator helps the player know if they are crouching.
- The Red Screen Tint is a dynamic feedback system for detection. Scales with detection percentage



Menus:

- Start
Press to start
- Options
Lets you change volume
- Level selector
Select what level to play. Unlocks more levels based on previously completed levels.
- Pause
Press P to activate
- Mission failed (lose screen)
Plays when detected or fallen into fog
- Mission complete (win screen)
Plays on win condition





Game Loop & Scene Management:

Game over state:

There is no death; the player can instead get caught, which loses the game.
The player can replay the level.

Game Win state:

The player enters the exit elevator. The doors close, triggering a first-person celebration animation ("YESSS"), followed by a prompt to proceed to the next level.

You cannot bring resources to the next level; all stats, ammo, and such restart per level, and on level reset.

What is the final scene of the game? Funny easter egg!

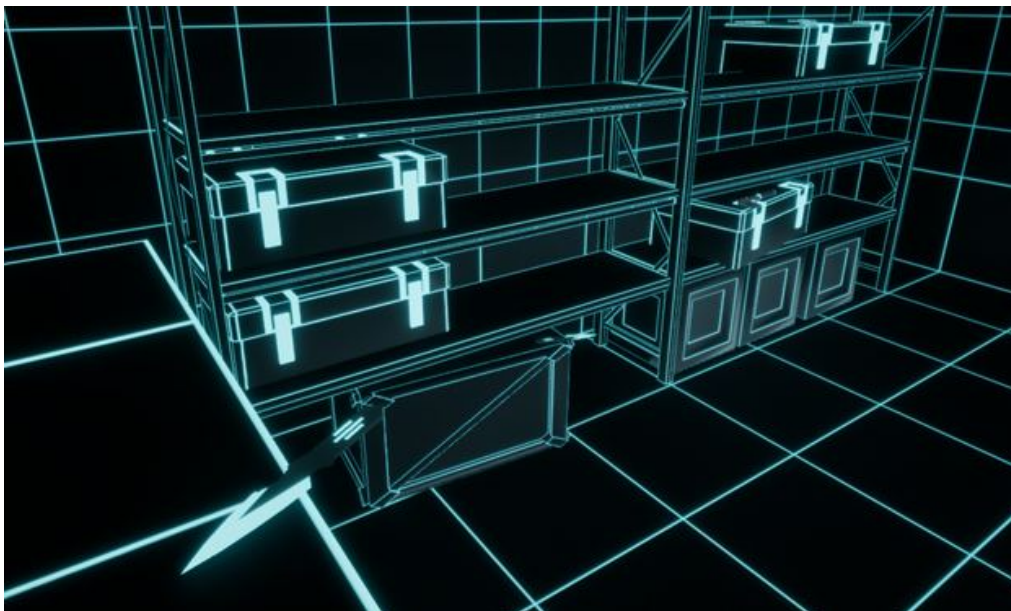
How do the levels start

Start:

The player exits an elevator.

The area outside of the elevator is filled up with ammo that the player can collect.
Either displayed on pedestals or scattered around.

A pop-up with the current objective appears on the side UI in bright yellow (e.g., "Find a way to open the door").



Ammo
scattered
around.

Technicians overview

List of programmer responsibilities

Integration, HUD, Menus, Cleanup, Greyboxing, Interactions and movement AI and Detection, Keycard integration and Door interactions

List of systems

- **Player Movement & State System:**
 - Handles velocity, crouch height adjustment, and camera attachment.
- **AI Perception System:**
 - Drives the vision cones, noise detection, and updates the AI State Machine
- **Interaction Interface System:**
 - A universal blueprint interface (BPI) handling raycasts from the center of the camera to trigger environmental objects.
- **Keycard system:**
 - A lightweight system to track if the player holds a keycard.
- **Minimap system:**
 - Uses a scene capture camera and materials manipulation to showcase a stream of footage from a top down orthographic view, uses depth functions built into unreal for drawing objects of different sizes and heights.
- **PHud:**
 - Controls most UI on the screen during gameplay, including minimap, ammo counter, detection overlays and more.
- **Game Instance / State System:**
 - Tracks current level and triggers the win/loss UI overlays.

Camera & Control Scheme

Perspective: True First Person (Camera attached to Player Capsule).

Body Awareness: Floating arms, no legs, automatic weapon swap system based on last action and ammo. Bobs on movement.

Movement: Standard WASD. High acceleration/deceleration (snappy movement).

