



ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΤΕΧΝΩΝ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ:
ΑΚΟΥΣΤΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΨΗΦΙΑΚΟΣ ΗΧΟΣ

Πτυχιακή / Διπλωματική Εργασία

Ηχητικός σχεδιασμός για παιχνίδια και
διαδραστικές εφαρμογές εμπύθισης

Πονηρίδης Ηλίας

Επιβλέπων καθηγητής: Δημούλας Χαράλαμπος

Καθηγητής, Τμ. Δημοσιογραφίας & Μέσων Μαζικής Επικοινωνίας

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

2^{ος} Επιβλέπων καθηγητής: Λάζαρος Βρύσης

Μεταδιδακτορικός Ερευνητής Τμήμα Δημοσιογραφίας και Μέσων Μαζικής Επικοινωνίας

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Αθήνα, Σεπτέμβριος 2026

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του

συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

Ηχητικός σχεδιασμός για παιχνίδια και διαδραστικές εφαρμογές εμβύθισης

Πονηρίδης Ηλίας

Επιτροπή Επίβλεψης Μεταπτυχιακής Διπλωματικής Εργασίας:

Επιβλέπων καθηγητής: Δημούλας
Χαράλαμπος

Καθηγητής, Τμ. Δημοσιογραφίας & Μέσων
Μαζικής Επικοινωνίας Αριστοτέλειο
Πανεπιστήμιο Θεσσαλονίκης

2^{ος} Επιβλέπων καθηγητής: Λάζαρος Βρύσης
Μεταδιδακτορικός Ερευνητής Τμήμα
Δημοσιογραφίας και Μέσων Μαζικής
Επικοινωνίας Αριστοτέλειο Πανεπιστήμιο
Θεσσαλονίκης

Αθήνα, Σεπτέμβριος 2026

Εισαγωγή

Αντικείμενο της διπλωματικής εργασίας είναι η δημιουργία ενός ηλεκτρονικού παιχνιδιού με διαδραστικές λειτουργίες εμπύθισης. Αρχικά γίνεται η ανάλυση σε θεωρητικό στάδιο. Στη συνέχεια γίνεται η περιγραφή της μεθοδολογίας ανάπτυξης παιχνιδιού και της περιγραφής των ηχητικών σεναρίων σε ένα παιχνίδι. Τέλος στο 5^ο κεφάλαιο, διεξάγεται εξαγωγή συμπερασμάτων και τελικών παρατηρήσεων

Αυτή η εργασία έχει ως στόχο την δημιουργία και επεξεργασία ηχητικών εφέ και γεγονότων για παιχνίδια και διαδραστικές εφαρμογές μέσω του λογισμικού Wwise σε χρήση με τη μηχανή παιχνιδιού Unity 3D. Σκοπός αυτής της εργασίας είναι η ανάλυση των δυνατοτήτων και λειτουργιών του λογισμικού Wwise και πως μπορεί να βελτιώσει την ακουστική εμπειρία του παίκτη και θα γίνει σύγκριση με άλλα αντίστοιχα προγράμματα διαθέσιμα στο εμπόριο. Τα assets του Unity 3D και τα ηχητικά αρχεία για λόγους ευκολίας είναι διαθέσιμα στο διαδίκτυο και η βιβλιογραφία είναι διαθέσιμη. Τα προσδοκώμενα αποτελέσματα της διπλωματικής εργασίας είναι η δημιουργία ενός παιχνιδιού demo που θα περιλαμβάνει τις λειτουργίες του προγράμματος και η εξαγωγή συμπερασμάτων σχετικά με την όλη διαδικασία.

Η διπλωματική μου εργασία διερευνά τις δυνατότητες του ήχου στα παιχνίδια. Ως λάτρης των βιντεοπαιχνιδιών, εκφράζω το ενδιαφέρον μου να χρησιμοποιώ καινούργια εργαλεία που κυκλοφορούν για να δημιουργήσω συναρπαστικά πράγματα και να πειραματίζομαι πάνω σε αυτά.

Λέξεις – Κλειδιά

Αμφιωτικός ήχος-Μηχανή παιχνιδιού-Προσανατολισμός-Διαδραστικός ήχος

Introduction

The purpose of this thesis is to create a video game with interactive, immersive features. The thesis begins with a theoretical analysis. This is followed by a description of the game development methodology and the audio scenarios in a game. Finally, in Chapter 5 (Λάζαρος, 2023), conclusions and final observations are drawn.

The aim of this thesis is to create and edit sound effects and events for games and interactive applications using Wwise software in conjunction with the Unity 3D game engine. The purpose of this project is to analyze the capabilities and functions of the Wwise software and how it can enhance the player's audio experience, and to compare it with other similar commercially available programs. For convenience, the Unity 3D assets and audio files are available online, and the bibliography is provided. The expected outcomes of this thesis are the creation of a demo game that will showcase the program's features and the drawing of conclusions regarding the entire process.

My thesis explores the possibilities of sound in games. As a video game enthusiast, I am interested in using new tools that are released to create exciting things and experiment with them.

Keywords

Surround sound—Game engine—Orientation—Interactive Sound

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την Κυρία Σωτηροπούλου για την ευκαιρία που μου δόθηκε να ολοκληρώσω το μεταπτυχιακό πρόγραμμα και την οικογένεια μου για την στήριξη τους.

Επίσης τους καθηγητές του μεταπτυχιακού προγράμματος που με βοήθησαν να κατανοήσω τα ακουστικά φαινόμενα που είναι χρήσιμα για την επαγγελματική πορεία μου, την υπομονή και την επιμονή που έδειξα στο μεταπτυχιακό.

Συμπερασματικά, εκφράζω όλα τα άτομα που αναφέρθηκαν παραπάνω καθώς και σε όσους συνέβαλαν στην ολοκλήρωση της διπλωματικής εργασίας μου. Η υποστήριξη σας με βοήθησε στο να την ολοκληρώσω.

Περιεχόμενα

Εισαγωγή	i
Introduction.....	ii
Ευχαριστίες	iii
Κατάλογος Εικόνων/Σχημάτων	vii
Κατάλογος Πινάκων	ix
Ακρωνύμια	x
1. Ανάλυση	1
1.1 Ιστορική αναδρομή	1
1.2 Ήχος των παιχνιδιών	4
1.3 Τύποι εκτέλεσης μουσικής.....	4
1.3.1 Γενετική και προσαρμοστική μουσική	4
1.3.2 Τα οφέλη της αλγοριθμικής μουσικής	6
1.4 Αμφιωτικός ήχος	7
1.5 Ήχος surround	11
1.6 Διαδραστικός ήχος	13
1.7 MIDI	14
1.8 DAW – Digital Audio Workstation.....	15
1.9 Κωδικοποιητής ήχου	16
1.10 SDK.....	16
1.11 Σύνοψη 1 ^{ου} κεφαλαίου	17
2 Σχεδιασμός.....	18
2.1 Μηχανές Παιχνιδιών	18
2.1.1 Unity	18
2.1.2 Unreal Engine	18
2.2 Ενδιάμεσο λογισμικό ήχου (Middleware)	20
2.2.1 Wwise	20
2.2.2 fmod	20
2.2.3 Atmoky	22
2.3 Χάρτες εμπειρίας χρήστη (UX Maps)	23

2.4	Ανάλυση ακροατηρίου	26
2.5	Κύκλος ανάπτυξης παιχνιδιού.	27
2.6	Εργαλεία διανομής κώδικα	28
2.7	Σύνοψη 2 ^{ου} κεφαλαίου	30
3	Ανάπτυξη παιχνιδιού	31
3.1	Unity	32
3.1.1	Αντικείμενα παιχνιδιού (GameObjects).....	32
3.1.2	Δομή κώδικα Unity	32
3.1.3	Αντικείμενα (Assets) παιχνιδιού	35
3.1.4	Υλικά (Materials).....	39
3.1.5	Χειρισμός χαρακτήρα	40
3.1.6	Ενσωματωμένος ήχος	42
3.1.7	Άκαμπτο σώμα (Rigidbody)	44
3.1.8	Συγκρούσεις (Collisions)	45
3.1.9	Κίνηση (Animation).....	46
3.1.10	Κάμερα παιχνιδιού	49
3.1.11	Σκηνές (Scenes)	51
3.1.12	Πράκτορες (Agents).....	51
3.1.13	Έκδοση (Build)	54
3.2	Wwise	55
3.2.1	Ήχοι παιχνιδιού.....	56
3.2.2	Αντικείμενα παιχνιδιού (Game Objects).....	58
3.2.3	Τράπεζες ήχων (Soundbanks).....	62
3.2.4	Δίαυλος ήχου (Bus).....	65
3.2.5	Γεγονότα (Events).....	65
3.2.6	Έλεγχος παραμέτρων σε πραγματικό χρόνο (Real-Time Parameter Controls - RTPC)	69
3.2.7	Προφίλ (Profiler)	73
3.2.8	Πίεση και αποσυμπίεση αρχείων ήχων	74
3.2.9	Ενσωμάτωση του AudioKinetic Wwise με το Unity	77
3.3	Ανασκόπηση 3 ^{ου} κεφαλαίου.....	80

4	Αποτελέσματα.....	81
4.1	Προβλήματα..... Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.	
5	Συμπεράσματα	83
5.1	Μελλοντικά πλάνα	83
6	Βιβλιογραφικές αναφορές.....	84
	Παράρτημα Α: Αρχεία Κώδικα του παιχνιδιού.....	86
	CountdownTimer.cs	86
	BulletBehavior.cs	87
	EnemyBehavior.cs	89
	HealthBar.cs	92
	PlayerMovment.cs	93
	Instantiate.cs.....	96
	PlayerHealth.cs	97
	Explodable.cs	99
	Παράρτημα Β: Διαγράμματα παιχνιδιού	100

Κατάλογος Εικόνων/Σχημάτων

Εικ. 1-1 Παιχνίδι Pong.....	2
Εικ. 1-2 Τσιπ YM2612FM της εταιρίας Yamaha.....	2
Εικ. 1-3 Μικρόφωνο προτομή(dummy) Neumann KU 100.	8
Εικ. 1-4 Μικρόφωνο κεφαλής Head and Torso Simulator (HATS) Type 4128D	9
Εικ. 1-5 Διάγραμμα διάταξης 10.2 surround sound.....	12
Εικ. 1-6 Dolby Atmos	12
Εικ. 1-7 Στιγμιότυπο από το παιχνίδι Rez Infinite.....	13
Εικ. 1-8 Διάγραμμα διασύνδεσης ηλεκτρονικών μουσικών οργάνων με υπολογιστή.....	14
Εικ. 1-9 Audacity.	15
Εικ. 2-1 Παράδειγμα διαγράμματος χάρτη εμπάθειας (Empathy Map)	23
Εικ. 2-2 Παράδειγμα διαγράμματος χάρτη ταξιδιού πελάτη (Customer Journey Map).....	24
Εικ. 2-3 Παράδειγμα διαγράμματος χάρτη εμπειρίας (Experience Map).....	25
Εικ. 2-4 Παράδειγμα διαγράμματος υπηρεσίας (Service Blueprint)	26
Εικ. 2-5 Παράθυρο Github Desktop	29
Εικ. 2-6 Παράθυρο Unity DevOps	30
Εικ. 3-1 Brotato.....	31
Εικ. 3-2 Αρχική οθόνη παιχνιδιού	31
Εικ. 3-3 Παράθυρο περιβάλλοντος Unity.....	32
Εικ. 3-4 synty polygon city pack	35
Εικ. 3-5 Παράθυρο Package Manager	36
Εικ. 3-6 Mixamo	37
Εικ. 3-7 Kenney assets.....	37
Εικ. 3-8 Παράθυρο Unity Asset Store	38
Εικ. 3-9 Τρισδιάστατο μοντέλο χαρακτήρα (Unity).....	38
Εικ. 3-10 Τρισδιάστατο μοντέλο ξωτικού (Unity).....	39
Εικ. 3-11 Παράθυρο Inspector για υλικό	40
Εικ. 3-12 Καρτέλα Player Input.....	41
Εικ. 3-13 Παράθυρο CharacterInputActions	42
Εικ. 3-14 Λίστα συστατικών για τον ήχο στο Unity	43
Εικ. 3-15 Λίστα συστατικών για τα Colliders στο Unity.....	46
Εικ. 3-16 Παράθυρο αντικειμένου GameObject Pickup	46
Εικ. 3-17 Παράθυρο Event ενός Animation Clip.....	47
Εικ. 3-18 Καρτέλα Animation στο Unity.....	48
Εικ. 3-19 Μηχανή κατάστασης Animator στο Unity.....	49
Εικ. 3-20 Παράθυρο στοιχείων (Components) του αντικειμένου VirtualCamera.....	50

Εικ. 3-21 Σκηνή παιχνιδιού.....	51
Εικ. 3-22 Παράθυρο NavMesh Surface.....	52
Εικ. 3-23 Παράθυρο NavMesh Surface.....	53
Εικ. 3-24 Παράθυρο Build Profiles	54
Εικ. 3-25 Παράθυρο εκκίνησης Audiokinetic Launcher	56
Εικ. 3-26 Audiokinetic Wwise.....	56
Εικ. 3-27 Παράθυρο Soundly.	57
Εικ. 3-28 AkEvent Component.....	57
Εικ. 3-29 Παράθυρο Event Editor στο Wwise.....	58
Εικ. 3-30 Παράθυρο Wwise reflect	60
Εικ. 3-31 Παράθυρο Unity Editor.....	61
Εικ. 3-32 Εικόνα The Wwise adventure game.....	62
Εικ. 3-33 Παράθυρο Soundbank με την καρτέλα Add.	63
Εικ. 3-34 Παράθυρο επεξεργασίας ήχου	64
Εικ. 3-35 Παράθυρο profiler του Wwise	64
Εικ. 3-36 Παράθυρο Audio Bus.....	65
Εικ. 3-37 Παράδειγμα μεταβλητής RTPC	73
Εικ. 3-38 Παράθυρο Profiling του Wwise.....	73
Εικ. 3-39 Παράθυρο Remote Connection του Wwise.	77
Εικ. 3-40 Παράθυρο Project Settings Wwise Integration.....	78
Εικ. 3-41 Παράθυρο WwiseGlobal.....	79
Εικ. 4-1 Παράθυρο profiler.....	81
Εικ. 0-1 Διάγραμμα Animator παίκτη.....	100
Εικ. 0-2 Διάγραμμα animation εχθρού.....	100
Σχήμα 1-1 Απεικόνιση συνάρτησης για αμφιωτική παραγωγή σε ανηχικό περιβάλλον	10
Σχήμα 3-1 Κύκλος ζωής κώδικα MonoBehaviour.....	34
Σχήμα 3-2 Σχεδιάγραμμα γεγονότος Event στο Wwise.	66
Σχήμα 3-3 Σχεδιάγραμμα γεγονότος για την επιστροφή του παίκτη στο μενού	66
Σχήμα 3-4 Παράδειγμα γεγονότος δράσης στο Audiokinetic Wwise	67
Σχήμα 3-5 Παράδειγμα γεγονότος διαλόγου στο Wwise	68

Κατάλογος Πινάκων

Πίνακας 3-1 Πίνακας περιγραφής συστατικών Audio του Unity.....	44
Πίνακας 3-2 Πίνακας συστατικών του NavMesh.....	52
Πίνακας 3-3 Πίνακας ενσωματωμένων (Built in) παραμέτρων RTPC του Wwise.....	72
Πίνακας 3-4 Πίνακας υποστηριζόμενων κωδικοποιητών ήχου για Wwise	74
Πίνακας 3-5 Πίνακας συμπίεσης ήχου ανά πλατφόρμα	75
Πίνακας 3-6 Πίνακας περιγραφής συστατικών του Wwise στο Unity.	80

Ακρωνύμια

ADPCM	Adaptive Differential Pulse Code Modulation
AI	Artificial Intelligence
CPU	Central Processing Unit
DAW	Digital Audio Workstation
DLC	Downloadable Content
FM	Frequency Modulation
HRTF	Head-related Transfer Function
LPC	Linear Predictive Coding
MDCT	Modified Discrete Cosine Transform
MIDI	Musical Instrument Digital Interface
NES	Nintendo Entertainment System
NPC	Non Playable Character
RTPC	Real-time Parameter Controls
SDK	Software Development Kit
SNES	Super Nintendo Entertainment System
VR	Virtual Reality
XR	Extended Reality

1. Ανάλυση

Στο κεφάλαιο αυτό ξεκινάει με μία συνοπτική αναδρομή στην ιστορία του ρόλου του ήχου στα βιντεοπαιχνίδια. Στη συνέχεια θα παρουσιαστεί δοθεί η περιγραφή του ρόλου του ήχου στο παιχνίδι με ανάλυση εννοιών σχετικά με το interactive audio (binaural audio, spatial audio, randomized sound) .

1.1 Ιστορική αναδρομή

Ο ήχος στο παιχνίδι πρωτοεμφανίστηκε το 1972 στο παιχνίδι Pong του Atari με το χαρακτηριστικό “boop” που κάνει ο χτύπος από το μπαλάκι. Η μουσική υπόβαθρου ή αλλιώς background music έκανε αργότερα την εμφάνιση της στο παιχνίδι Super Mario Bros του 1985 που έμελλε να γίνει ένα από τα πιο αναγνωρίσιμα κομμάτια όλων των εποχών σε βιντεοπαιχνίδι που φυλάσσεται στο εθνικό μητρώο καταγραφών της βιβλιοθήκης του κογκρέσου (O’Dell, 2023), και παράλληλα έγινε η μετάβαση προς στο μονοφωνικό ήχο που παρείχε ο επεξεργαστής Ricoh 2A03 που διέθετε το NES. Κατά τη μετάβαση από τη δεκαετία του 80 στη δεκαετία του 90 έγινε η μετάβαση στα 16-bit όπου οι κονσόλες μπορούσαν να αναπαράγουν πιο πολλά χρώματα και διέθεταν πιο προχωρημένα τσιπ ήχου όπως για παράδειγμα το YM2612 FM που χρησιμοποιήθηκε στο Sega Mega drive και παρήγαγε πιο οξύς ήχους (YM2612: The chip that powered music on the Mega Drive, 2021) και του Sony SPC700 που χρησιμοποιήθηκε στο SNES και παρήγαγε πιο ορχηστρικούς ήχους. Οι συνθέτες είχαν την δυνατότητα να συνθέσουν πιο σύνθετους και εκφραστικούς ήχους χρησιμοποιώντας συνθεσάιζερ όπως για παράδειγμα η μουσική του Sonic the Hedgehog. Μέσα στη δεκαετία, ο οπτικός δίσκος ή αλλιώς CD έκανε την πρώτη της επιτυχημένη εμπορική εμφάνιση στην κονσόλα Playstation 1 που επέτρεπε την σύνθεση καλύτερης ποιότητας μουσική που θα συγκρινόταν με την μουσική υπόκρουση των ταινιών, μεγαλύτερη χωρητικότητα που χωρούσε ηχητικά εφέ και φωνητικές γραμμές. Στην δεκαετία του 2000, άρχισε να υιοθετείται το δυναμικά συστήματα ήχου που προσαρμόστε τη μουσική και τα ηχητικά εφέ σε πραγματικό χρόνο με βάση τις ενέργειες του παίκτη. Το dolby digital καθιερώθηκε ως στάνταρντ δυνατότητα ακούγοντας τα βήματα του χαρακτήρα, όσο μεγάλωναν τα παιχνίδια σε κλίμακα τόσο πιο περίτεχνα γινόταν η μουσική στα παιχνίδια με ορχηστρικές συνθέσεις και δυνατές ερμηνείες χαρακτήρων όπως για παράδειγμα στο The Witcher 3. Η εμφάνιση των Middleware όπως για παράδειγμα το FMOD και το Wwise συνέβαλε στην δυναμική σύνθεση ήχου σε μεγάλες όσο και σε μικρές ομάδες ανάπτυξης. Το VR ή αλλιώς εικονική πραγματικότητα δίνει μεγάλη βαρύτητα στον ήχο που θα πρέπει να προσαρμόζεται για να προσφέρει καθηλωτική εμπειρία στον παίκτη και η χρήση τέτοιων εργαλείων κρίνεται επιτακτική. Στο μέλλον αναμένεται να εφαρμοστούν τεχνολογίες AI και Ray tracing για καλύτερη απόδοση ζωντανού ήχου στο παιχνίδι.



Εικ. 1-1 Παιχνίδι Pong



Εικ. 1-2 Τσιπ YM2612FM της εταιρίας Yamaha

Πηγή: https://en.wikipedia.org/wiki/Yamaha_YM2612

1.2 Ήχος των παιχνιδιών

Ο ήχος των παιχνιδιών κατατάσσεται συνήθως στις κατηγορίες: ομιλίας, ήχου/ηχητικών εφέ και μουσικής. Η ομιλία αναφέρεται στον ηχητικό ήχο του παιχνιδιού που προέρχεται από έναν χαρακτήρα του παιχνιδιού. Αυτό περιλαμβάνει τον χαρακτήρα παίκτη (PC) και τους χαρακτήρες που δεν παίζει ο παίκτης (NPC). Οι ηχογραφήσεις ανθρώπινων φωνών αποτελούν την πιο συνηθισμένη πηγή για την ομιλία, αλλά η ομιλία μπορεί επίσης να συντεθεί. Ο ήχος/εφέ αναφέρεται γενικά σε ηχητικά στοιχεία που είναι μη περιοδικά και μη μουσικά, συχνά εμπνευσμένα από πραγματικά ηχητικά εφέ, όπως η πυροβολισμοί ή ο άνεμος που σφυρίζει στα δέντρα. Τα στοιχεία ήχου/εφέ μπορούν επίσης να είναι αφηρημένα και τεχνητά, όπως το «γουακα-γουακα» του Pac-man, ο ήχος του Mario που πηδάει, ή ένα απλό ηχητικό σήμα όταν πατιέται ένα κουμπί. Η τελική κατηγορία του ήχου στα παιχνίδια είναι η μουσική, και σε αυτήν εστιάζουμε.

Αν και ο ορισμός της «μουσικής» είναι αμφιλεγόμενο ζήτημα, χρησιμοποιούμε τον ορισμό του Luciano Berio για τη μουσική ως οτιδήποτε προορίζεται να είναι μουσική.

Γενικά έχει συγκεκριμένη τονικότητα και κάποια τακτική χρονική διαίρεση, αλλά αυτά τα χαρακτηριστικά δεν είναι απαραίτητα. Είναι σημαντικό να σημειωθεί ότι, ενώ οι διαχωρισμοί ομιλίας/ήχου/ μουσικής είναι χρήσιμοι για την περιγραφή του ηχητικού περιεχομένου στα παιχνίδια, οι διαχωρισμοί αυτοί είναι περιγραφικοί και όχι κανονιστικοί, και τα όρια μεταξύ αυτών των κατηγοριών δεν είναι αμετάβλητα. Εν ολίγοις, ενώ περιορίζουμε το πεδίο μας στη μουσική των παιχνιδιών, υιοθετούμε μια περιεκτική προοπτική ως προς το τι αποτελεί μουσική παιχνιδιών.

(Cale Plut, 2020)

1.3 Τύποι εκτέλεσης μουσικής

1.3.1 Γενετική και προσαρμοστική μουσική

Η πιο συνηθισμένη χρήση της μουσικής στα παιχνίδια είναι η αναπαραγωγή ενός γραμμικού, συνθετικού μουσικού κομματιού κατά τη διάρκεια του παιχνιδιού. Σε πολλά παιχνίδια, η μουσική συνδέεται άμεσα με το τρέχον επίπεδο ή/και την κατάσταση του παιχνιδιού. Στην περίπτωση της γραμμικής συνθετικής μουσικής, η αναπαραγωγή του μουσικού κομματιού ξεκινά όταν φορτώνεται το αντίστοιχο επίπεδο. Όταν η μουσική φτάσει στο τέλος του κομματιού, επαναλαμβάνεται. Όταν φορτώνεται ένα νέο επίπεδο, η μουσική είτε αλλάζει απότομα είτε σβήνει γρήγορα και αντικαθίσταται από τη μουσική που σχετίζεται με το νέο επίπεδο.

Αυτή η χρήση της μουσικής συνδέεται συχνά με παλαιότερα παιχνίδια αλλά εξακολουθεί να χρησιμοποιείται ακόμη και σήμερα

Η γραμμική μουσική σύνθεση μπορεί επίσης να χρησιμοποιηθεί χωρίς σαφείς διαχωρισμούς επιπέδων. Καθώς η μνήμη των υπολογιστών έχει γίνει πιο ευρύχωρη και οι τεχνικές προγραμματισμού επιτρέπουν την απρόσκοπτη φόρτωση περιεχομένου, τα παιχνίδια βασίζονται λιγότερο σε σαφείς διαχωρισμούς επιπέδων μεταξύ των καταστάσεων του παιχνιδιού. Το είδος του «ανοιχτού κόσμου/Open World» αποτελεί παράδειγμα αυτού, με τα παιχνίδια ανοιχτού κόσμου να ελαχιστοποιούν ή να εξαλείφουν εντελώς τις διαχωριστικές γραμμές μεταξύ των επιπέδων. Τέτοια παιχνίδια εξακολουθούν συχνά να έχουν σαφείς διαχωριστικές γραμμές μεταξύ των καταστάσεων του παιχνιδιού. Η πιο συνηθισμένη αλλαγή κατάστασης είναι αυτή της μετάβασης από «μάχης» και «περιπλάνηση».

Υπάρχουν δύο βασικές τεχνικές για την επέκταση της γραμμικής, συνθετικής μουσικής στα παιχνίδια. Η πρώτη τεχνική – η προσαρμοστική μουσική – αφορά τη γραμμική χρήση της μουσικής. Η προσαρμοστική μουσική ονομάζεται μερικές φορές «διαδραστική μουσική» και είναι μουσική που αντιδρά στην κατάσταση ενός παιχνιδιού. Η προσαρμοστική μουσική μπορεί να προσφέρει μεγάλο εύρος μοναδικής μουσικής από περιορισμένο μουσικό περιεχόμενο. Η προσαρμοστική μουσική συνδέει άμεσα τα μουσικά χαρακτηριστικά με τις μεταβλητές του παιχνιδιού. Αυτά τα χαρακτηριστικά μπορούν να περιλαμβάνουν την προσθήκη ή αφαίρεση οργάνων, την αλλαγή του τέμπο, την προσθήκη ή αφαίρεση επεξεργασίας, την αλλαγή του ύψους του ήχου κ.λπ. Αυτές οι αλλαγές στην προσαρμοστική μουσική συνδέονται άμεσα με τις μεταβλητές του παιχνιδιού.

Η προσαρμοστικότητα της μουσικής μπορεί να γίνει κατανοητή ως μια διάσταση. Τα χαμηλά επίπεδα προσαρμοστικότητας μπορεί να προσαρμόζονται μόνο σε ένα μικρό σύνολο μεταβλητών εντός του παιχνιδιού, ενώ τα υψηλότερα επίπεδα προσαρμοστικότητας μπορεί να προσαρμόζονται σε δεκάδες ή εκατοντάδες μεταβλητές εντός του παιχνιδιού.

Η άλλη τεχνική – η γενετική μουσική – αφορά τη δημιουργία του ίδιου του μουσικού περιεχομένου. Το μεγαλύτερο μέρος της μουσικής συντίθεται από ένα άτομο ή μια ομάδα ανθρώπων συνθετών. Η υπολογιστική δημιουργικότητα είναι ένα πεδίο που διερευνά την αυτοματοποίηση των δημιουργικών εργασιών, ενώ η μουσική μεταδημιουργία (MuMe) αποτελεί παρακλάδι της υπολογιστικής δημιουργικότητας που ασχολείται με την αυτοματοποίηση της δημιουργίας της μουσικής. Η γενετική μουσική είναι μουσική που δημιουργείται μέσω συστημικής αυτοματοποίησης και μερικές φορές αποκαλείται και ως διαδικαστική μουσική, μουσική μεταδημιουργία ή αλγοριθμική μουσική. Αυτοί οι όροι είναι ως επί το πλείστον

συνώνυμοι και μπορούν να χρησιμοποιηθούν εναλλακτικά, αλλά θα χρησιμοποιήσουμε τον όρο «γενετική μουσική» για λόγους απλότητας.

Η γενετική μουσική μπορεί να προσφέρει ατελείωτη μοναδική μουσική σε ένα παιχνίδι και μπορεί να είναι προσαρμοστική σε πολύ βαθύτερο επίπεδο από τη συνθετική προσαρμοστική μουσική, παρέχοντας μουσική που είναι εξατομικευμένη στις ενέργειες του παίκτη σε ένα

παιχνίδι. Παρά τα πιθανά οφέλη αυτά, η γενετική μουσική δεν έχει ακόμη επιτύχει ευρεία χρήση στα βιντεοπαιχνίδια.

1.3.2 Τα οφέλη της αλγοριθμικής μουσικής

Τα περισσότερα παιχνίδια διαθέτουν soundtrack διάρκειας μερικών ωρών, αλλά ο χρόνος παιχνιδιού μπορεί να κυμαίνεται από 6 έως και πάνω από 100 ώρες. Αυτό συχνά έχει ως αποτέλεσμα ο παίκτης να ακούει τα μουσικά κομμάτια να επαναλαμβάνονται πολλές φορές. Ενώ η επανάληψη είναι σημαντική στη μουσική, η υπερβολική επανάληψη μπορεί να διαταράξει την εμβύθιση και να γίνει ενοχλητική για τον παίκτη. Ένας παράγοντας που συμβάλλει στην επανάληψη της μουσικής στα παιχνίδια είναι το κόστος, δηλαδή εάν το παιχνίδι έχει αρκετές ώρες μοναδικής μουσικής τότε τα κόστη μελοποίησης αυξάνονται.

Ένα σύστημα γενετικής μουσικής (generative music) μπορεί να προσφέρει ποσότητα περιεχομένου που ξεπερνά κατά πολύ τις δυνατότητες ενός συνθέτη, με πολύ χαμηλότερο κόστος ανά λεπτό μουσικής. Αυτό θα επέτρεπε στους σχεδιαστές ενός μεγάλης σε έκτασης παιχνίδι να γεμίσουν και τις 60 ώρες παιχνιδιού με μοναδική μουσική, με κόστος πολύ χαμηλότερο από το κόστος 60 ωρών συνθετικής μουσικής. Η γενετική μουσική μπορεί ακόμη και να χρησιμοποιηθεί για να παρέχει επιπλέον μουσική χωρίς δημιουργία σε πραγματικό χρόνο – ένας συνθέτης μπορεί να χρησιμοποιήσει τη γενετική μουσική για να δημιουργήσει μια μεγάλη βιβλιοθήκη μουσικής για μια προσαρμοστική παρτιτούρα

Η γενετική μουσική επιτρέπει επίσης στον παίκτη να προσαρμόζει και να εξατομικεύει τη μουσική του. Η προσαρμοστική μουσική επιτρέπει στη μουσική να συντονίζεται πιο στενά με τις ενέργειες ενός διαδραστικού παιχνιδιού, και έχει αποδειχθεί ότι έχει μεγαλύτερη επίδραση στην ένταση που βιώνει ο παίκτης σε σύγκριση με τη γραμμική μουσική αλλά, η προσαρμοστική μουσική δεν μπορεί απαραίτητα να ανταποκριθεί στο εξαιρετικά ευρύ φάσμα των δυνατοτήτων του παιχνιδιού. Η γενετική μουσική μπορεί να συντεθεί σε πραγματικό χρόνο καθώς ο παίκτης αλληλεπιδρά με το παιχνίδι, επιτρέποντάς της να προσαρμόζεται πιο πλήρως στις ενέργειες του παίκτη.

Τέλος, η γενετική μουσική μπορεί να ενδυναμώσει και να βοηθήσει τους ανθρώπινους συνθέτες. Ενώ οι περιορισμοί στη δημιουργική ελευθερία συχνά, παραδόξως, επιτρέπουν μεγαλύτερη δημιουργικότητα, οι υπερβολικοί δημιουργικοί περιορισμοί μπορεί να προκαλέσουν απογοήτευση στους συνθέτες και να περιορίσουν το εκφραστικό τους εύρος.

Όταν συνθέτουν για προσαρμοστική μουσική, οι συνθέτες πρέπει να διασφαλίζουν ότι οποιοσδήποτε από τους πιθανούς συνδυασμούς μουσικής που μπορεί να προκύψει θα ακούγεται αποδεκτός συνολικά. Αυτό περιορίζει σε μεγάλο βαθμό το πιθανό εκφραστικό εύρος του συνθέτη. Η γενετική μουσική μπορεί να επιτρέψει στους συνθέτες να εστιάσουν την προσοχή

τους στις καλλιτεχνικές πτυχές της σύνθεσης μουσικής και όχι στις τεχνικές λεπτομέρειες της προετοιμασίας της μουσικής για προσαρμοστικότητα. Αυτό θα επιτρέψει μεγαλύτερη δημιουργική ελευθερία στους σχεδιαστές παιχνιδιών, τους σχεδιαστές ήχου και τους συνθέτες.

(Cale Plut, 2020)

1.4 Αμφιωτικός ήχος

Η ιδέα πίσω από την τεχνική της βιναυρικής ηχογράφησης είναι η εξής: Η είσοδος στην ακοή αποτελείται από δύο σήματα που το καθένα αντιστοιχεί σε ηχητική πίεση σε κάθε ένα από τα τύμπανα. Εάν αυτά ηχογραφηθούν στα αυτιά ενός ακροατή και αναπαραχθούν ακριβώς όπως ήταν, τότε θεωρείται ότι αναπαράγεται η πλήρης ακουστική εμπειρία, συμπεριλαμβανομένων του ηχοχρώματος και των χωρικών χαρακτηριστικών. Ο όρος «βιναυρική ηχογράφηση» αναφέρεται στο γεγονός ότι οι δύο εισόδους του ακουστικού συστήματος αναπαράγονται σωστά.

Η ηχογράφηση μπορεί να πραγματοποιηθεί με μικρά μικρόφωνα τοποθετημένα στους αυτιούς πόρους ενός ακροατή, αλλά συνήθως χρησιμοποιείται ένα ομοίωμα ανθρώπινου κεφαλιού. Το ομοίωμα έχει το σχήμα ενός μέσου ανθρώπινου κεφαλιού, συμπεριλαμβανομένης της μύτης, των οφθαλμικών κόγχων, των πτερυγίων και των αυτιών πόρων, ενώ μερικές φορές το κεφάλι είναι ακόμη και συνδεδεμένο με ένα ομοίωμα κορμού. Επίσης, μερικές φορές προσομοιώνεται η ακουστική αντίσταση του τυμπανικού υμένα. Με την ακριβή αναπαραγωγή ενός ανθρώπινου κεφαλιού διασφαλίζεται ότι τα ηχητικά κύματα που φτάνουν στο κεφάλι, υφίστανται την ίδια μετάδοση κατά τη διαδρομή τους προς τους ακουστικούς πόρους, σαν να έφταναν σε έναν πραγματικό ακροατή. Ένα αντίγραφο του ανθρώπινου κεφαλιού ονομάζεται τεχνητό κεφάλι, κούκλα κεφαλιού, προσομοιωτής κεφαλιού ή, στα γερμανικά, Kunstkopf. Αυτοί οι όροι έχουν εμπνεύσει εναλλακτικούς όρους για την τεχνική της στερεοφωνικής ηχογράφησης: τεχνική ηχογράφησης με τεχνητό κεφάλι, τεχνική κούκλας κεφαλιού και Kunstkopftechnik. Χρησιμοποιείται επίσης η έκφραση τεχνική σχετιζόμενη με το κεφάλι.

Κάποια αμφιωτικά μικρόφωνα λαμβάνουν υπόψιν και τις διαστάσεις και το σχήμα του προσώπου και του στήθους εξηγώντας τυχόν διακυμάνσεις στον αμφιωτικό ήχο από διαφορετικούς ανθρώπους.

Σκοπός του κάθε μικροφώνου είναι να καταγράψει ήχους με την ίδια χωρική σχέση, κάθε ηχητικό σήμα έχει σχετικές διαφορές μεταξύ των ακουστικών χρόνων, επιπέδων και ηχοχρωμάτων που θα είχαν τα δύο αυτιά ενός ανθρώπου αν βρίσκονταν στην ίδια θέση με το αμφιωτικό κεφάλι. Τα ακουστικά παίζουν τους ήχους που παράγουν τα μικρόφωνα ξεχωριστά στο κάθε αυτί. Ο εγκέφαλος μας μπορεί να ερμηνεύσει χωρικά ερεθίσματα μέσα στις ηχογραφήσεις για να δημιουργήσει ένα τρισδιάστατο ηχοτοπίο μέσα στη συνείδησή μας επειδή

αντικατοπτρίζει το δικό μας ακουστικό σύστημα. Όπως και στα αυτιά ένα αμφιωτικό κεφάλι καταγράφει ήχους από όλες τις κατευθύνσεις στον χώρο αλλά και τυχών θορύβους

(Møller, 1992)



Εικ. 1-3 Μικρόφωνο προτομή(dummy) Neumann KU 100.

Πηγή: https://en.wikipedia.org/wiki/Binaural_recording



Εικ. 1-4 Μικρόφωνο κεφαλής Head and Torso Simulator (HATS) Type 4128D

Πηγή: <https://www.bksv.com/en/transducers/simulators/head-and-torso/hats-type-4128d>

(David Murphy, 2010)

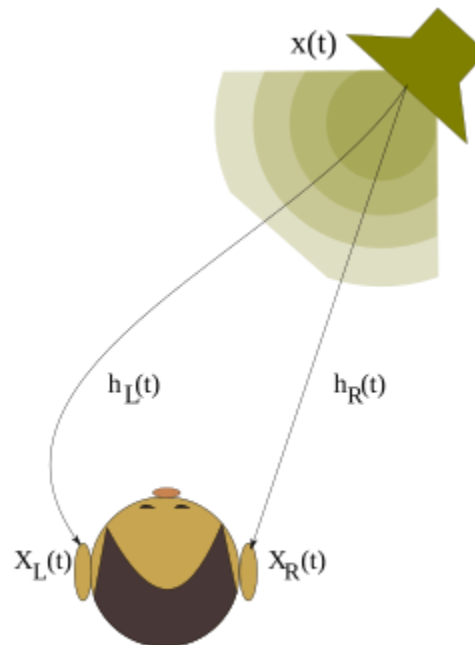
Η συνάρτηση HRTF (Head-related transfer function) χρησιμοποιείται στο χωρικό ήχο. Χαρακτηρίζεται από την ηχητική αντίληψη του ακροατή σε μία πηγή ήχου στον χώρο και των επιμέρους χαρακτηριστικών (Ένταση, συχνότητα και χρόνος). Ρόλο στην αντίληψη του ήχου παίζουν το σχήμα του κεφαλιού και των επιμέρους στοιχείων (μήκος στοματικής κοιλότητας και αυτιών).

Υπάρχουν δύο είδη συναρτήσεων μετασχηματισμού Fourier για αμφιωτική παραγωγή, σε ανηχωικό και σε εικονικό χώρο περιβάλλον .

Σε ανηχωικό περιβάλλον:

$$y_L(t) = x(t) * HRTF_L(t)$$

$$y_R(t) = x(t) * HRTF_R(t)$$



Σχήμα 1-1 Απεικόνιση συνάρτησης για αμφιωτική παραγωγή σε ανηχωικό περιβάλλον

Πηγή: https://en.wikipedia.org/wiki/Head-related_transfer_function

Σε εικονικό περιβάλλον:

$$y_L(t) = x(t) * h_L(t) * HRTF_L(t)$$

$$y_R(t) = x(t) * h_R(t) * HRTF_R(t)$$

(Χατζηαντωνίου)

1.5 Ήχος surround

Ο ήχος τύπου Surround αποτελεί μία τεχνική ενίσχυσης του βάθους και της πιστότητας του ήχου χρησιμοποιώντας πολλαπλά κανάλια ήχου από ηχεία περιτριγυρισμένα γύρω από τον ακροατή. Τα ηχεία είναι έτσι τοποθετημένα έτσι ώστε ο θεατής να έχει την αντίληψη της χωροθέτησης ήχου. Η διαφορά του ήχου Surround σε σχέση με το Stereo είναι ότι το Stereo χρησιμοποιεί δύο ηχεία τοποθετημένα αριστερά και δεξιά ενώ με το Surround πολλαπλά ηχεία που παίζουν από πολλές διευθύνσεις δημιουργώντας ρεαλιστικούς ήχους.

Τα συστήματα ήχου surround χρησιμοποιούν ακουστικά ερεθίσματα για να επηρεάσουν την αντίληψη του ανθρώπινου εγκεφάλου σχετικά με τη θέση και την απόσταση μέσω αμφιβληστροειδών ερεθισμάτων και της προσεκτικής τοποθέτησης των ηχείων σε έναν εσωτερικό χώρο, ώστε να δημιουργήσουν ισχυρά εφέ εμβύθισης.