Mouse Look: Unrestricted pitch/yaw.

KEYBINDINGS & DEBUG GUIDE		
NORMAL KEYS		DEBUG MODE KEYS
W A S D	Movement	X Damage All Enemies
LMB 	Shoot with Pistol	K Increase Observer Count
RMB 	Throw Duckie	I Decrease Observer Count
MMB 	Throw Knife	U Damage Self / Gameover Test
E	Interact	O Full Ammo / All Keycards
CTRL	Crouch	F Generate Noise
TAB	X-ray	C Spawn Enemy
P ESC	Pause menu	V Exploding Duck

Gunplay & Interaction

Aiming:

The crosshair is permanently centered on the screen.

Shooting:

Raycast originates directly from the Camera center, not the weapon muzzle. This ensures 100% accuracy of the crosshair without complex offsets.

On shot fired the gun rotates to indicate recoil.



Stealth Mechanics:

Visual:

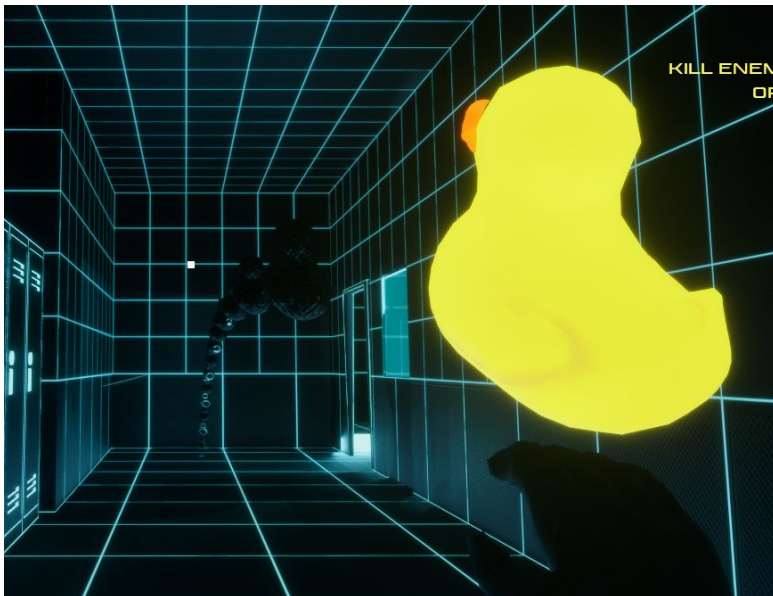
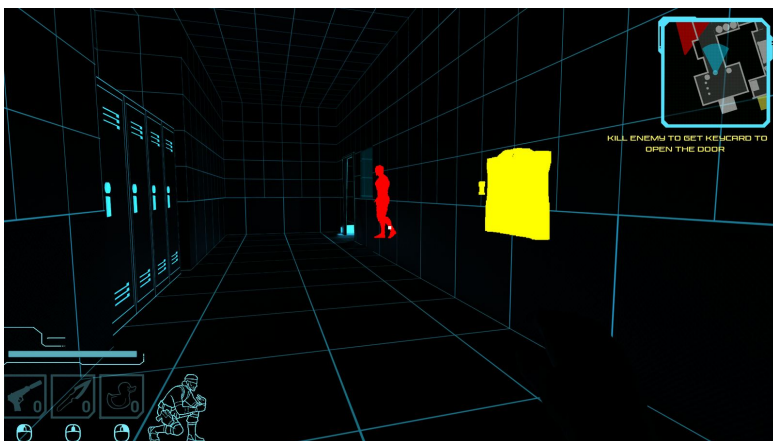
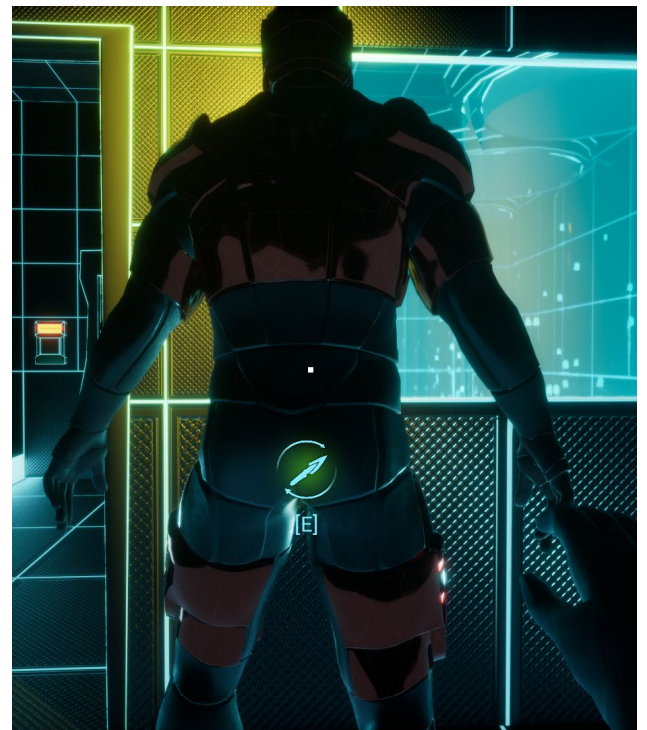
- Uses a "Crouch" toggle that lowers the camera height and reduces noise radius/detection speed.
- TAB to turn on and off X-ray mode.
- In the settings menu you can change between "Hold" or "toggle" of these keys.