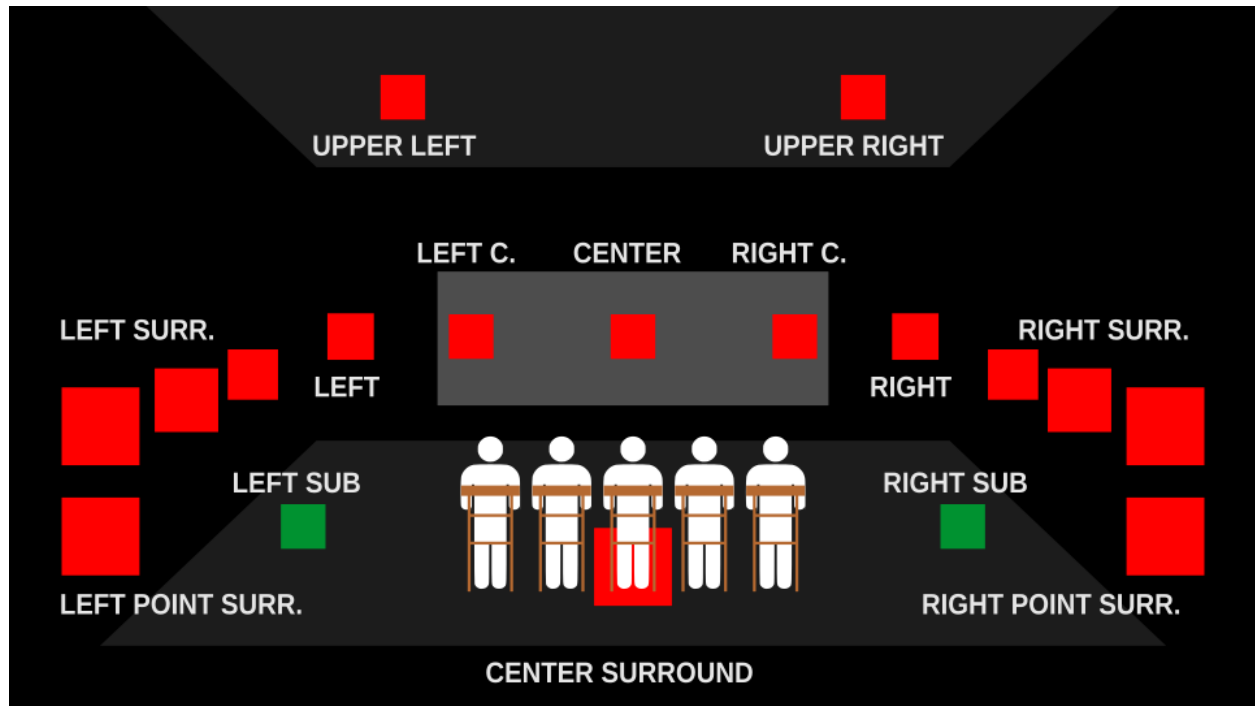
Ο ήχος surround δημιουργεί ένα εμβυθιστικό ηχητικό πεδίο, προσαρμόζοντας τις παρακάτω παραμέτρους στο σύνολο των ηχείων:

- Συγχρονισμός ήχων και θορύβων
- Ηχητική ισορροπία και δυναμική ένταση κάθε πηγής ήχου
- Λεπτές διακυμάνσεις στο ηχόχρωμα με κινούμενους ήχους
- Αντιληπτή θέση των ήχων στον χώρο

Ο ήχος ταξιδεύει σε όλο το δωμάτιο για να δημιουργήσει ένα πραγματικό ηχητικό περιβάλλον που μιμείται το αρχικό σκηνικό για παράδειγμα σε μία ταινία ή ένα παιχνίδι. επιτρέπει στο κοινό να νιώσει κάτι παραπάνω από μία απλή θέαση ταινίας σαν να βρίσκεται στον ίδιο χώρο με αυτήν.

Ο αριθμός του συστήματος ήχου Surround αποτελείται από τον αριθμό των ηχείων που παίζουν τους ήχους οι οποίοι είναι αντιληπτοί από το ανθρώπινο αυτί, των subwoofer που παίζουν τις χαμηλές συχνότητες των ήχων και των ηχείων που είναι τοποθετημένοι σε ύψος. αντίστοιχα όπως παρακάτω στο διάγραμμα. Έχουμε 10 κανονικά ηχεία και 2 subwoofer.

Η τεχνολογία αυτή χρησιμοποιείται κυρίως στους κινηματογράφους και άρχισε να εφαρμόζεται και σε διάφορα καταναλωτικά αγαθά όπως ακουστικά και τηλεοράσεις αλλά και σύγχρονες κονσόλες παιχνιδιών προσφέροντας στους παίκτες γρηγορότερους χρόνους αντίδρασης, διέλευση χαμηλής καθυστέρησης και συναρπαστικές εμπειρίες παιχνιδιών.



Εικ. 1-5 Διάγραμμα διάταξης 10.2 surround sound



Εικ. 1-6 Dolby Atmos

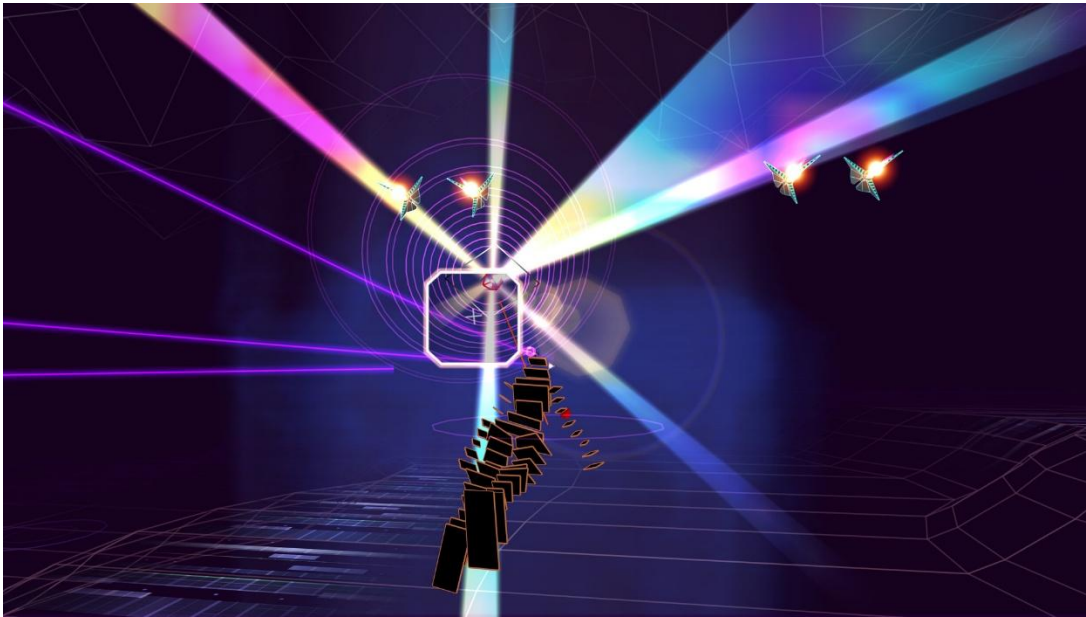
Πηγή: <https://www.dolby.com/>

(Dolby, 2025)

1.6 Διαδραστικός ήχος

Ο διαδραστικός ήχος (interactive audio) είναι ο ήχος που τα δυναμικά χαρακτηριστικά του αλλάζουν ανάλογα με την είσοδο του χρήστη ή τις αλλαγές του περιβάλλοντος, δημιουργώντας καθηλωτικές εμπειρίες σε παιχνίδια και διαδραστικές εφαρμογές. Γίνεται μέσω αλληλεπιδράσεων ανθρώπου-υπολογιστή, οποιαδήποτε είσοδος είναι μία συσκευή όπως για παράδειγμα ένα μικρόφωνο ή ένα κινητό τηλέφωνό που οι πληροφορίες χρησιμοποιούνται για την μεταβολή χαρακτηριστικών και συμπεριφορών ήχου. Μέσα σε τρισδιάστατα εικονικά περιβάλλοντα, ο ήχος αλλάζει ανάλογα με τις ενέργειες ή τα συμβάντα που ενεργοποιούνται.

Το παιχνίδι Rez Infinite αποτελεί ένα από τα πιο σημαντικά παραδείγματα στον διαδραστικό ήχο λόγω του συναισθητικού ήχου όπου η μουσική και ο ήχος συνδυάζονται μαζί με τα εφέ, δημιουργώντας μία ατμόσφαιρα. Εκεί ο διαδραστικός ήχος παίζει μεγάλη σημασία όπου κάθε βολή και χτύπημα στον εχθρού συμβάλλει στη μουσική διασφαλίζοντας ότι ο ήχος αντιδρά δυναμικά στο ρυθμό του παίκτη.

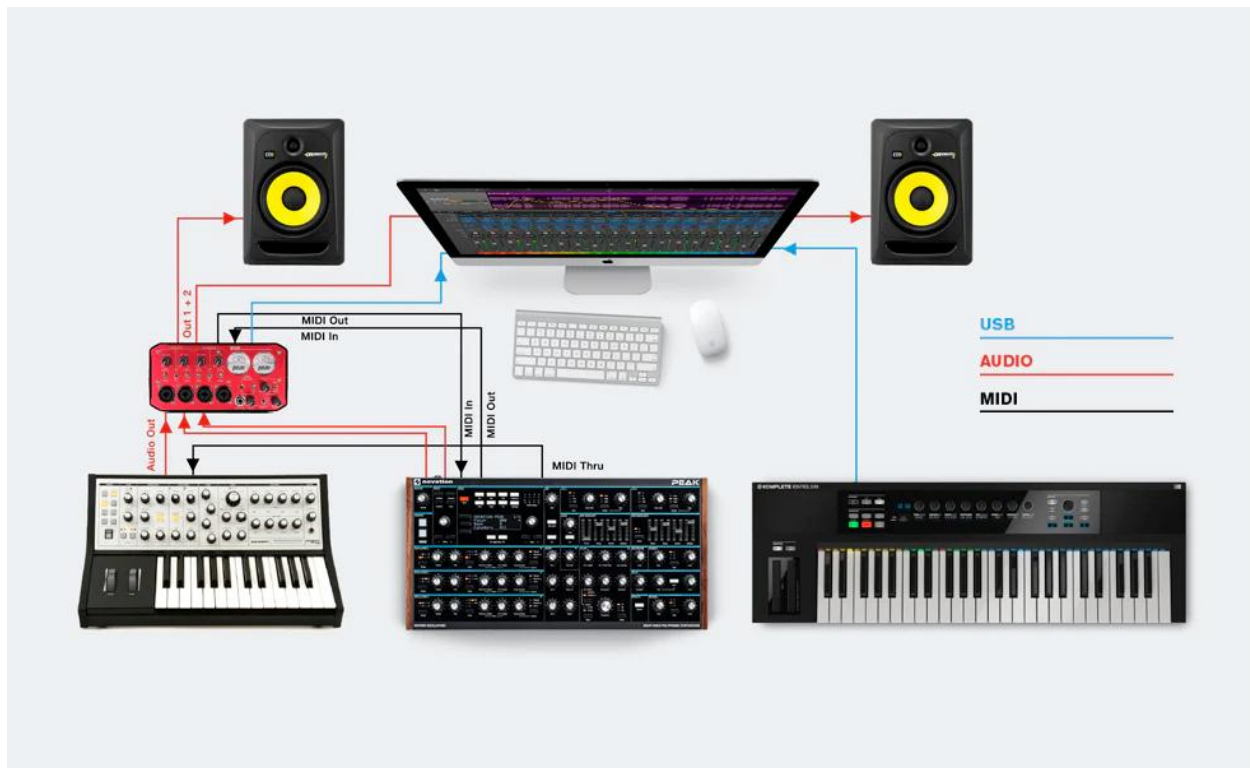


Εικ. 1-7 Στιγμιότυπο από το παιχνίδι Rez Infinite

Πηγή: https://store.steampowered.com/app/636450/Rez_Infinite.

1.7 MIDI

Το MIDI αποτελεί ένα πρωτόκολλο που καθιερώθηκε το 1983 που διασυνδέει υπολογιστές και ηλεκτρονικά μουσικά όργανα για να επικοινωνούν μεταξύ τους και χαρακτηρίζεται από την ευκολία στην επεξεργασία της μουσικής, την ευελιξία και το μικρό μέγεθος των αρχείων. Το MIDI λειτουργεί όταν πατιέται ένα πλήκτρο από μία συσκευή ή αλλιώς MIDI device συνδεδεμένο με USB ή καλώδιο 5 pin και στέλνει ένα σήμα ή μία σειρά από οδηγίες εκτέλεσης σήματος στον υπολογιστή αντί του ηχογραφημένου ήχου και περιλαμβάνει πληροφορίες του σήματος όπως διάρκεια, τυχόν αλλαγές στην ένταση και τόνο.



Εικ. 1-8 Διάγραμμα διασύνδεσης ηλεκτρονικών μουσικών οργάνων με υπολογιστή

πηγή: https://simplified.us/blogs/news/what-is-midi?srltid=AfmBOoqn9k_470zCgvF5UGms7SM23IUysMn4maSf1X-fl4P_No4icB2L

1.8 DAW – Digital Audio Workstation

Το DAW (Digital Audio Workstation) είναι μία κατηγορία λογισμικού που χρησιμοποιείται στην ηχογράφηση, και στην δημιουργία και επεξεργασία αρχείων ήχων. Μπορεί να πάρει την μορφή είτε λογισμικού είτε σύνολο από ηλεκτρονικά μουσικά όργανα συνδεδεμένα σε έναν υπολογιστή. Εφαρμόζεται στην ηχογράφηση μουσικής, τραγουδιού, podcasts αλλά και ηχητικών εφέ. Κυκλοφορούν πάρα πολλά τέτοια προγράμματα στο εμπόριο όπως για παράδειγμα το FL Studio και Ableton αλλά και ανοιχτού κώδικα όπως το Audacity.



Εικ. 1-9 Audacity.

Πηγή: <https://www.audacityteam.org/>

1.9 Κωδικοποιητής ήχου

Ένας κωδικοποιητής ήχου είναι μια συσκευή ή ένα πρόγραμμα υπολογιστή ικανό να κωδικοποιεί ή να αποκωδικοποιεί ένα ψηφιακό ρεύμα δεδομένων (κωδικοποιητής) που κωδικοποιεί ή αποκωδικοποιεί ήχο. Στο λογισμικό, ένας κωδικοποιητής ήχου είναι ένα πρόγραμμα υπολογιστή που εφαρμόζει έναν αλγόριθμο που συμπιέζει και αποσυμπιέζει ψηφιακά δεδομένα ήχου σύμφωνα με ένα δεδομένο αρχείο ήχου ή μορφή κωδικοποίησης ήχου μέσω ροής. Ο στόχος του αλγορίθμου είναι να αναπαραστή το σήμα ήχου υψηλής πιστότητας με τον ελάχιστο αριθμό bit, διατηρώντας παράλληλα την ποιότητα. Αυτό μπορεί να μειώσει αποτελεσματικά τον αποθηκευτικό χώρο και το εύρος ζώνης που απαιτείται για τη μετάδοση του αποθηκευμένου αρχείου ήχου. Τα περισσότερα κωδικοποιητικά λογισμικού υλοποιούνται ως βιβλιοθήκες που διασυνδέονται με έναν ή περισσότερους πολυμεσικούς παίκτες. Οι περισσότεροι σύγχρονοι αλγόριθμοι συμπίεσης ήχου βασίζονται στην κωδικοποίηση τροποποιημένου διακριτού συνημιτόνου (MDCT) και στην γραμμική προγνωστική κωδικοποίηση (LPC).

Στο υλικό υπολογιστή, ο κωδικοποιητής ήχου αναφέρεται ως μια μεμονωμένη συσκευή που κωδικοποιεί αναλογικό ήχο σε ψηφιακά σήματα και αποκωδικοποιεί τα ψηφιακά σήματα ξανά σε αναλογικά. Με άλλα λόγια, περιέχει τόσο έναν αναλογικό-ψηφιακό μετατροπέα (ADC) όσο και έναν ψηφιακό-αναλογικό μετατροπέα (DAC) που λειτουργούν με το ίδιο σήμα ρολογιού. Χρησιμοποιείται, για παράδειγμα, σε κάρτες ήχου που υποστηρίζουν τόσο είσοδο όσο και έξοδο ήχου.

(Audio Codec, 2025)

1.10 SDK

Το SDK ή αλλιώς Software Development Kit αποτελείται από μια συλλογή εργαλείων ανάπτυξης λογισμικού συγκεντρωμένων σε ένα ενιαίο, εγκαταστάσιμο πακέτο. Διευκολύνουν τη δημιουργία εφαρμογών, καθώς περιλαμβάνουν μεταγλωττιστή, εργαλείο αποσφαλμάτωσης και, ενίοτε, πλαίσιο ανάπτυξης λογισμικού (software framework). Συνήθως, αφορούν συγκεκριμένους συνδυασμούς πλατφόρμας υλικού και λειτουργικού συστήματος. Για τη δημιουργία εφαρμογών με προηγμένες λειτουργίες, όπως η προβολή διαφημίσεων, οι ειδοποιήσεις push κ.ά., οι περισσότεροι προγραμματιστές εφαρμογών χρησιμοποιούν εξειδικευμένα πακέτα ανάπτυξης λογισμικού

1.11 Σύνοψη 1^{ου} κεφαλαίου

Σε αυτό το κεφάλαιο έγινε η περιγραφή των τεχνολογιών που έχουν σχέση με την διπλωματική εργασία με αναφορά στην ιστορία του ήχου στα ηλεκτρονικά παιχνίδια. Κάποιες από αυτές είναι ανερχόμενες και μπορούν να χρησιμοποιηθούν στο μέλλον

2 Σχεδιασμός

Στο κεφάλαιο αυτό γίνεται η ανάλυση του ρόλου των μηχανών παιχνιδιών στην διαδικασία μας ως ένα πρωτεύον εργαλείο μαζί με ορισμένα παραδείγματα δημοφιλών μηχανών παιχνιδιών. Στη συνέχεια περιγράφεται η σχέση μεταξύ ήχου με τις μηχανές παιχνιδιών με την χρήση διαφορετικών εργαλείων. Τέλος περιγράφεται η μεθοδολογία ανάπτυξης ενός παιχνιδιού.

2.1 Μηχανές Παιχνιδιών

Οι μηχανές παιχνιδιών (Game Engines) αποτελούν ένα ολοκληρωμένο και επαναχρησιμοποιήσιμο πλαίσιο λογισμικού που παρέχει απόδοση(Rendering), φυσική (Physics), τεχνητή νοημοσύνη, ήχο, κινούμενα σχέδια(Animation) και άλλα, παρέχοντας ένα ενοποιημένο περιβάλλον για την επιτάχυνση και την απλοποίηση της ανάπτυξης βιντεοπαιχνιδιών και όχι μόνο σε όλες τις γλώσσες και πλατφόρμες για παράδειγμα το Unity και το Unreal Engine. (Arm, 2021)

2.1.1 Unity

Το Unity είναι μια από τις πιο διάσημες μηχανές παιχνιδιών στον κόσμο λόγω ότι είναι προσβάσιμη, ευκολία στην χρήση με εργαλεία οπτικού κώδικα (visual scripting tools) όπως το bolt και υποστηρίζει την αντικειμενοστραφής γλώσσα προγραμματισμού C#, μεγάλο αριθμό χρηστών (Userbase) και δωρεάν πρόσβαση μέσω του πλάνου “Personal”. Επίσης παρέχει υποστήριξη σε πάρα πολλές πλατφόρμες όπως κινητά (Android και iOS), κονσόλες (Playstation 5, Xbox series S/X και Nintendo Switch 2) και συσκευές εικονικής πραγματικότητας (Meta Quest και Valve Index VR). Η Unity παρέχει ένα μεγάλο αριθμό δυνατοτήτων και πρόσθετων (Packages) όπως μηχανές φυσικής (physics engine), εργαλεία σχεδιασμού πιστών (Probuilder) και μηχανές ήχου (wwise, fmod).

2.1.2 Unreal Engine

Το Unreal Engine της Epic Games είναι επίσης δημοφιλή μηχανή παιχνιδιών λόγω ότι είναι δωρεάν στην χρήση υπό έναν όρο ότι αν το παιχνίδι συγκεντρώνει κέρδη ενός εκατομμυρίου, παίρνει προμήθεια 5%, παρέχει ρεαλιστικά γραφικά, φωτισμό και υφές παράγοντας υψηλής πιστότητάς φωτογενή αποτελέσματα, χρησιμοποιείται και στην αρχιτεκτονική και στον σχεδιασμό και παρέχει τα Blueprints που επιτρέπουν στην ανάπτυξη παιχνιδιών χωρίς κώδικα. Αρκετά παιχνίδια που έχουν αναπτυχθεί με το Unreal Engine έχουν σημειώσει εμπορική επιτυχία όπως το Fortnite, το Hogwarts Legacy και το black myth wukong.

Να αναφέρουμε ότι οι δύο παραπάνω μηχανές εξυπηρετούν διαφορετικές ανάγκες όπως για πιο ‘δυνατούς τίτλους’ και καλύτερη πιστότητα καλύπτει το Unreal Engine ενώ για prototypes, εύκολη εισαγωγή στην ανάπτυξη παιχνιδιών και πιο ελαφρείς τίτλους καλύπτει το Unity.

Ο λόγος που επιλέχθηκε το Unity είναι ότι είναι από τις πιο διαδεδομένες και προσιτές μηχανές παιχνιδιών στον κόσμο με μεγάλη κοινότητα προγραμματιστών που μπορεί να μας βοηθήσουν

σε κάθε πρόβλημα που προκύψει και μεγάλη συμβατότητα με διάφορες πρόσθετες τεχνολογίες που βοηθούν στην κάλυψη των αναγκών μας στα πλαίσια της διπλωματικής εργασίας μας.

2.2 Ενδιάμεσο λογισμικό ήχου (Middleware)

Το ενδιάμεσο λογισμικό ήχου ή αλλιώς Audio Middleware είναι λογισμικό τρίτου που γεφυρώνει το χάσμα μεταξύ μηχανών παιχνιδιών όπως το Unity με το υλισμικό ήχου (Audio Hardware). Επιτρέπει τη δημιουργία και μίξη ήχων, άμεση εφαρμογή σύνθετων, δυναμικών ηχοτοπίων, και σύνθεση διαδραστικού ήχου μειώνοντας την εξάρτηση από τους προγραμματιστές. Επίσης βοηθά στη διαχείριση του προϋπολογισμού των δεδομένων ήχου παρέχοντας προηγμένες επιλογές συμπίεσης και μορφοποίησης, μειώνοντας το φόρτο εργασίας στην CPU και τη μνήμη.

2.2.1 Wwise

Το Wwise είναι ένα πρόγραμμα της εταιρίας Audiokinetic που δραστηριοποιείται στον διαδραστικό ήχο και προσφέρει επεξεργασία ήχου σε πραγματικό χρόνο, εντός παιχνιδιού, προηγμένος χωρικός ήχος, διαδραστικά μουσικά συστήματα και βαθιά ενσωμάτωση της με μηχανές παιχνιδιών. Υποστηρίζει όλες τις σύγχρονες κονσόλες βιντεοπαιχνιδιών και μπορεί να συνδεθεί με μηχανές παιχνιδιών (Game Engines).

2.2.2 fmod

Το fmod είναι ένα άλλο αντίστοιχο λογισμικό που κάνει επεξεργασία ήχου για παιχνίδια. Είναι εύκολο στην χρήση, μπορεί να κάνει μίξη σε πραγματικό χρόνο με άλλους ήχους παιχνιδιών αλλά υποστηρίζει μόνο τις δύο γνωστές μηχανές παιχνιδιών (Unity και Unreal Engine).



Εικ. 2-1 Παράθυρο του λογισμικού fmod

Πηγή: <https://www.fmod.com/>

2.2.3 Atmoky



Εικ. 2-2 Λογότυπο atmoky trueSpatial

Πηγή: <https://atmoky.com/products/ears/>

Το atmoky αποτελεί μία ολοκληρωμένη σουίτα σύνθεσης που χρησιμοποιείται στην σύνθεση αμφιωτικών ήχων και δημιουργία ρεαλιστικών περιβάλλοντων ήχου για παιχνίδια. Μπορεί να ενσωματωθεί στις μηχανές παιχνιδιών Unity και Unreal Engine και υποστηρίζει τις σύγχρονες κονσόλες παιχνιδιών.

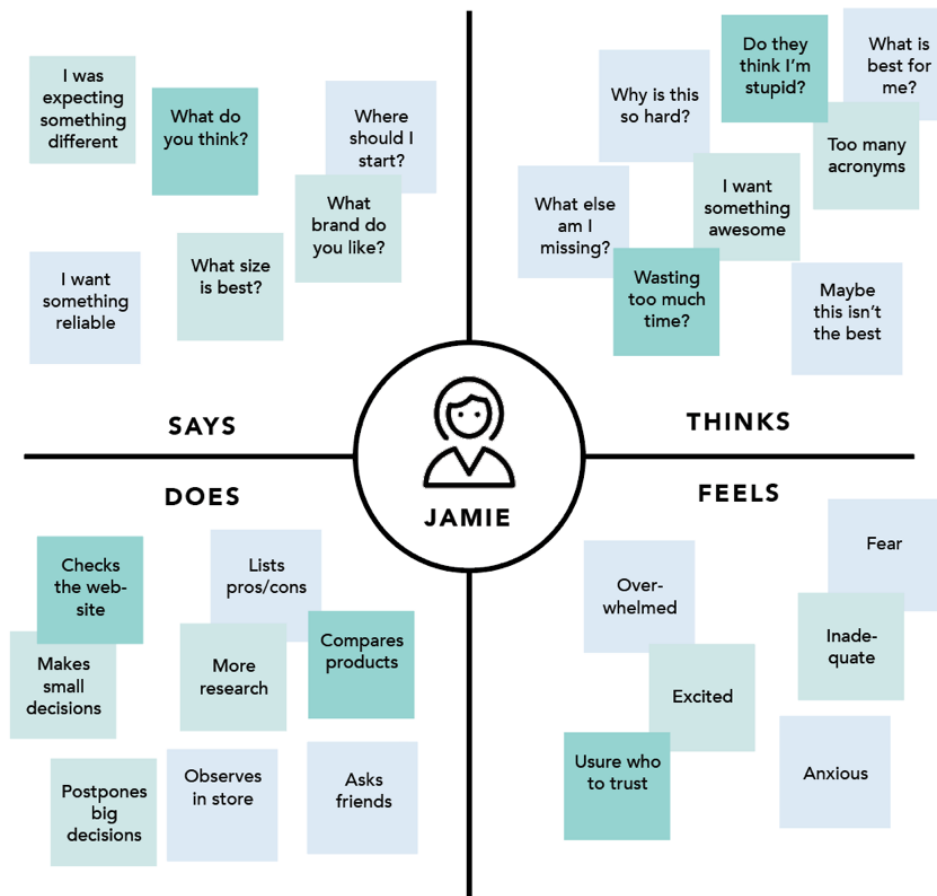
Η διαφορά είναι ότι το fmod έχει πιο απλή διεπαφή χρήστη που μοιάζει με DAW και απευθύνεται σε μικρές ομάδες (indie) ενώ το Wwise έχει πιο εξειδικευμένα παράθυρα εργασίας με σύστημα ιεράρχησης που καθιστά πιο κατάλληλο για μεγάλες ομάδες ανάπτυξης παιχνιδιών.

2.3 Χάρτες εμπειρίας χρήστη (UX Maps)

Οι χάρτες UX (UX Maps) περιγράφουν διάφορες πτυχές και διαδικασίες που σχετίζονται με την εμπειρία του χρήστη σχετικά με μία υπηρεσία ή ένα προϊόν, δείχνοντας τις πράξεις, τις ενέργειες και τα συναισθήματα του χρήστη σε κάθε στάδιο. παρουσιάζοντας τις . Παρακάτω δίνονται οι περιγραφές των τεσσάρων τύπων χάρτων UI.

Χάρτης εμπάθειας (**Empathy Map**): Καταγράφει τι ξέρουμε για τον τύπο χρήστη ενός προϊόντος για να δημιουργήσουν μια κοινή αντίληψη και να βοηθήσουμε στη λήψη αποφάσεων. Χωρίζεται σε τέσσερα τεταρτημόρια, τι σκέφτεται, τι λέει, τι κάνει και τι νιώθει.

EMPATHY MAP Example (Buying a TV)



NNGROUP.COM **NN/g**

Εικ. 2-1 Παράδειγμα διαγράμματος χάρτη εμπάθειας (Empathy Map)

Πηγή: <https://www.nngroup.com/articles/ux-mapping-cheat-sheet/>

Χάρτης ταξιδιού πελάτη (**Customer Journey Map**): εστιάζει στην αλληλεπίδραση ενός συγκεκριμένου πελάτη με ένα προϊόν ή μια υπηρεσία. Καταγράφει μία σειρά από στόχους σε χρονολογική μορφή. Χωρίζεται σε τέσσερις στήλες: φάσεις, δράσεις, σκέψεις και συναισθήματα.

CUSTOMER JOURNEY MAP *Example (Switching Mobile Plans)*

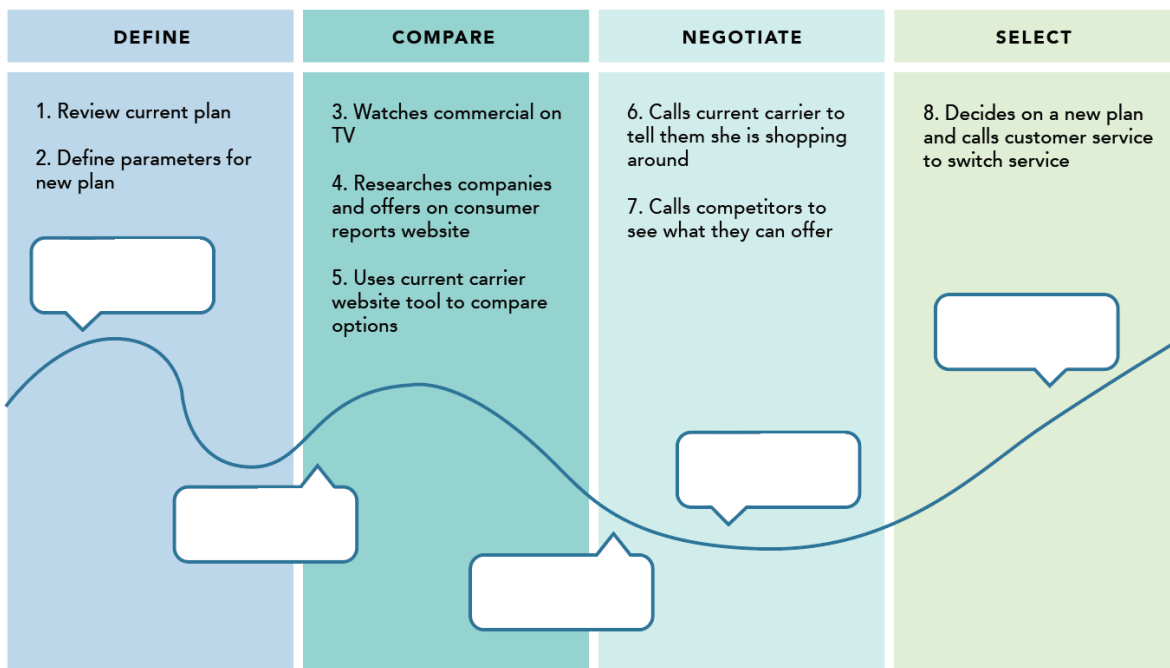


JAMIE

Scenario: Jamie needs to switch her current mobile plan. She wants a plan that can save her money without having to sacrifice usage limits.

EXPECTATIONS

- Clear online information
- Ability to compare plan breakdowns
- Friendly and helpful customer support



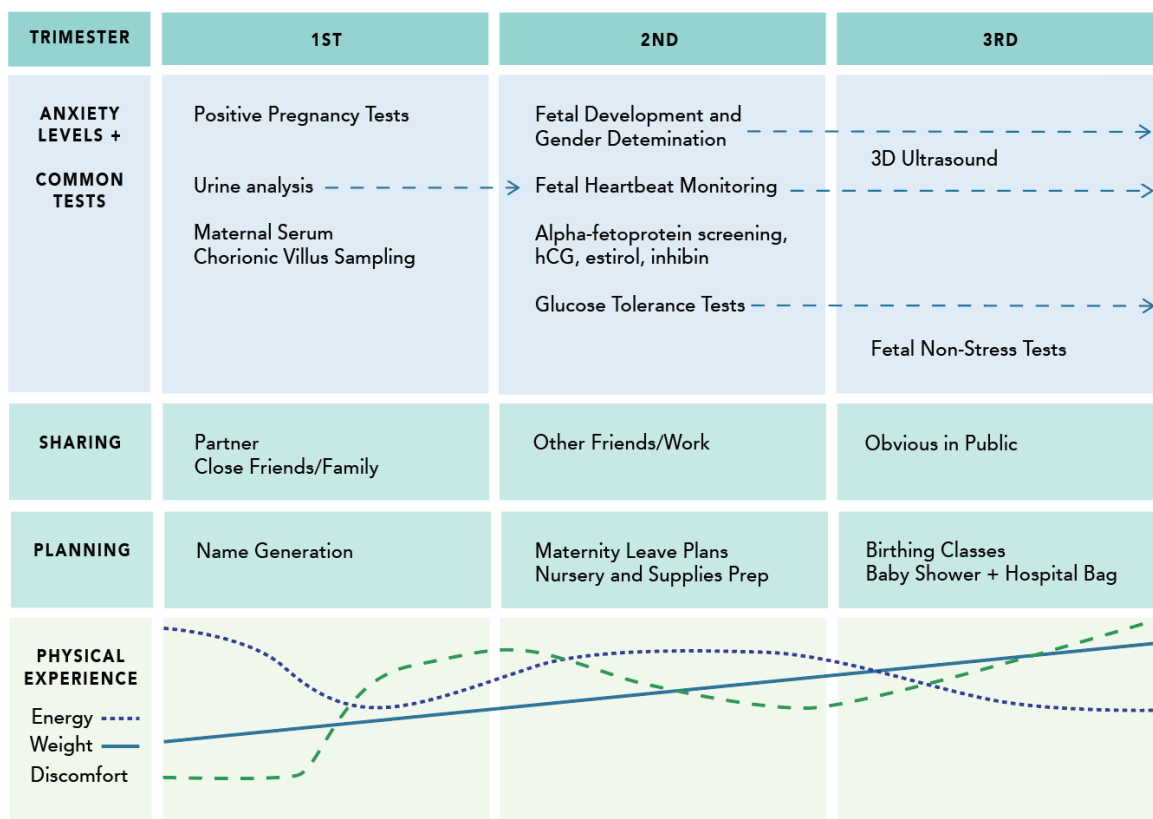
NNGROUP.COM **NN/g**

Εικ. 2-2 Παράδειγμα διαγράμματος χάρτη ταξιδιού πελάτη (Customer Journey Map)

Πηγή: <https://www.nngroup.com/articles/ux-mapping-cheat-sheet/>

Χάρτες εμπειρίας (**Experience Map**): Οπτικοποιεί την όλη διαδικασία από άκρη σε άκρη όπου ένας 'γενικός' πελάτης περνάει για να ολοκληρώσει έναν στόχο. Η εμπειρία είναι ανεξάρτητη από ένα συγκεκριμένο προϊόν ή επιχείρηση. Χρησιμοποιείται για την κατανόηση της γενικής ανθρώπινης συμπεριφοράς.

EXPERIENCE MAP Example (Pregnancy)



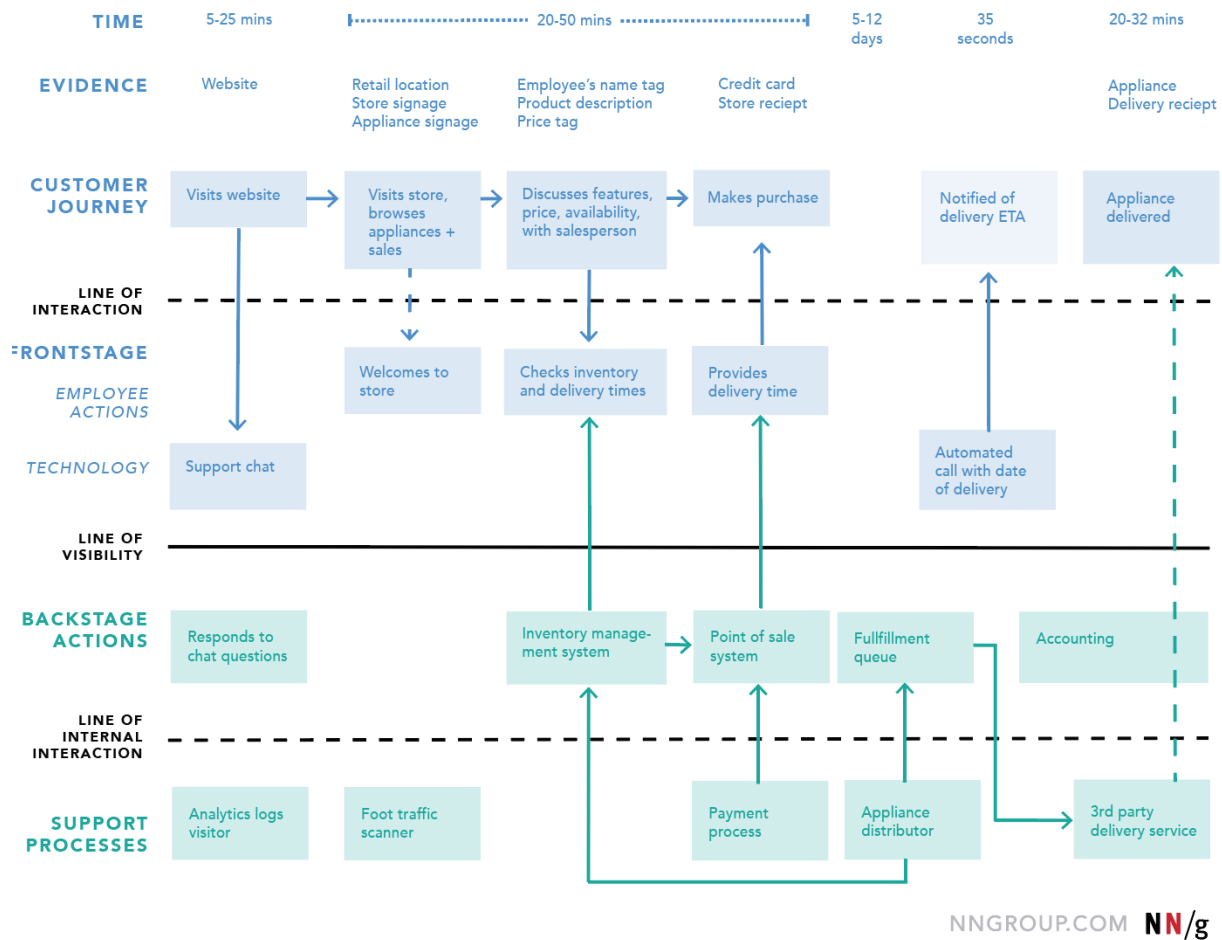
NNGROUP.COM **NN/g**

Εικ. 2-3 Παράδειγμα διαγράμματος χάρτη εμπειρίας (Experience Map)

Πηγή: <https://www.nngroup.com/articles/ux-mapping-cheat-sheet/>

Χάρτης υπηρεσιών (**Service Blueprints**): Είναι παρόμοιοι με τον χάρτη εμπειρίας αλλά καλύπτουν σύνθετα σενάρια που καλύπτουν πολλά σημεία επαφής. Το διάγραμμα απεικονίζει μία αλληλουχία χρόνου μεταξύ εργασιών που είναι ιεραρχικές. Χωρίζεται σε τέσσερις οριζόντιες στήλες: ενέργειες πελατών, ενέργειες στο προσκήνιο, ενέργειες στα παρασκήνια και διαδικασίες υποστήριξης. (Gibbons, 2017)

SERVICE BLUEPRINT Example (Appliance Retailer)



Εικ. 2-4 Παράδειγμα διαγράμματος υπηρεσίας (Service Blueprint)

Πηγή: <https://www.nngroup.com/articles/ux-mapping-cheat-sheet/>

2.4 Ανάλυση ακροατηρίου

Η ανάλυση ακροατηρίου περιλαμβάνει τον προσδιορισμό του κοινού και την προσαρμογή μιας ομιλίας στα ενδιαφέροντα, το επίπεδο κατανόησης, τις στάσεις και τις πεποιθήσεις στο κοινό. Η υιοθέτηση μιας προσέγγισης που επικεντρώνεται στο κοινό είναι σημαντική, διότι η αποτελεσματικότητα ενός ομιλητή θα βελτιωθεί εάν η παρουσίαση δημιουργηθεί και παραδοθεί με τον κατάλληλο τρόπο. Ο προσδιορισμός του κοινού μέσω εκτεταμένης έρευνας είναι συχνά δύσκολος, οπότε η προσαρμογή στο κοινό βασίζεται συχνά στην υγιή χρήση της φαντασίας.

Όπως συμβαίνει με πολλά πολύτιμα εργαλεία, η ανάλυση του κοινού μπορεί να χρησιμοποιηθεί σε υπερβολικό βαθμό. Η προσαρμογή μιας ομιλίας στο κοινό δεν είναι το ίδιο με το να λέτε

απλά στο κοινό αυτό που θέλει να ακούσει. Η ανάλυση του κοινού δεν σημαίνει «επιδείκνυση» ή «υποκλίσεις» στο κοινό. Αντίθετα, η προσαρμογή καθοδηγεί τις επιλογές στυλ και περιεχομένου που κάνει ένας ομιλητής για μια παρουσίαση. Η προσαρμογή στο κοινό συχνά συνεπάγεται μια πολύ λεπτή γραμμή μεταξύ υπερβολικής και ανεπαρκούς προσαρμογής – μια διάκριση που μπορεί να εκτιμηθεί καλύτερα με την κατανόηση των γενικών στοιχείων αυτής της δεξιότητας. Οι ρόλοι που παίζουν στην ανάλυση ακροατηρίου είναι οι προσδοκίες του κοινού, η γνώση σε ένα θέμα, η προσέγγιση προς σε ένα θέμα, το μέγεθος κοινού, το περιβάλλον διεξαγωγής, τα δημογραφικά και ο εγωκεντρισμός του κοινού. (Audience Analysis, χ.χ.)