Distraction:

- Toss a "Distraction duckie" to lure enemies to a new spot
- Knife, and gunshots also make noise, and can be used as distractions.

Takedown:

- Kills enemies when interacted with from behind. Has to be very close.



Asset & Animation Pipeline

Constraints set by the faculty

Programmers:

Do not wait for final custom animations to build logic. Use the default UE5 Mannequin (Manny/Quinn/FPSBlueprint) and standard Unreal animations to build the interaction and AI state machines.

Artists:

Models, rigs, and animations for the player and enemy characters should be taken from the internet.

Do not waste time making them from scratch. All custom rigs and animations must be mapped using UE5's IK Retargeter to match the standard Unreal Skeleton.

This ensures that, when the final models are ready, they can be simply swapped into the programmer's working blueprints without breaking the code.

All assets must be created by the artists, except for those that require complex rigs.

Scale and Metrics

Use dimensions in the 3D software as a basis. Import the Unreal Player figure into 3D software at the correct scale if unsure.

- Doors are 2m high.
- Crates have a dimension of 1m x 1m x 1m.
- The hallway between enemies is 6m long.

The blue grid texture in the UE project provides a strong basis for asset scale.

Two squares of the floor grid texture = 1m.

Four squares of the wall grid texture = 1m.

Objects that have been too large or too small due to model or import issues have been scaled up or down in the UE5 project to their correct size.

Artist responsibilities

- 3D Assets
- UI Design & 2D elements
- Level layout & building them
 - Level 1
 - Level 2
 - Lobby
- Lighting
- SFX & Music
- Player arms animation
- Enemy Animations
- Materials

3D assets list

All assets should be made as placeholders first, so the programmers have something to work with as early as possible. They must all be fully textured by the end.

- Liquid tank (with bubbles)
- Door
- Doorframe
- Heavy keycard door
- Floor, Ceiling, and Walls
- Half wall
- Plant
- Shelf
- "Distraction duckie"
- Cone for camera
- Table
- Crate
- Shipping container
- Keycard
- keycard scanner
- Elevator
- Locker
- Gun for the enemy
- A silenced pistol for the main character
- Throwing knife
- Ceiling Lamp
- Security camera
- Small crate
- ladder
- podium for a weapon
- Vent

Player Avatar model

“Knock-off Snake”

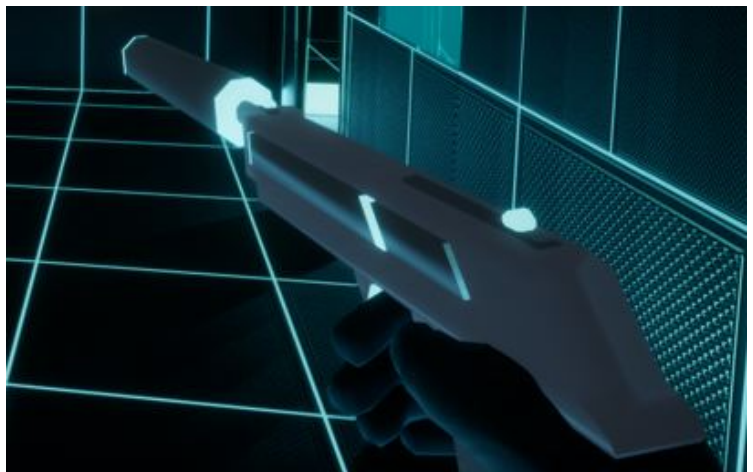
Big boss - only arms will be visible, so we can call him Snake, like the other MGS games.

After many trials and errors working with various models we found online, we settled on this big boss model from MGS: Phantom Pain.

Player animation list (Just arms)

Animations are mainly from Mixamo with some tweaks in Unreal.

- Interact
- Duckie aim
- Duckie toss
- Knife equip
- Knife idle
- Knife walk
- Knife sneak
- Knife throw
- Pistol equip
- Pistol idle
- Pistol walk
- Pistol sneak
- Pistol shoot
- Empty hands idle
- Empty hands walk
- Empty hands sneak
- Knife stab
- Pistol whip
- Head snap
- “YESSS” fist pump



Enemy model(s)

Sci-Fi Soldier 02

The enemy model was found on the Unreal Engine marketplace called Fab.

It has 5 different skin variations, but

The red one works best as it adds to the red = danger.

Enemy animation list:

Animations are mainly from Mixamo with some tweaks in Unreal.

- Idle
- Walk
- Punch
- Ragdolls when dying



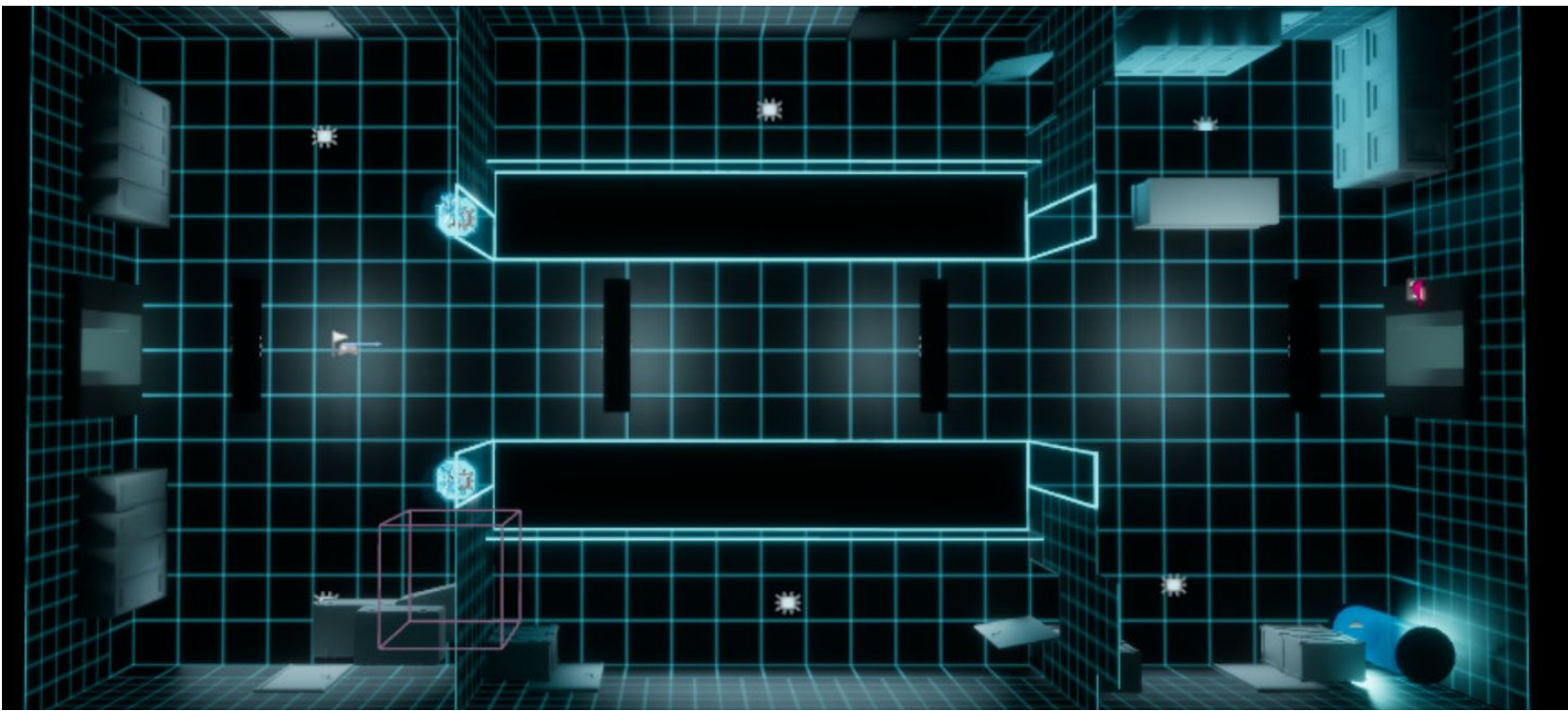
Level Design Overview

Visual profile / Game world:

- Set in a VR world
- Vibe: Dark, futuristic, simplistic, with blue neon lights.
- Mainly blue grid texture on floor, walls etc.
- A mix of a blue outline on a dark base and a glowy blue texture on objects
- The first and second levels are going to be mainly storage building and lab-inspired.
- When enemies die, they dissipate into thin air and release glowing particles.
- Locked areas and danger are visualised with red.
- The world turns from blue to red when the player has been spotted.
- Objectives are marked with bright yellow. Example: Keycard.
- Fog on the bottom of the map. The player dies if they fall into it.
- Add different sizes in the grid texture to add a bit of variation to the walls, etc.
- Bubble particles in the liquid tank make it more lively.
- Levitating holograms for screens. Used for tutorial screens in the lobby and screens for computers to turn off cameras.
- Add a balcony in certain areas with no enemies as a break, where you can stop and look into an endless void of blue glitchy particles with a background gradient going from black at the top to bright blue at the bottom. Like "wow, we're in the matrix but blue!"

Level list:

1. Introduction video + Main Menu / Settings Menu
2. The lobby - Level for tutorials and testing of mechanics.
3. Level 1 - Tutorial level, introduction to the movement and systems.
4. Level 2 - Bridge with 3 enemies at the side, shoot enemies to be able to pass.
5. Level 3 - Tank Hangar, large level with several entry points and obstacles to overcome.
6. "Final Floor" - Easter eggs + Playground.



Level Layout design

Needs to guide the player towards the end.

The levels have been designed so the player must pass through certain obstacles before reaching the goal. In the level above (Tutorial level), they start at the elevator on the left.

From there, they have to observe the enemies on each side of the bridge in the middle. They can hide behind the plants by the glass or sneak to see the enemy's location. The enemies are patrolling back and forth.

The player can't just run over to the other side without being spotted by the enemy's line of sight and has to neutralize them with ranged weapons before they can move towards the other side.

When the player reaches the other side via the end-elevator, it's locked and requires a keycard to open. They then have to go look for the enemy that dropped the keycard. Pick it up and unlock the elevator. And then they win that level and advance towards the next.

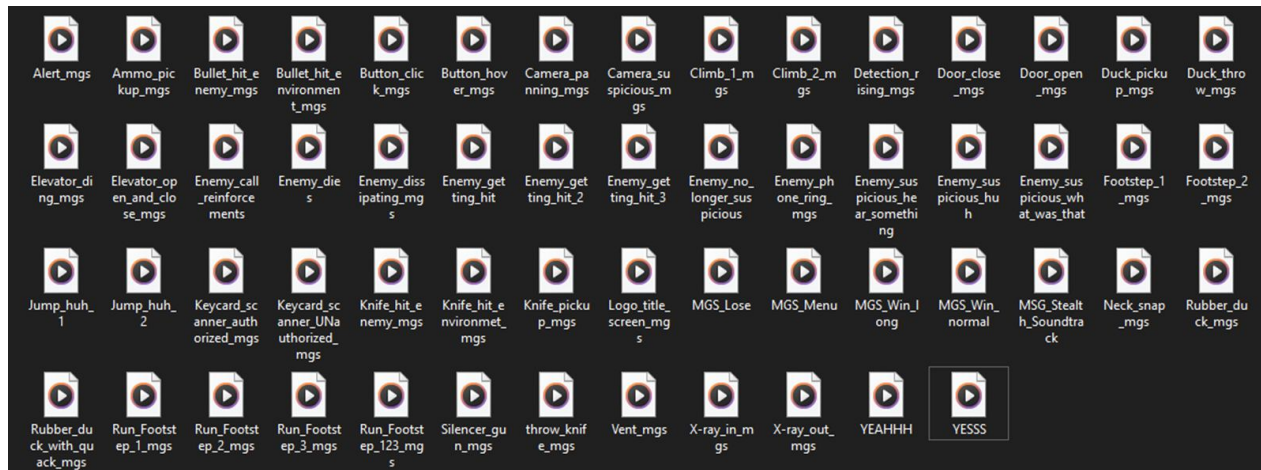
Audio Design and Overview

Sounds are recorded, taken from online websites with royalty free sounds to use such as Pixabay (<https://pixabay.com/sound-effects/>) . Files have then been tweaked in FL Studio before being put into Unreal where audio is tweaked into 2D and 3D space and audio levels are adjusted. Music has been made entirely in FL Studio.