2.5 Κύκλος ανάπτυξης παιχνιδιού.

Ένα παιχνίδι παίρνει πολλά χρόνια και απασχολεί μεγάλες ομάδες και επιτυγχάνεται με πολύ μεγάλα κόστη (budget). Αντίστοιχες μεθοδολογίες που χρησιμοποιούνται σε περίπλοκο λογισμικό χρησιμοποιείται και σε παιχνίδια λόγω της τεχνικής περιπλοκότητας τους.

Αποτελείται κυρίως από τρία στάδια, της Προ-παραγωγής, Παραγωγής και μετά την παραγωγή:

- Προπαραγωγής(Pre-production): Εκεί καταστρώνονται ιδέες και σχέδια, ξεκινάει η τεκμηρίωση, ομάδες συστήνονται και καθορίζονται το κόστος(budget) και χρονοδιάγραμμα
- Παραγωγής (Production): Το μεγαλύτερο στάδιο σε διάρκεια. Εκεί οι ιδέες πραγματοποιούνται μετά από διαρκής ανάπτυξη, σχεδιασμό και βελτίωση.
- Μετά την παραγωγή (Post-Production): Γίνεται η προώθηση και η συνεχής υποστήριξη του παιχνιδιού.

Και πιο αναλυτικά:

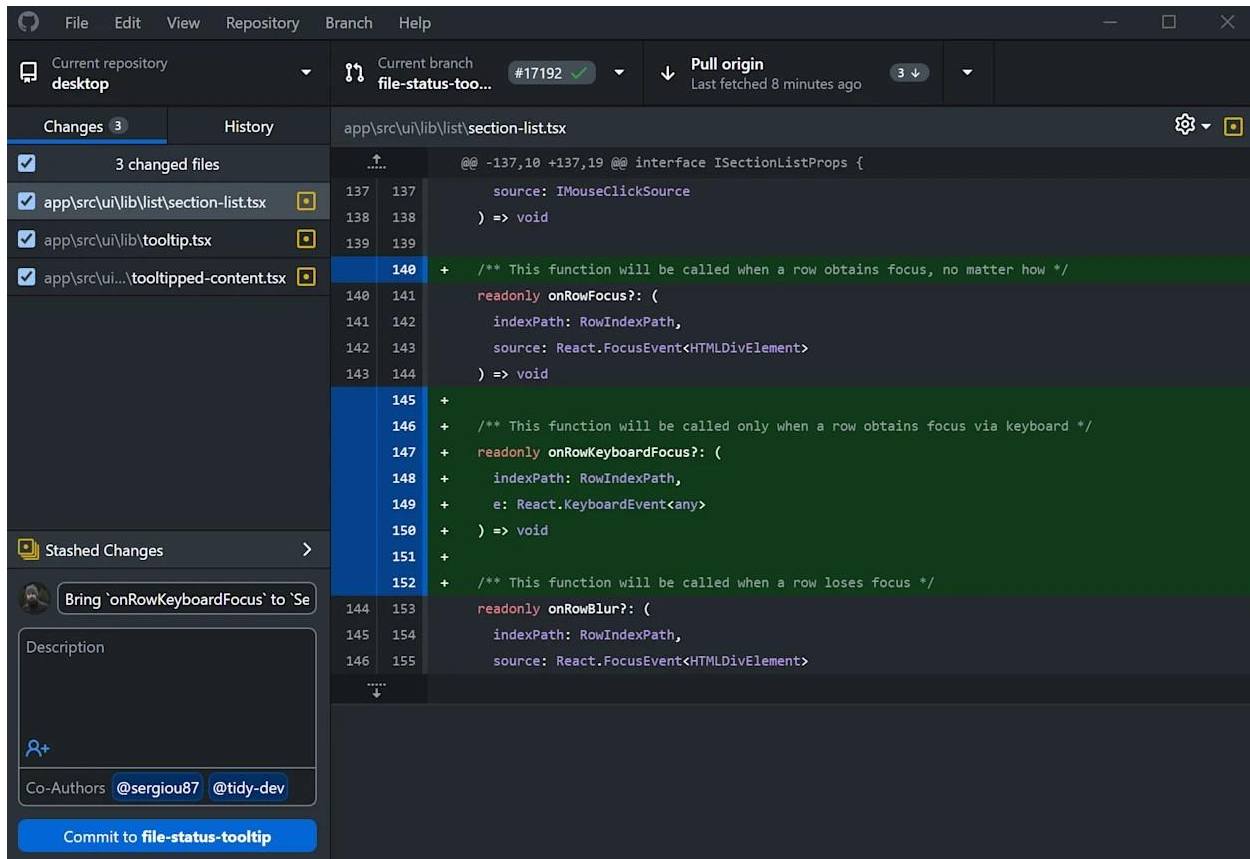
- Σχεδιασμού (Planning): Από τα πιο σημαντικά στάδια. Πραγματοποιούνται σχέδια για την κυκλοφορία του παιχνιδιού και αξιολογούνται ιδέες σχετικά με το παιχνίδι και γίνεται η σύσταση των ομάδων ανάπτυξης.
- Προ-παραγωγής (Preproduction): Γίνεται ο αρχικός σχεδιασμός του παιχνιδιού και η δημιουργία προσχεδίων σε διάφορα επιμερή στοιχεία όπως την ιστορία του και πρωτότυπα και αποφασίζονται οι ιδέες που θα εφαρμοστούν.
- Παραγωγή (Production): Το πιο εκτενές στάδιο στην ανάπτυξη ενός παιχνιδιού. Γίνεται η σύνταξη του κώδικα, η γραφή της πλοκής, η δημιουργία των οπτικών στοιχείων, η σύνθεση των ήχων και της μουσικής στο παιχνίδι και η ηχογράφηση των γραμμών των χαρακτήρων.
- Δοκιμές (Testing): Αποτελείται από ένα εκτενές στάδιο που γίνονται οι δοκιμές στο παιχνίδι είτε στο λειτουργικό τομέα είτε στον οπτικό τομέα είτε στην εκτέλεση του.
- Πριν την κυκλοφορία (Prelaunch): Γίνεται η προώθηση του μέσω demo, trailer και παρουσιάσεις και επιδείξεις σε εκδηλώσεις.

- Λανσάρισμα (Launching): Το παιχνίδι βρίσκεται κοντά στην ημερομηνία διάθεσης και γίνεται το τελικό γυάλισμα, διορθώνονται τυχόν bugs ή λάθη στον κώδικα και μπαίνουν οι τελικές πινελιές στα οπτικά στοιχεία του παιχνιδιού
- Μετά την κυκλοφορία: (Post Launch). Μπαίνει πρόσθετο περιεχόμενο είτε μέσω αναβαθμίσεων, πρόσθετων (DLC), επεκτείνεται η υποστήριξη του και γίνονται επιπλέον διορθώσεις και προσθήκες. (GameMaker, 2023)

2.6 Εργαλεία διανομής κώδικα

Τα εργαλεία διανομής κώδικα αποτελούνται από κατανεμημένα συστήματα ελέγχου εκδόσεων, σχεδιασμένα να χειρίζεται τα πάντα, από μικρά έως πολύ μεγάλα έργα κώδικα, με ταχύτητα και αποτελεσματικότητα. Τα δύο κύρια συστήματα ελέγχου έκδοσης κώδικα είναι το Git και το Unity Version Control.

Το Git είναι το κύριο σύστημα διαχείρισης έκδοσης κώδικα ανοιχτού κώδικα που χρησιμοποιείται στην παρακολούθηση αλλαγών και διορθώσεων του κώδικα (codebase) σε χρονική σειρά, την σύγκριση και ενσωμάτωση διαφόρων αλλαγών σε παρακλάδια του κώδικα. Το Github είναι η πιο διάσημη πλατφόρμα που κάνει την ελεύθερη διανομή, την συνεργασία με άλλους για την σύνταξη κώδικα και την παρακολούθηση αλλαγών. Βασίζεται στο Git. Διατίθεται και ως πρόγραμμα υπολογιστή (Github Desktop) που διευκολύνει τον χειρισμό του Git.

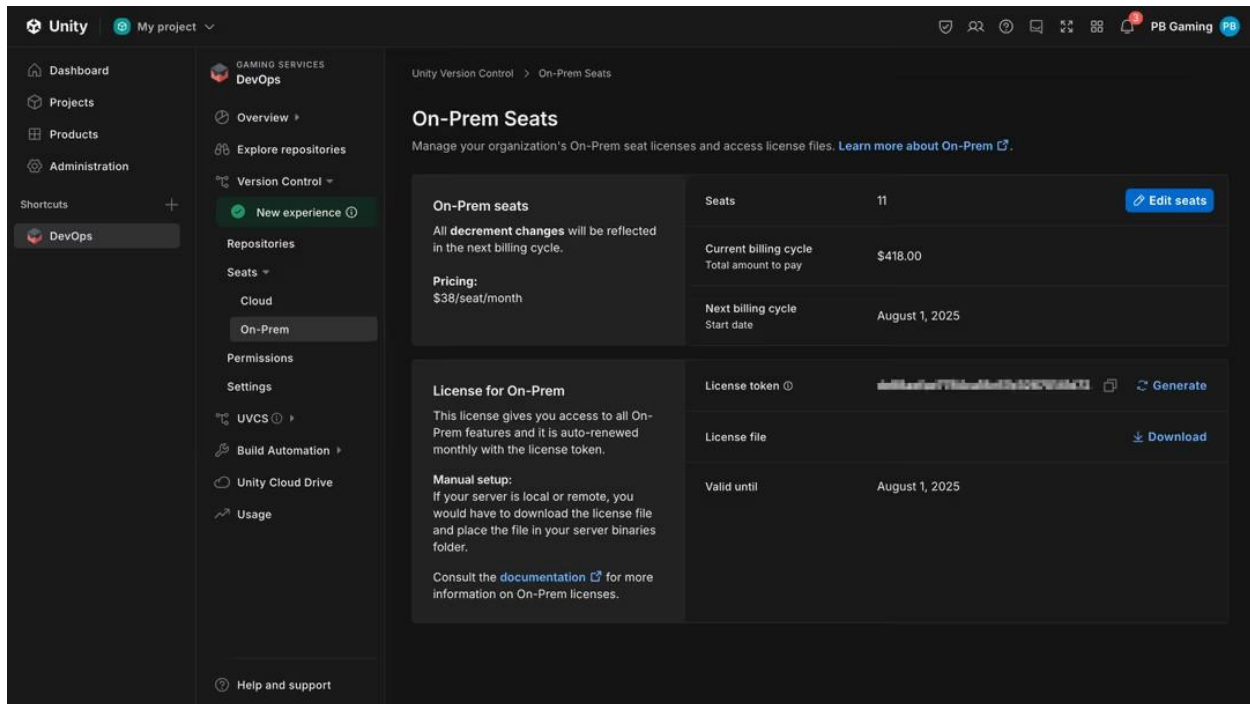


Εικ. 2-5 Παράθυρο Github Desktop

πηγή: <https://github.com/apps/desktop>

Το Unity version control είναι το ενσωματωμένο εργαλείο κατανημμένου ελέγχου έκδοσης κώδικα που αναπτύχθηκε από την Códice Software και τρέχει σε όλα τα λειτουργικά συστήματα (Cross-platform). Περιλαμβάνει διάφορα συστατικά όπως γραμμή εντολών, περιβάλλον χρήση web και σέρβερ. Μπορεί να ενσωματωθεί σε άλλες μηχανές παιχνιδιών, να φορτώσει γρήγορα αρχεία και μεγάλες δεξαμενές (repos) κώδικα και να συνεργαστεί με άλλους προγραμματιστές μέσω εργαλείων devops.

(Unity, 2026)



Εικ. 2-6 Παράθυρο Unity DevOps

πηγή: <https://unity.com/features/version-control>

2.7 Σύνοψη 2^{ου} κεφαλαίου

Σε αυτό το κεφάλαιο έγινε η περιγραφή και η σύγκριση των διαφόρων προγραμμάτων λογισμικού και μεθοδολογιών που χρησιμοποιούνται στην ανάπτυξη των παιχνιδιών. Επιλέχθηκε το Unity για την ευκολία του και την πληθώρα των πρόσθετων που βοηθούν στην ανάπτυξη του παιχνιδιού και το Audiokinetic Wwise γιατί χρησιμοποιείται στην βιομηχανία (Industry Standard) και έχει πληθώρα εργαλείων.

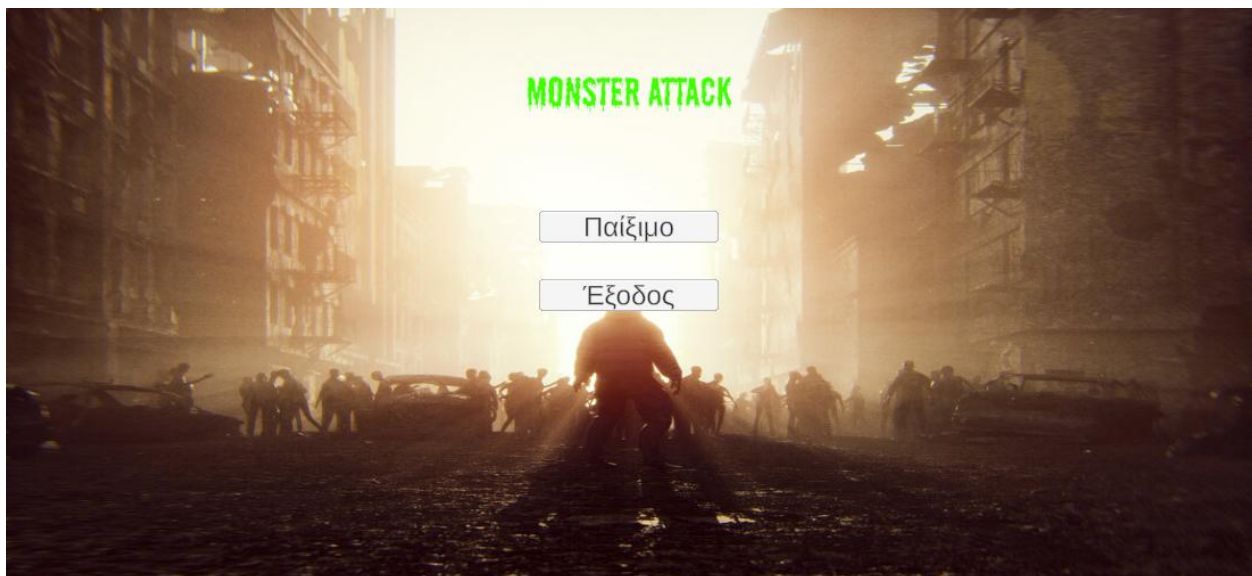
3 Ανάπτυξη παιχνιδιού

Το παιχνίδι που θα φτιάξουμε θα είναι εμπνευσμένο από τα top down (Προβολή κάτοψης) shooter όπως για παράδειγμα το Brotato που ο χαρακτήρας θα έχει ένα όπλο και θα εξουδετερώνει εχθρούς όπου ο ήχος ακούγεται ανάλογα με την κατεύθυνσή της σφαίρας. Στο κεφάλαιο αυτό θα γίνει η περιγραφή των χαρακτηριστικών των επιμέρους εργαλείων και η διαδικασία κατά την ανάπτυξη του παιχνιδιού συνοδευόμενα από Screenshots και πίνακες περιγραφής των συναρτήσεων και των συστατικών για καλύτερη κατανόηση.



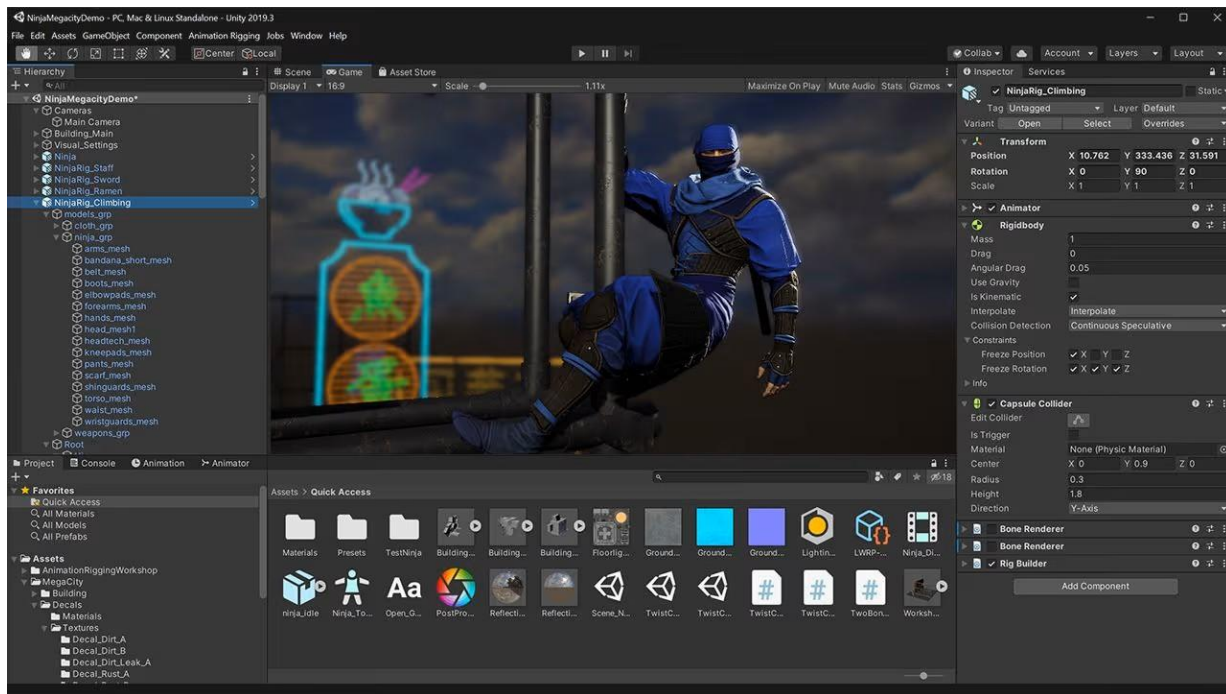
Εικ. 3-1 Brotato

πηγή: <https://store.steampowered.com/app/1942280/Brotato/>



Εικ. 3-2 Αρχική οθόνη παιχνιδιού

3.1 Unity



Εικ. 3-3 Παράθυρο περιβάλλοντος Unity

Πηγή: <https://unity.com/releases/2019-3/editor-tools>

3.1.1 Αντικείμενα παιχνιδιού (GameObjects)

Τα αντικείμενα παιχνιδιού στο Unity αναπαριστούν διάφορες οντότητες αποτελούμενες από συστατικά (Components) ακόμη και λειτουργίες κώδικα μη ορατές στο παιχνίδι μέσω του MonoBehaviour. Προβάλλονται μέσω της σκηνής (Scene).

Επίσης τα Prefabs χρησιμοποιούνται ως κεντρικό αντικείμενο για να οριστεί το αρχικό αντικείμενο που τα παράγωγα του μπορούν να γίνουν επεξεργασία προσθέτοντας νέες ιδιότητες ή αλλάζοντας τις παραμέτρους του. Οι αλλαγές μπορούν και να γυρίσουν στο παράγωγο αντικείμενο.

Τα αντικείμενα που είναι συγκεντρωμένα στην καρτέλα Project του Unity Editor λέγονται και Assets και συνήθως οργανώνονται με βάση το είδος της πχ Κώδικας, μοντέλα, αρχεία ήχου.

3.1.2 Δομή κώδικα Unity

Η μηχανή παιχνιδιών Unity είναι γραμμένη στην γλώσσα προγραμματισμού C++ όπου χαρακτηρίζεται από άμεση επικοινωνία με το hardware, μεγαλύτερο έλεγχο στην επίδοση του παιχνιδιού και γρήγορο χρόνο εκτέλεσης(runtime) αλλά το μειονέκτημα του είναι η δυσκολία στην εγκατάσταση ενός διερμηνέα (compiler), ενώ η διεπαφή χρήστη είναι γραμμένη σε C# που

χαρακτηρίζεται από την ευκολία στην εκμάθηση της, μεγαλύτερη κοινότητα προγραμματιστών και μεγαλύτερη ευελιξία

Οι πιο σημαντικές συναρτήσεις είναι το Awake, Start και Update.

Το void Awake() χρησιμοποιείται στην αρχικοποίηση των μεταβλητών πριν καν ξεκινήσει

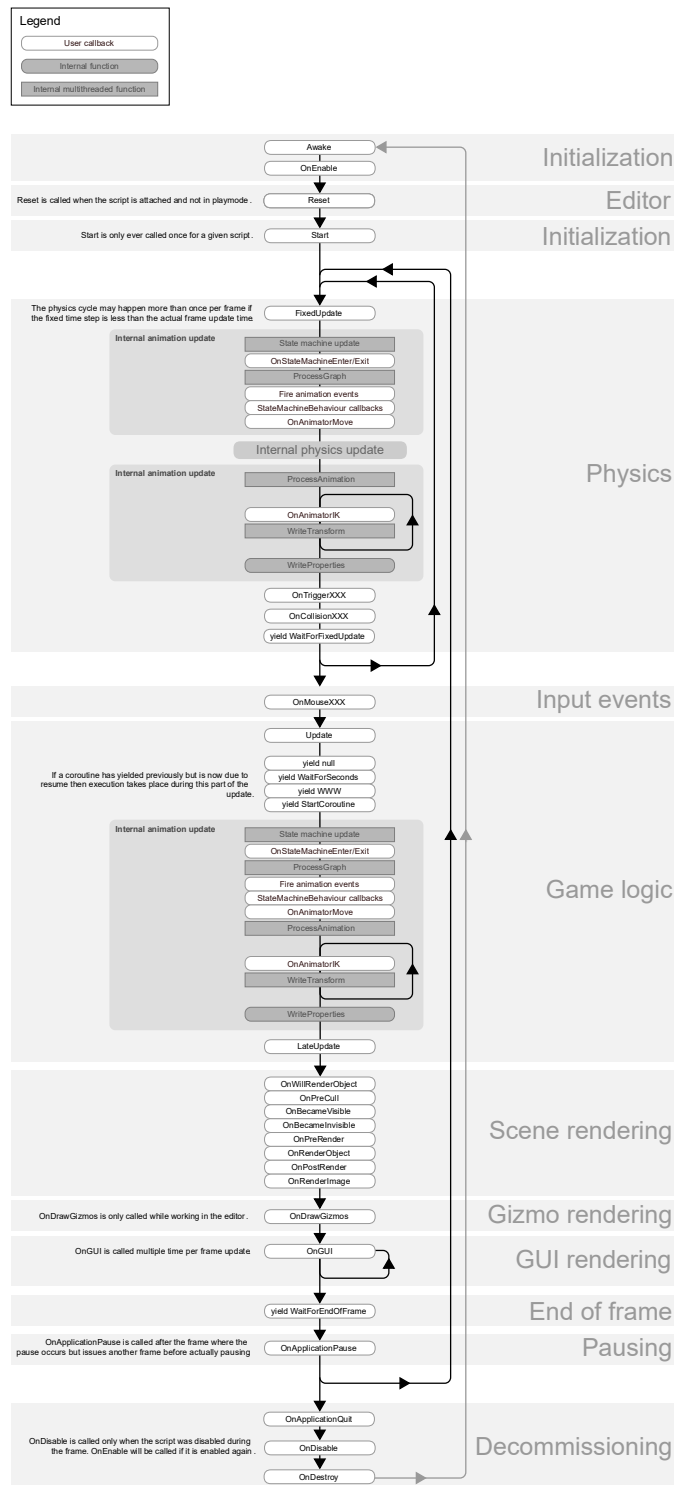
Το void Start() τρέχει μόνο μία φορά πριν από το πρώτο καρέ

Το void Update() τρέχει κάθε δευτερόλεπτο της διάρκειας του παιχνιδιού.

Το void FixedUpdate() χρησιμοποιείται στους υπολογισμούς της φυσικής των αντικειμένων παιχνιδιού και μπορεί να καλέσει πολλές φορές ανά καρέ δευτερολέπτου καθιστώντας κατάλληλο για υπολογισμούς φυσικής που πρέπει να παραμείνουν σταθεροί και ντετερμινιστικοί σε σύγκριση με την συνάρτηση Update() που καλείται μόνο μία φορά για κάθε καρέ δευτερολέπτου.

Τα συστατικά παιχνιδιού (Game Components) λειτουργούν με τον ίδιο τρόπο όπως καλούμε ένα αντικείμενο σε μία αντικειμενοστραφής γλώσσα προγραμματισμού.

Τα αντικείμενα στο Unity βασίζονται στο αντικείμενο MonoBehaviour που περιγράφει τον κύκλο ζωής τους όπως φαίνεται στο διάγραμμα παρακάτω.



Σχήμα 3-1 Κύκλος ζωής κώδικα MonoBehaviour

Επίσης υπάρχει η δυνατότητα οπτικού κώδικα (Visual Scripting) αλλά το μειονέκτημα του είναι αργό σε επίδοση σε σχέση με την σύνταξη κώδικα και δεν προτείνεται για την διπλωματική εργασία.

(Unity, 2026)

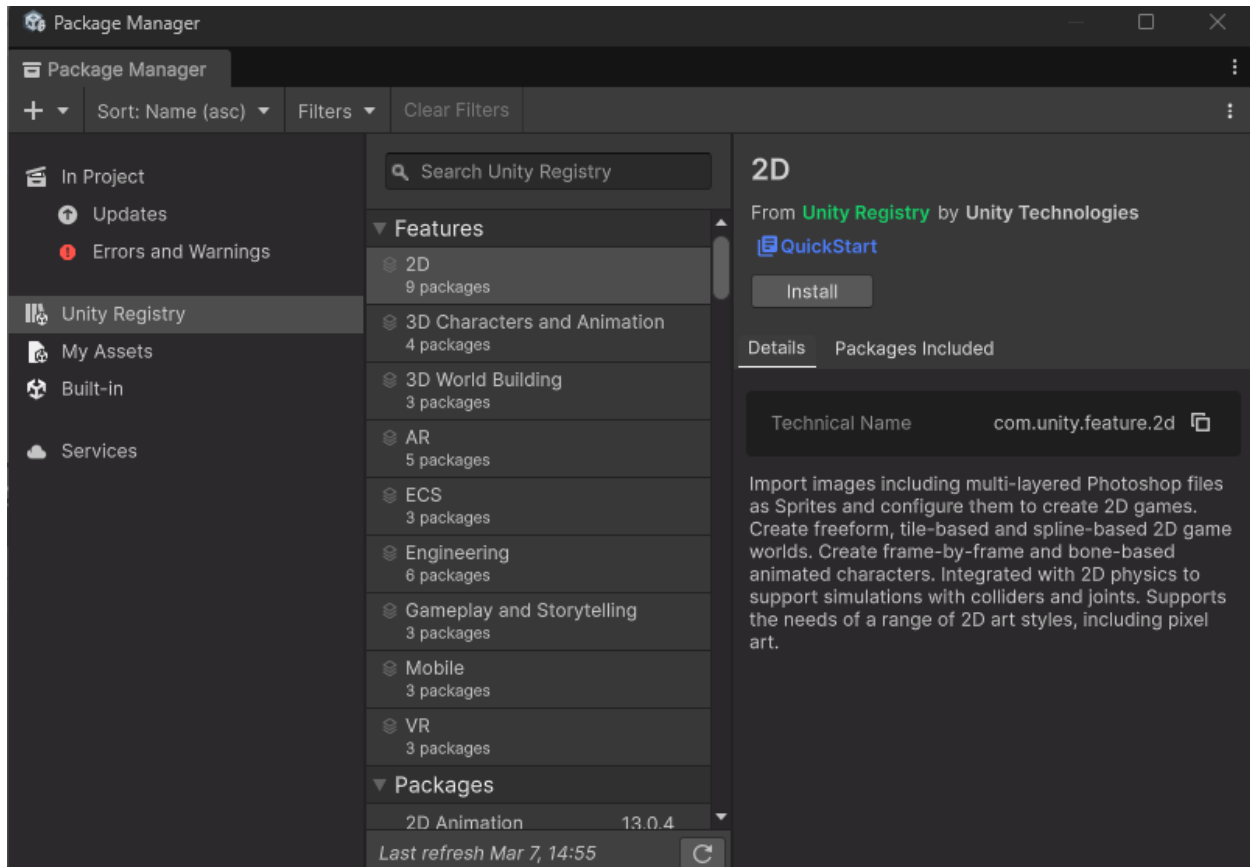
3.1.3 Αντικείμενα (Assets) παιχνιδιού

Το Unity Asset Store παρέχει αντικείμενα παιχνιδιού δωρεάν και επί πληρωμή χρήσιμα για την εκπόνηση της διπλωματικής μας εργασίας. Ένα από αυτά είναι το Polygon City Pack της synthy που περιλαμβάνει χαρακτήρες και αντικείμενα χαμηλής ανάλυσης πολύγωνων που αποτελεί το κύριο πακέτο αντικειμένων για τον κόσμο του παιχνιδιού. Τα animations μπορούν να ενσωματωθούν εύκολα σε αυτούς του χαρακτήρες. Η διαχείριση τους γίνεται μέσα από το παράθυρο Project Manager που παρέχει και επιπλέον ειδικά εργαλεία μέσω του Unity Register ενώ τα αποθηκευμένα Assets βρίσκονται στο My Assets.



Εικ. 3-4 synty polygon city pack

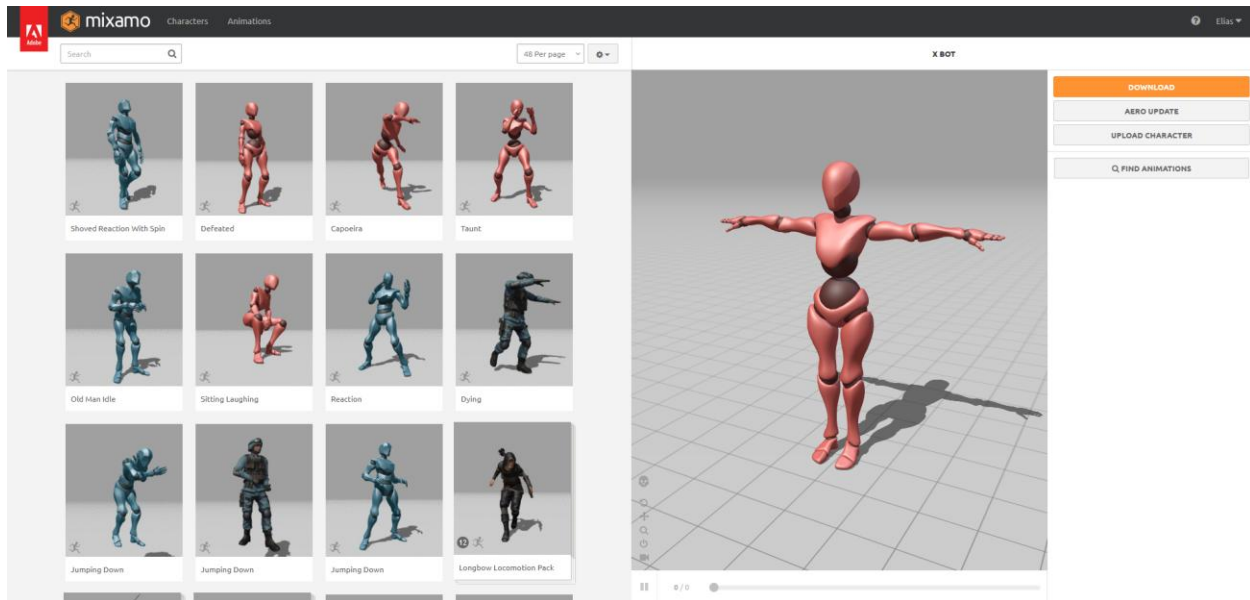
πηγή: <https://assetstore.unity.com/packages/3d/environments/urban/polygon-city-pack-art-by-synty-95214>



Εικ. 3-5 Παράθυρο Package Manager

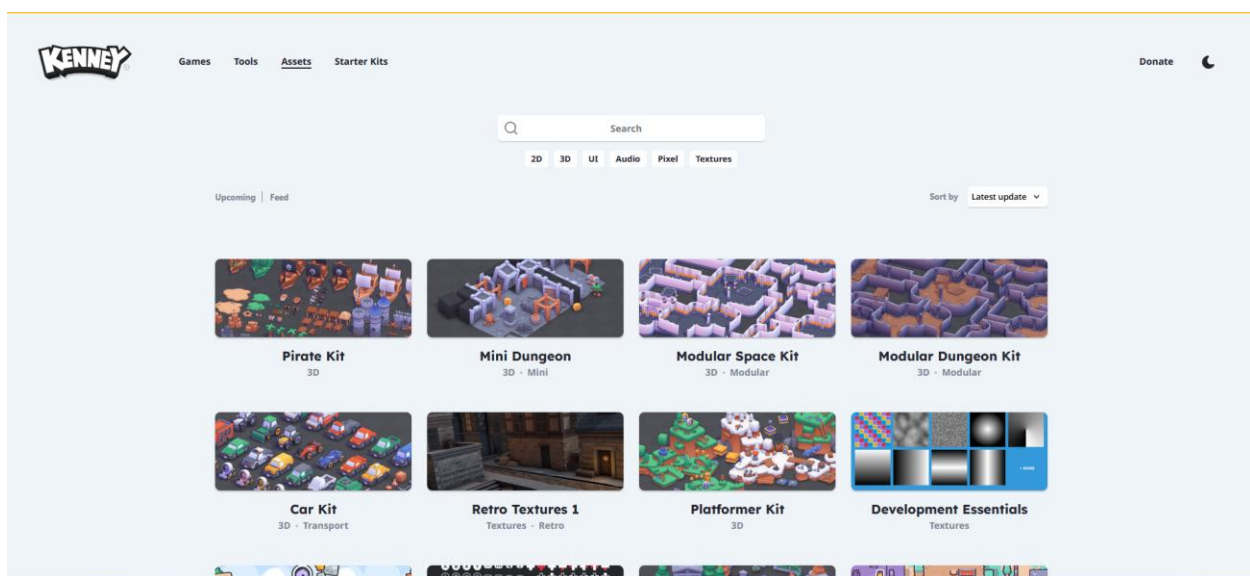
Η ιστοσελίδα kenney.nl/assets παρέχει έτοιμα δισδιάστατα και τρισδιάστατα μοντέλα έτοιμα προς χρήση σε οποιαδήποτε μηχανή παιχνιδιών δωρεάν το οποίο μας διευκολύνει κατά πολύ την διαδικασία ανάπτυξης παιχνιδιού.

Για την επιλογή των Animation, χρησιμοποιήθηκε το mixamo το οποίο παρέχει διάφορες επιλογές σχετικά με την κίνηση, τις σκηνές καρέ(framerates), τον τύπο αρχείου και την συμπερίληψη ή μη του δέρματος για το mockup(skin).



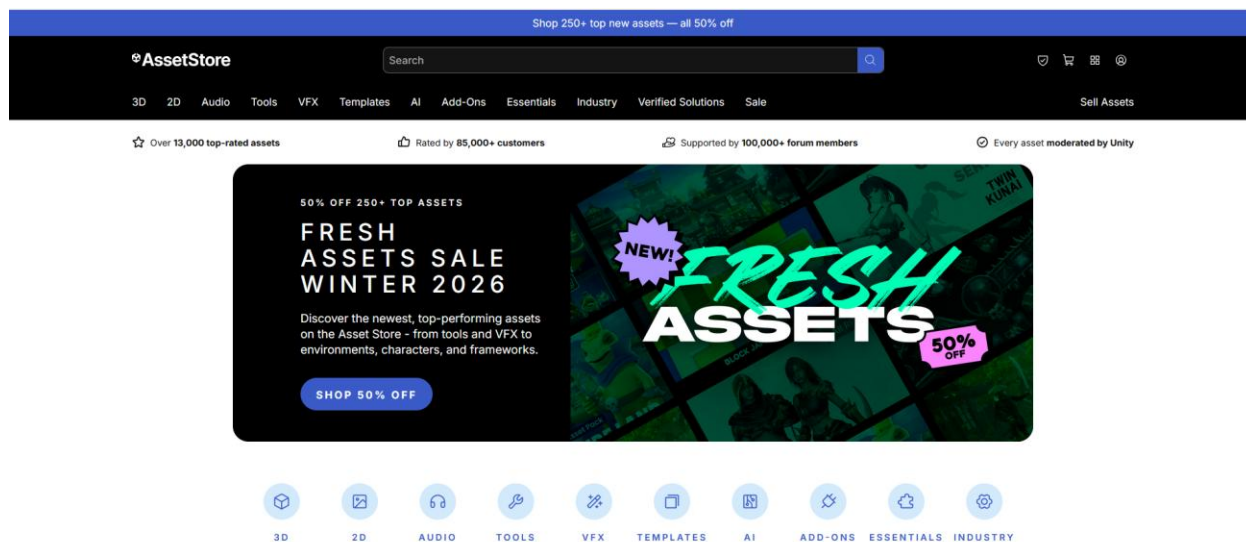
Εικ. 3-6 Mixamo

πηγή: <https://www.mixamo.com/>



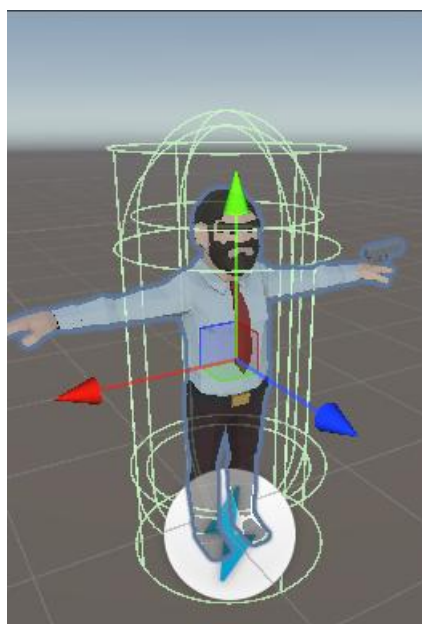
Εικ. 3-7 Kenney assets

πηγή: kenney.nl/assets

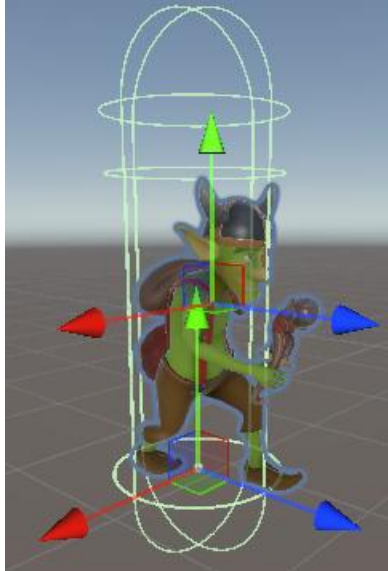


On sale assets

Εικ. 3-8 Παράθυρο Unity Asset Store



Εικ. 3-9 Τρισδιάστατο μοντέλο χαρακτήρα (Unity)



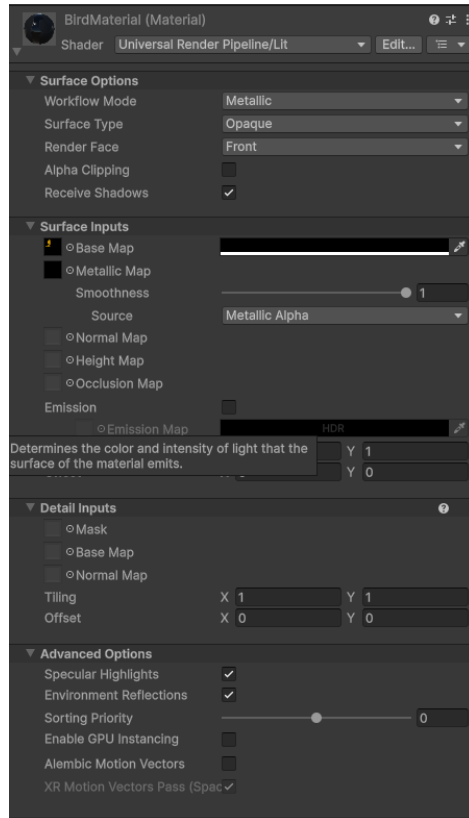
Εικ. 3-10 Τρισδιάστατο μοντέλο ζωτικού (Unity)

3.1.4 Υλικά (Materials)

Τα Materials ή αλλιώς υλικά είναι αντικείμενα παιχνιδιού που εφαρμόζονται σε άλλα πλέγματα (meshes) ή αλλιώς τρισδιάστατα μοντέλα με στόχο την εμφάνιση τους. Περιλαμβάνει διάφορες ιδιότητες όπως η αντανάκλαση, έμφραξη(ρυθμίζει τις σκιές φωτισμού σε εσωτερικές εσοχές) και ύψος λεπτομερειών μέσω των χαρτών (Maps).

Τα Shaders είναι μία σειρά οδηγιών που δίνονται στο Unity για να δημιουργηθούν νέα υλικά για το αντικείμενο, ελέγχει πως το αντικείμενο αντιδρά στον φωτισμό, πόσο γυαλιστερό φαίνεται ή μουντό και εάν φωτίζεται στο σκοτάδι.