Sound effects list

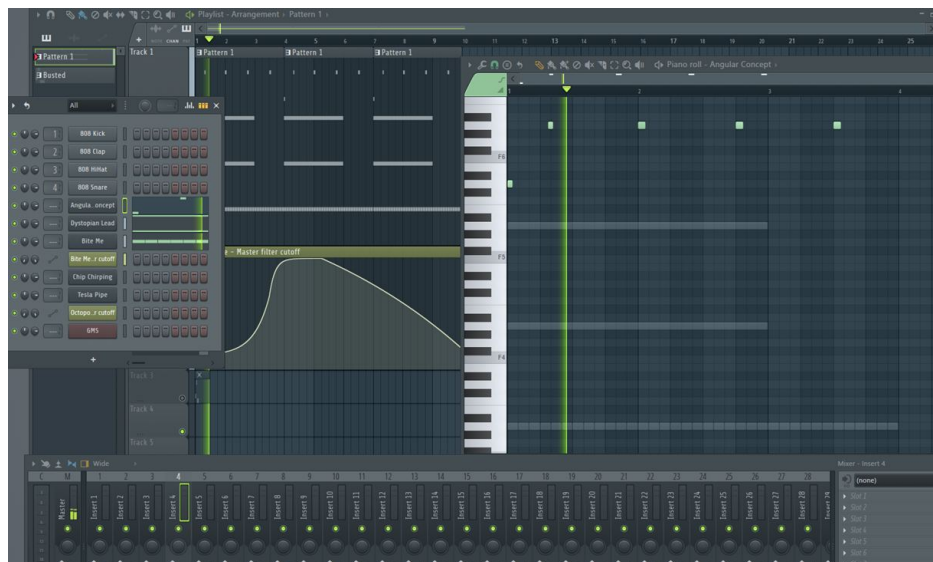
1. Walking footsteps
2. Crouch Xray mode sound (in and out)
3. Jumping (voice)
4. Climbing
5. Shooting with the silencer
6. Throwing knife
7. Knife impact into enemy
8. Hitting the environment with a knife
9. Pick up ammo (silencer, knives, and "Distraction duckie")
10. Throwing a "Distraction duckie"
11. "Distraction duckie" hitting an object or an enemy
12. "Distraction duckie distraction sound (distracts enemies)
13. Detection rising
14. Camera suspicious
15. Camera panning
16. Enemy suspicious (voice)
17. Enemy no longer suspicious (voice)
18. Enemy getting hit (ugh sound)
19. Enemy dying
20. Enemy calling reinforcements
21. Reinforcements call (voice)
22. Enemy dissipating into particles
23. Neck snapping from neutralizing from behind
24. Enemy alert sound
25. Elevator ding
26. Elevator moving
27. Elevator open/close
28. Heavy door open/close
29. Door open /close

30. Vent open/close
31. Keycard scanner authorized
32. “YESSSS” end sound (voice)
33. Title screen
34. Menu Button hover
35. Menu button click



Music list

1. Menu song
2. Stealth mode song
3. Level complete Winning song
4. Game over screen Losing song



Scope Management:

Non-Goals:

- No complex inventory menus; items are used immediately via dedicated inputs.
 - This is to avoid building an extensive inventory system; there is no time for this.
 - It's also not needed for the scope of the project
- No persistent progression; all resources reset at the start of each mission.
 - This is to avoid databases and save information across levels.
- No complex AI squad tactics; enemies follow predictable, readable patterns.
 - Keeping the AI simple, like the original concept. And time management.

Sources:

Player character model:

- Trisutrisno. (2022). Metal Gear Solid Phantom Pain Big Boss Mixamo Rig [3D model]. Sktechfab. <https://skfb.ly/ovLqs>

Enemy character model:

- RAISOR 4096. (2021). Sci-Fi Soldier 02 [3D model]. Fab. <https://fab.com/s/bdcbc0ca94ee>