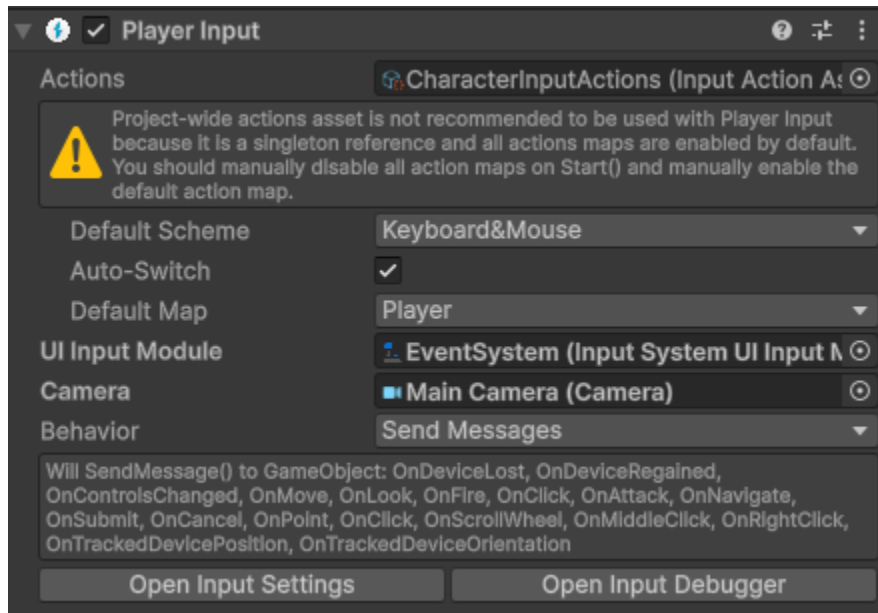
Τα Textures ή αλλιώς υφές είναι εικόνες που εφαρμόζονται σε ένα υλικό για να προσθέσει χρώμα, λεπτομέρειες και ειδικά εφέ. Μπορούν να αναπαραστήσουν λεπτομέρειες επιφάνειας όπως χρώμα, λάμψη και κανονικοί χάρτες (Normal Maps). Επίσης επιτυγχάνουν ρεαλισμό σε εφαρμογή υλικών.



Εικ. 3-11 Παράθυρο Inspector για υλικό

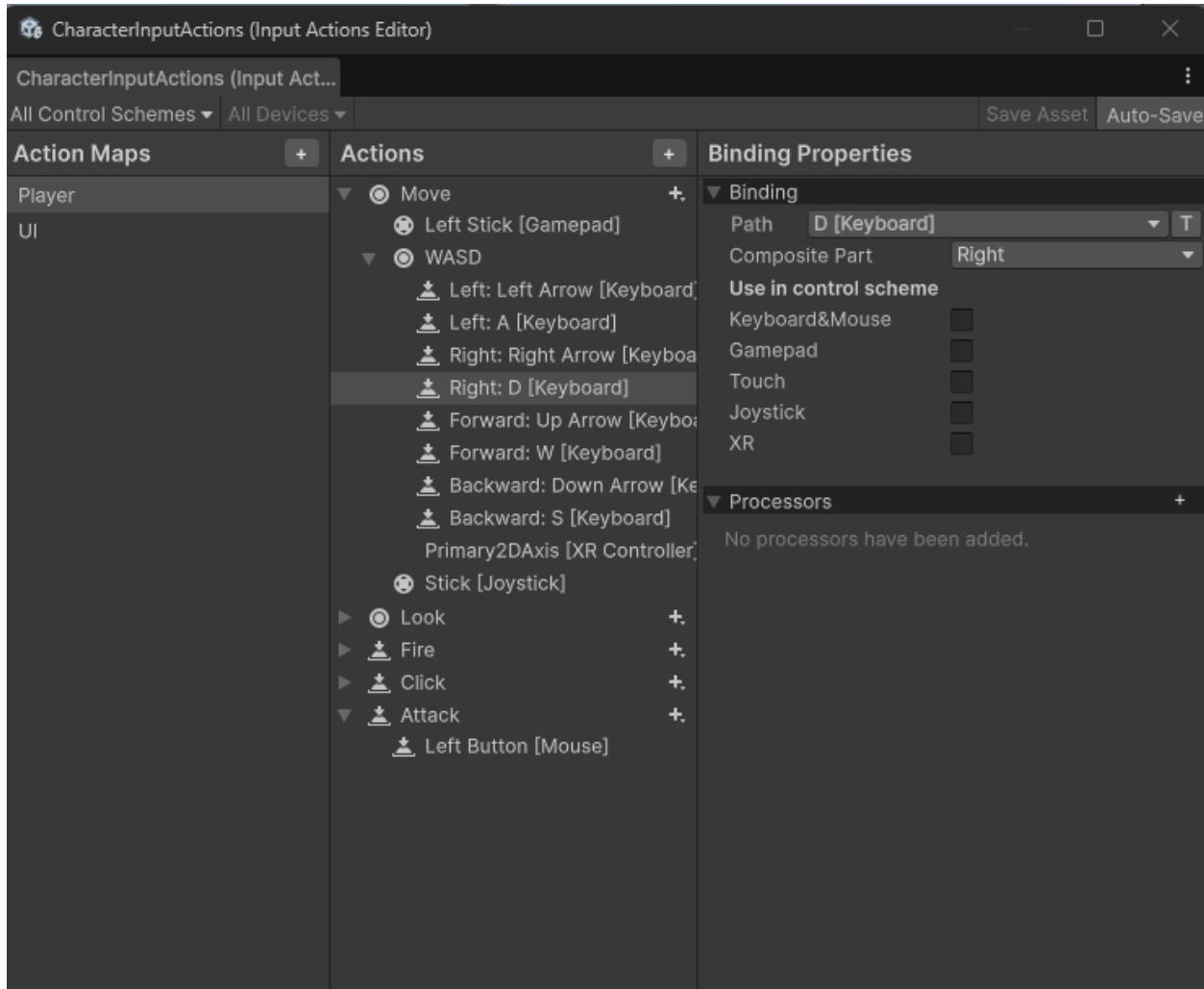
3.1.5 Χειρισμός χαρακτήρα

Για τον χειρισμό του χαρακτήρα, χρησιμοποιήθηκε το πρόσθετο Player Input. Το Player Input καθορίζει το χειρισμό του παίκτη ανάλογα με τη συσκευή εισόδου πχ Πληκτρολόγιο ή Χειριστήριο παιχνιδιού. Οι μεταβλητές που δέχονται είναι το παράθυρο εισόδου δράσεων, σύστημα Event για UI, κάμερα παιχνιδιού και συμπεριφορά που συνοδεύει την είσοδο.



Εικ. 3-12 Καρτέλα Player Input

Στο αντικείμενο script `CharacterInputActions`, βλέπουμε τις ενέργειες που συνοδεύονται από τα κουμπιά ή τα πλήκτρα και τον καθορισμό των συσκευών εισόδου για την εκτέλεση τους. Για παράδειγμα στην κίνηση του χαρακτήρα μπορούμε να καθορίσουμε τα κουμπιά ή τα Inputs για την εκτέλεση του με το WASD στο πληκτρολόγιο ή το Stick στο χειριστήριο σε κάθε κίνηση ξεχωριστά όπως προς στα πίσω.



Εικ. 3-13 Παράθυρο CharacterInputActions

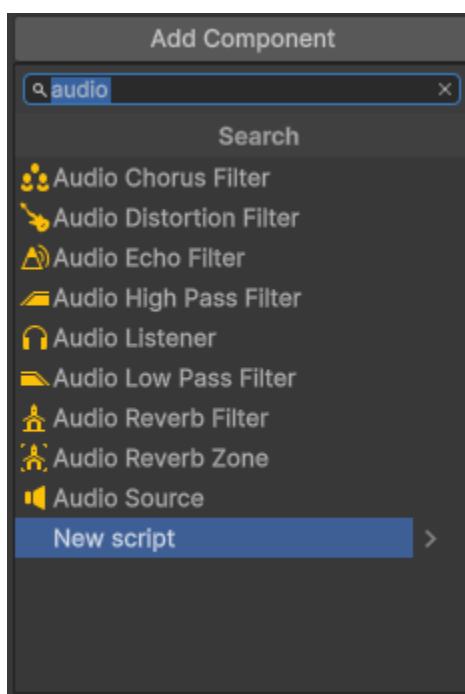
3.1.6 Ενσωματωμένος ήχος

Η μηχανή παιχνιδιών Unity παρέχει έτοιμο audio framework με ευκολία στην χρήση σε σχέση με άλλα framework που κυκλοφορούν στο εμπόριο. Υποστηρίζει μία γκάμα αρχείων ήχου για εισαγωγή ήχων.

Για να προσομοιώσει τα αποτελέσματα της θέσης, το Unity απαιτεί οι ήχοι να προέρχονται από πηγές ήχου που είναι συνδεδεμένες με αντικείμενα παιχνιδιού (Gameobjects). Οι ήχοι που εκπέμπονται στη συνέχεια συλλαμβάνονται από έναν ακροατή ήχου που είναι συνδεδεμένος με ένα άλλο αντικείμενο, συνήθως την κύρια κάμερα. Το Unity μπορεί στη συνέχεια να προσομοιώσει τα αποτελέσματα της απόστασης και της θέσης μιας πηγής από το αντικείμενο ακροατή και να τα αναπαράγει ανάλογα. Μπορούμε επίσης να χρησιμοποιήσουμε τη σχετική ταχύτητα των αντικειμένων πηγής και ακροατή μέσω της προσομοίωσης του φαινομένου Doppler για μεγαλύτερο ρεαλισμό.

Το Unity δεν μπορεί να υπολογίσει τις ηχώ αποκλειστικά από τη γεωμετρία της σκηνής (Scene), αλλά μπορεί να προσομοιωθεί προσθέτοντας φίλτρα ήχου στα αντικείμενα. Για παράδειγμα, ο ήχος από μέσα στην σπηλιά θα μπορούσατε να εφαρμοστεί με το φίλτρο ηχώ. Σε περιπτώσεις όπου τα αντικείμενα μπορούν να κινούνται μέσα και έξω από ένα χώρο με έντονη ηχώ, μπορεί να προστεθεί μια ζώνη αντήχησης στη σκηνή. Για παράδειγμα, το παιχνίδι μπορεί να περιλαμβάνει αυτοκίνητα που οδηγούν μέσα από ένα τούνελ. Αν τοποθετηθεί μια ζώνη αντήχησης μέσα στο τούνελ, οι ήχοι των κινητήρων των αυτοκινήτων αρχίζουν να αντηχούν καθώς εισέρχονται. Η ηχώ εξαφανίζεται καθώς τα αυτοκίνητα βγαίνουν από την άλλη πλευρά.

Παρακάτω είναι μία λίστα από τα αντικείμενα (Components) ήχου που παρέχει το Unity και δίνεται ένας πίνακας με την περιγραφή τους.



Εικ. 3-14 Λίστα συστατικών για τον ήχο στο Unity

Συστατικό (Component)	Περιγραφή
Audio Chorus Filter	Δημιουργεί ένα «χορωδιακό» εφέ
Audio Distortion Filter	Παραμορφώνει τον ήχο από μία πηγή (Source) ή ήχοι που φτάνουν στον Ακροατή (Listener)
Audio Echo Filter	Επαναλαμβάνει έναν ήχο μετά από μια δεδομένη καθυστέρηση, εξασθενίζοντας τις επαναλήψεις με βάση τον λόγο αποσύνθεσης.
Audio High/Low Pass Filter	Εφαρμόζει Υψηλοπερατό/Χαμηλοπερατό φίλτρο από μία πηγή ήχου
Audio Listener	Λειτουργεί ως συσκευή που μοιάζει με μικρόφωνο. Λαμβάνει είσοδο από

	οποιαδήποτε δεδομένη πηγή ήχου στη σκηνή και αναπαράγει ήχους μέσω των ηχείων. Για τις περισσότερες εφαρμογές είναι λογική η σύνδεση τον ακροατή (Listener) στην Κύρια Κάμερα της σκηνής. Εάν ένας ακροατής ήχου βρίσκεται εντός των ορίων μιας ζώνης αντήχησης, η αντήχηση εφαρμόζεται σε όλους τους ακουστούς ήχους στη σκηνή. Επιπλέον, τα εφέ ήχου μπορούν να εφαρμοστούν στον ακροατή σε όλους τους ακουστούς ήχους στη σκηνή.
Audio Reverb Filter	Λαμβάνει ένα ηχητικό κλιπ και το παραμορφώνει για να δημιουργήσει ένα προσαρμοσμένο εφέ αντήχησης.
Audio Reverb Zone	Προσθέτει ρεαλισμό ηχοτοπίου στη σκηνή ελέγχοντας τον τρόπο με τον οποίο ο χώρος ενός τρισδιάστατου περιβάλλοντος μπορεί να επηρεάσει τις ιδιότητες του ήχου που αντηχεί στις επιφάνειες.
Audio Source	Αναπαράγει ένα ηχητικό κλιπ στη σκηνή και παρέχει επιλογές για την προσαρμογή της συμπεριφοράς αναπαραγωγής του

Πίνακας 3-1 Πίνακας περιγραφής συστατικών Audio του Unity

3.1.7 Άκαμπτο σώμα (Rigidbody)

Το Rigidbody είναι ένα συστατικό που ελέγχει την θέση ενός αντικειμένου παιχνιδιού μέσω προσομοίωσης φυσικής.

Η προσθήκη ενός στοιχείου Rigidbody σε ένα αντικείμενο παιχνιδιού θα θέσει την κίνησή του υπό τον έλεγχο της μηχανής φυσικής της Unity. Ακόμα και χωρίς την προσθήκη κώδικα, ένα αντικείμενο Rigidbody θα έλκεται προς τα κάτω από τη βαρύτητα και θα αντιδρά σε συγκρούσεις με εισερχόμενα αντικείμενα εάν υπάρχει και το σωστό στοιχείο σύγκρουσης (Collider).

Διαθέτει επίσης ένα API scripting που επιτρέπει την εφαρμογή δυνάμεων στο αντικείμενο και τον έλεγχο του με έναν φυσικά ρεαλιστικό τρόπο. Για παράδειγμα, η συμπεριφορά ενός αυτοκινήτου μπορεί να καθοριστεί με βάση τις δυνάμεις που ασκούνται από τους τροχούς. Δεδομένων αυτών των πληροφοριών, η μηχανή φυσικής μπορεί να χειριστεί τις περισσότερες άλλες πτυχές της κίνησης του αυτοκινήτου, επομένως θα επιταχύνει ρεαλιστικά και θα αντιδρά σωστά στις συγκρούσεις.

Σε ένα σενάριο, η συνάρτηση FixedUpdate συνιστάται ως το σημείο εφαρμογής δυνάμεων και αλλαγής των ρυθμίσεων του Rigidbody (σε αντίθεση με το Update, το οποίο χρησιμοποιείται για τις περισσότερες άλλες εργασίες ενημέρωσης πλαισίου). Ο λόγος για αυτό είναι ότι οι

ενημερώσεις φυσικής εκτελούνται σε μετρημένα χρονικά βήματα που δεν συμπίπτουν με την ενημέρωση πλαισίου. Το FixedUpdate καλείται αμέσως πριν από κάθε ενημέρωση φυσικής και έτσι τυχόν αλλαγές που γίνονται εκεί θα υποβάλλονται σε άμεση επεξεργασία.

Ένα συνηθισμένο πρόβλημα με τα Rigidbodies είναι ότι η φυσική του παιχνιδιού φαίνεται να τρέχει σε "αργή κίνηση". Αυτό οφείλεται στην πραγματικότητα στην κλίμακα που χρησιμοποιείται για τα μοντέλα. Οι προεπιλεγμένες ρυθμίσεις βαρύτητας υποθέτουν ότι μία μονάδα κόσμου αντιστοιχεί σε ένα μέτρο απόστασης. Με τα μη φυσικά παιχνίδια, δεν έχει μεγάλη διαφορά αν τα μοντέλα έχουν όλα μήκος 100 μονάδες, αλλά όταν χρησιμοποιούνται αρχές φυσικής άρα θα αντιμετωπίζονται ως πολύ μεγάλα αντικείμενα. Εάν χρησιμοποιηθεί μεγάλη κλίμακα για αντικείμενα που υποτίθεται ότι είναι μικρά, θα φαίνονται να πέφτουν πολύ αργά - η μηχανή φυσικής νομίζει ότι είναι πολύ μεγάλα αντικείμενα που πέφτουν από πολύ μεγάλο ύψος. (επομένως, ένα αυτοκίνητο θα πρέπει να είναι περίπου όσο 4 μονάδες = 4 μέτρα, για παράδειγμα). (Unity, 2026)

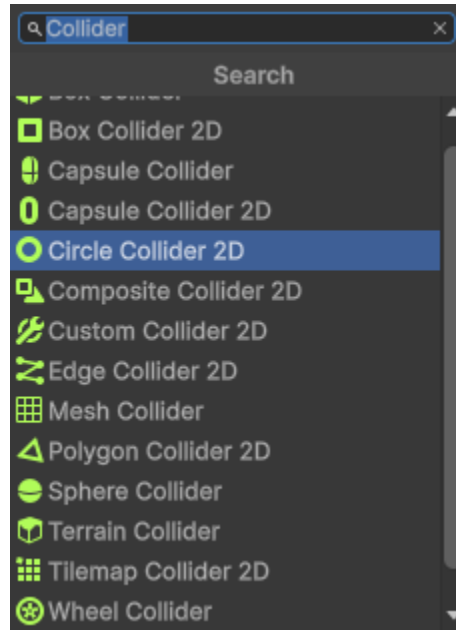
3.1.8 Συγκρούσεις (Collisions)

Τα Collision ή αλλιώς συγκρούστες συμβαίνουν όταν δύο αντικείμενα παιχνιδιού συγκρούονται μεταξύ τους και συνήθως αυτά παίρνουν το σχήμα του αντικειμένου με διαφορετικό τρόπο είτε είναι τρισδιάστατα είτε δισδιάστατα και είναι αόρατα. Το καθένα ξεχωριστά (2D και 3D) τρέχουν σε διαφορετικά συστήματα προσομοίωσης φυσικής. Το μέγεθος του σχήματος μπορεί να τροποποιηθεί από τον χρήστη.

Υπάρχουν δύο τύποι Collider, τα στατικά και τα Rigidbody. Τα στατικά Collider δεν ασκούν καμία δύναμη αρά δεν διαθέτουν Rigidbody, ενώ τα Rigidbody χωρίζονται σε δύο, τα δυναμικά και τα κινηματικά. Ενώ μπορούν να λειτουργήσουν ως αφετηρίες (Trigger) που δεν προκαλούν συγκρούσεις παρά μόνο ξεκινάνε μία ενέργεια μέσα στο παιχνίδι. Ενεργοποιούνται με την επιλογή isTriggered.

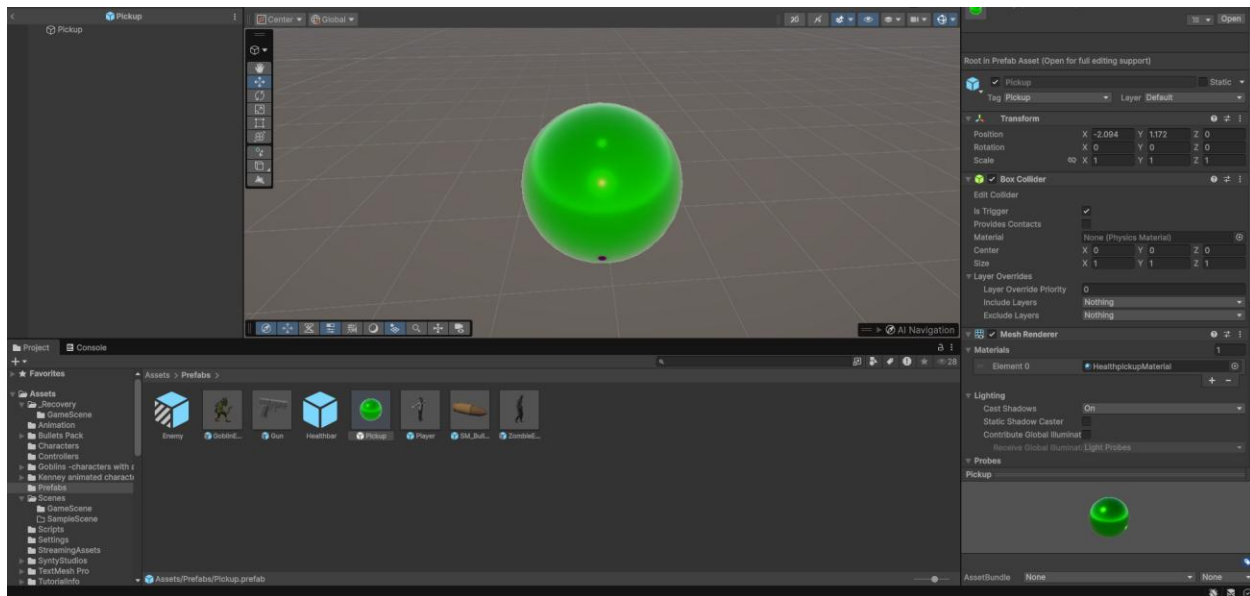
Υπάρχουν δύο τύποι συναρτήσεων στην C# που σχετίζονται με τα Collisions που συνοδεύονται με Enter, Stay και Exit στο τέλος, το OnTrigger.. που καλείται όταν ένας Collider επικαλύπτει έναν άλλο Collider και το OnCollision... όταν είναι σε επαφή με έναν άλλον Collider μέσω σύγκρουσης.

(Unity, 2026)



Εικ. 3-15 Λίστα συστατικών για τα Colliders στο Unity

Για παράδειγμα έχουμε ένα GameObject που αναπαριστά επιπλέον ζωή για τον παίκτη όπως απεικονίζεται παρακάτω αφού εξουδετερώνει έναν εχθρό και λειτουργεί ως Trigger. Όταν έρθει σε επαφή με τον παίκτη, τότε κερδίζει επιπλέον ζωή και διαγράφεται το αντικείμενο.



Εικ. 3-16 Παράθυρο αντικείμενου GameObject Pickup

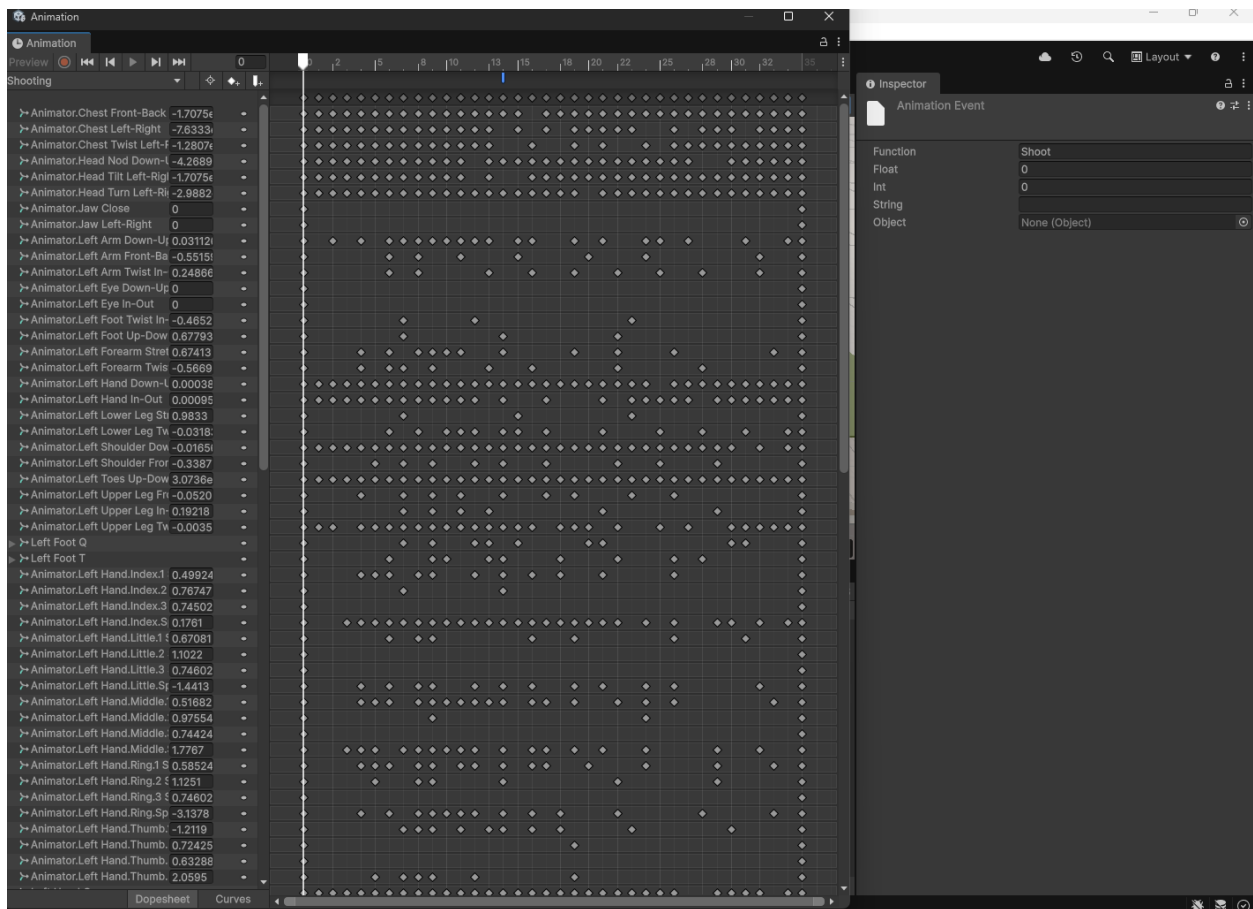
3.1.9 Κίνηση (Animation)

Τα σύστημα κίνησης του Unity αποτελείται από το Animator και το Animation. Το Animator είναι ένα είδος μηχανής κατάστασης (State Machine) που καθορίζει τις μεταβάσεις μεταξύ

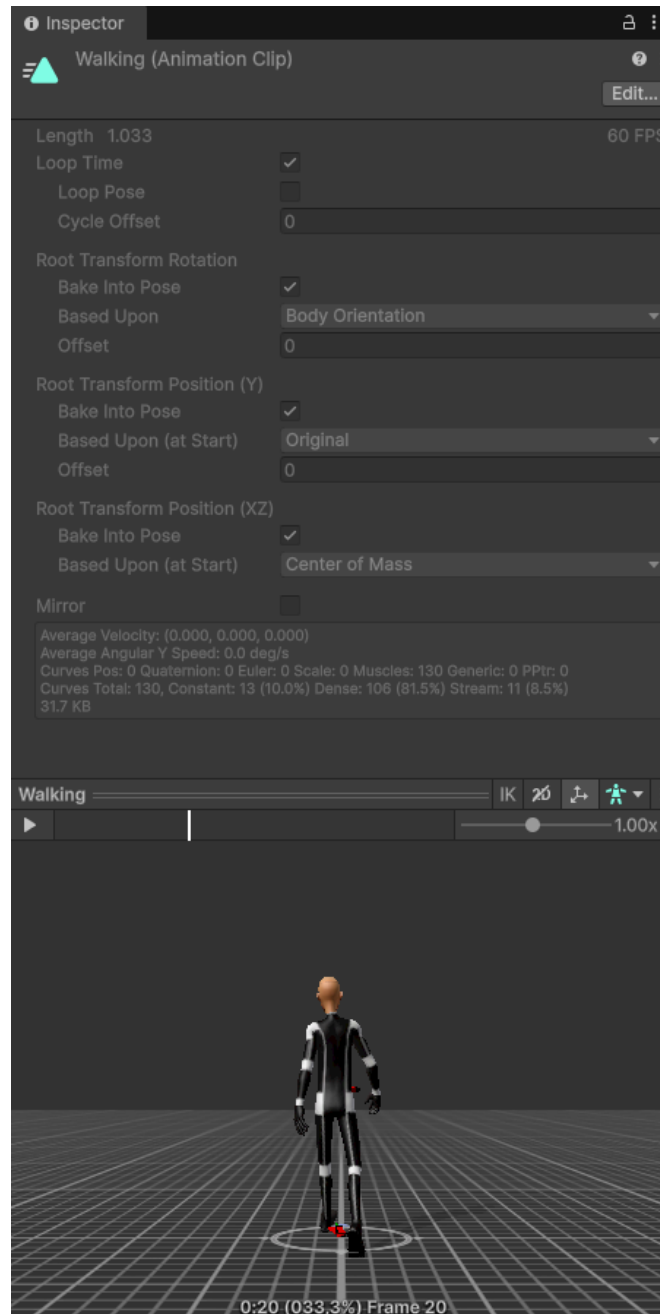
διαφορετικών καταστάσεων (states) και το Animation που είναι ένα κλιπ που αποτελείται από έναν σκελετό με ένα μοντέλο και ένα χρονοδιάγραμμα που αναπαριστά τις κινήσεις στον σκελετό.

Σε ένα Animator, υπάρχουν διάφορες ρυθμίσεις και αφετηρίες (Triggers) για την μετάβαση από ένα animation clip σε ένα άλλο. Τα triggers στις μεταβάσεις μπορούν να πάρουν την μορφή Boolean, Float, Trigger ή Integer που ελέγχονται μέσω της πρόσβασης στο ομώνυμο Component (Animator) στον κώδικα, Όταν υπάρχουν states που οδηγούν σε άλλα κρίνεται επιτακτική η ανάγκη για περιορισμό στις μεταβάσεις που λέγονται Mute και Solo. Στο Mute απενεργοποιείται μία μετάβαση και στο Solo παίζει μόνο αυτήν την μετάβαση. Όταν και τα δύο είναι ενεργοποιημένα, προβάδισμα έχει το Mute.

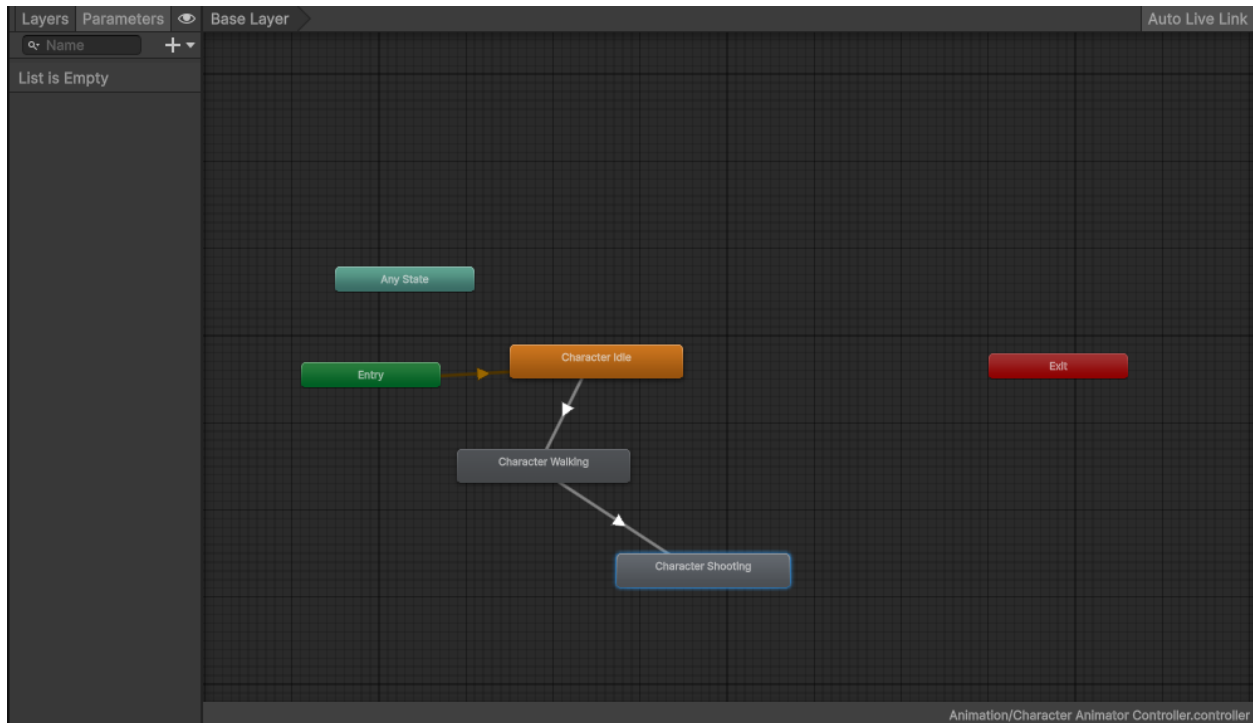
Σε κάθε Animation Clip μπορούμε να ορίσουμε Event σε ένα timestamp μία συνάρτηση ή μία μεταβλητή.



Εικ. 3-17 Παράθυρο Event ενός Animation Clip



Εικ. 3-18 Καρτέλα Animation στο Unity.



Εικ. 3-19 Μηχανή κατάστασης Animator στο Unity

(Unity, 2026)

3.1.10 Κάμερα παιχνιδιού

Η κάμερα λειτουργεί ως μία συσκευή που ο παίκτης βλέπει τον κόσμο.

Ένα σημείο στο χώρο της οθόνης ορίζεται σε εικονοστοιχεία (Pixels). Η κάτω αριστερή γωνία της οθόνης είναι (0,0)· η πάνω δεξιά γωνία είναι (pixelWidth,pixelHeight). Η θέση z εκφράζεται σε μονάδες κόσμου με βάση την Κάμερα

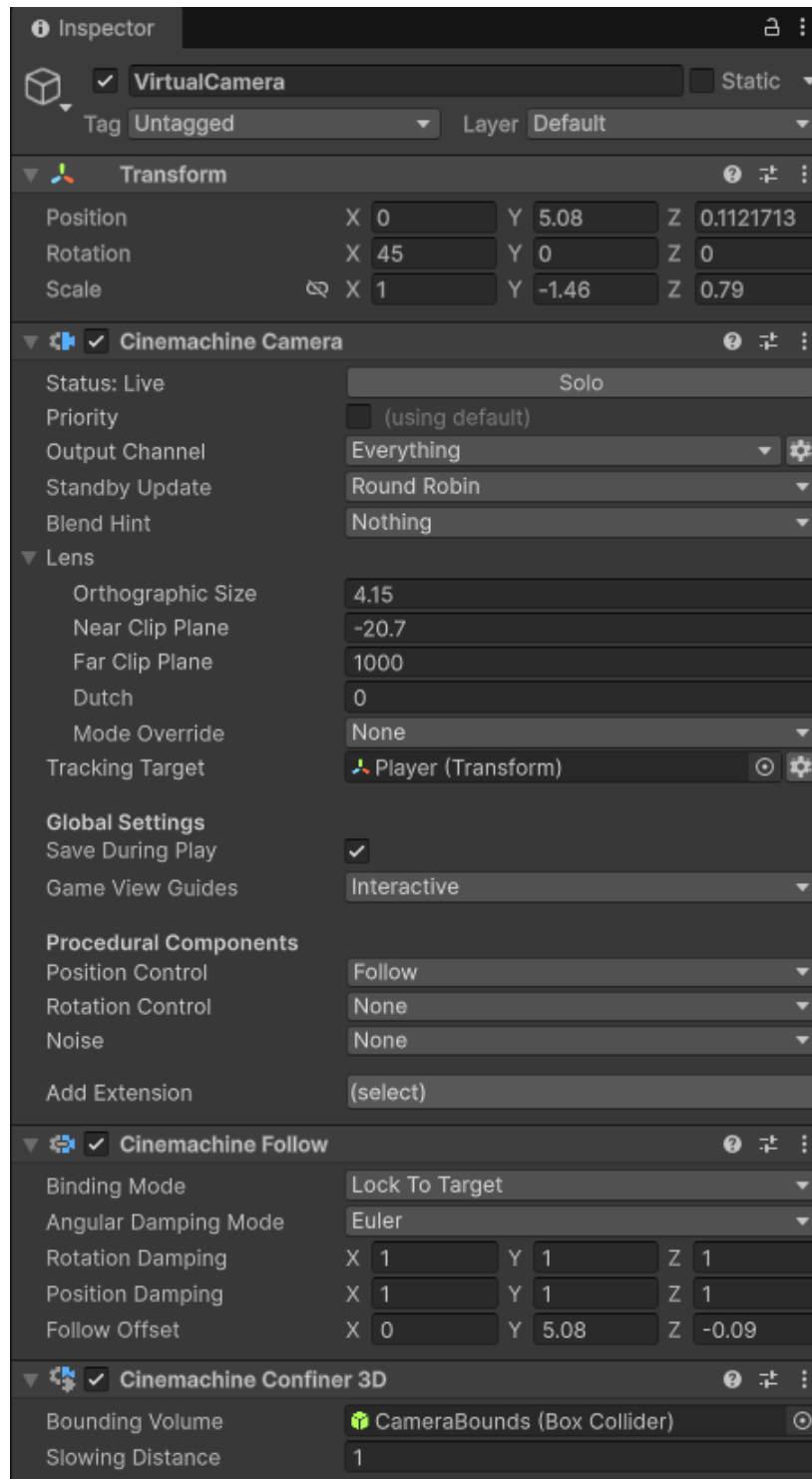
Ένα σημείο στο χώρο του παραθύρου προβολής είναι κανονικοποιημένο και σχετικό με την Κάμερα. Η κάτω αριστερή γωνία της Κάμερας είναι (0,0) και η πάνω δεξιά γωνία είναι (1,1). Η θέση z εκφράζεται σε μονάδες κόσμου από την Κάμερα.

Ένα σημείο στο χώρο του κόσμου ορίζεται σε προκαθορισμένες (world) συντεταγμένες (για παράδειγμα, Transform.position).

Χρησιμοποιήθηκε το Cinemachine γιατί βοηθάει στον καλύτερο έλεγχο της κάμερας του παιχνιδιού στο Unity χωρίς την ανάγκη για περίπλοκο κώδικα.

Το αντικείμενο VirtualCamera κανονίζει τις λειτουργίες του όπως ακολουθία παίκτη, ορισμός εμποδίων κάμερας με την χρήση του Collider και έλεγχος καταστάσεων (States) που ορίζουν την κίνηση και τον προσανατολισμό της κάμερας.

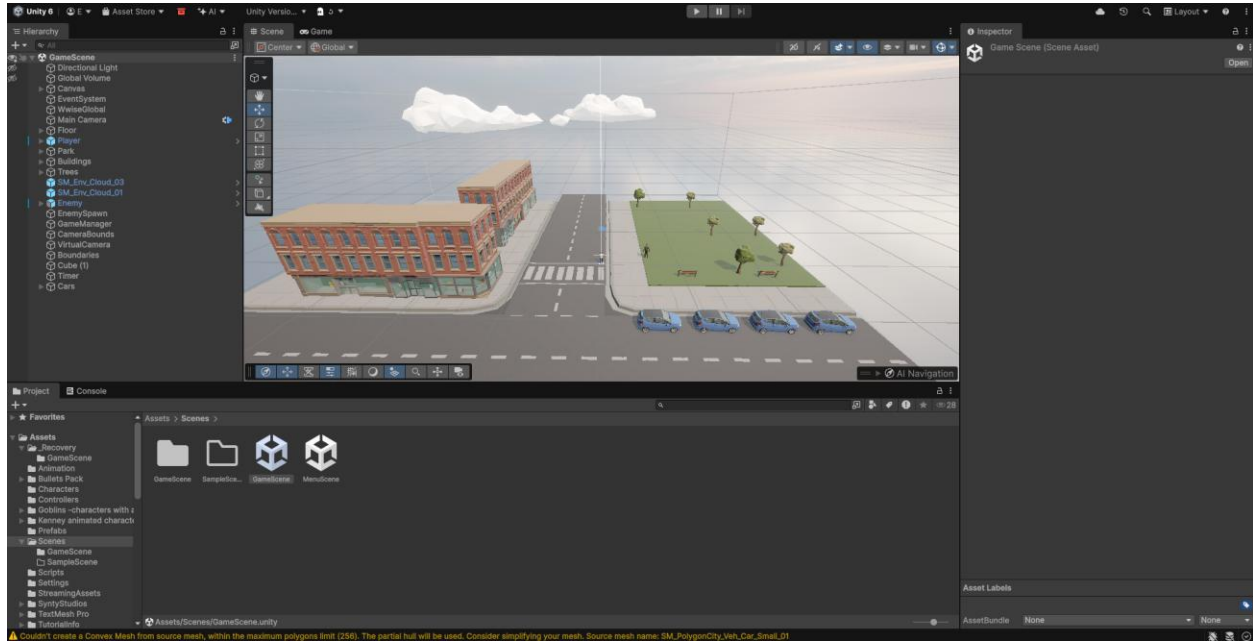
Ο έλεγχος του αντικειμένου παιχνιδιού (GameObject) VirtualCamera γίνεται στο αντικείμενο της κάμερας μέσω του στοιχείου VirtualCamera Brain.



Εικ. 3-20 Παράθυρο στοιχείων (Components) του αντικειμένου VirtualCamera.

3.1.11 Σκηνές (Scenes)

Μία σκηνή παιχνιδιού περιέχει δεδομένα, αντικείμενα παιχνιδιού και μπορεί να αποτελέσει και ως πίστα ή μενού σε κάποιο παιχνίδι. Μία βασική σκηνή περιέχει μία κύρια κάμερα και πηγή φωτισμού. Μπορούμε να δημιουργήσουμε μία σκηνή από ένα prefab ή από την αρχή.



Εικ. 3-21 Σκηνή παιχνιδιού

3.1.12 Πράκτορες (Agents)

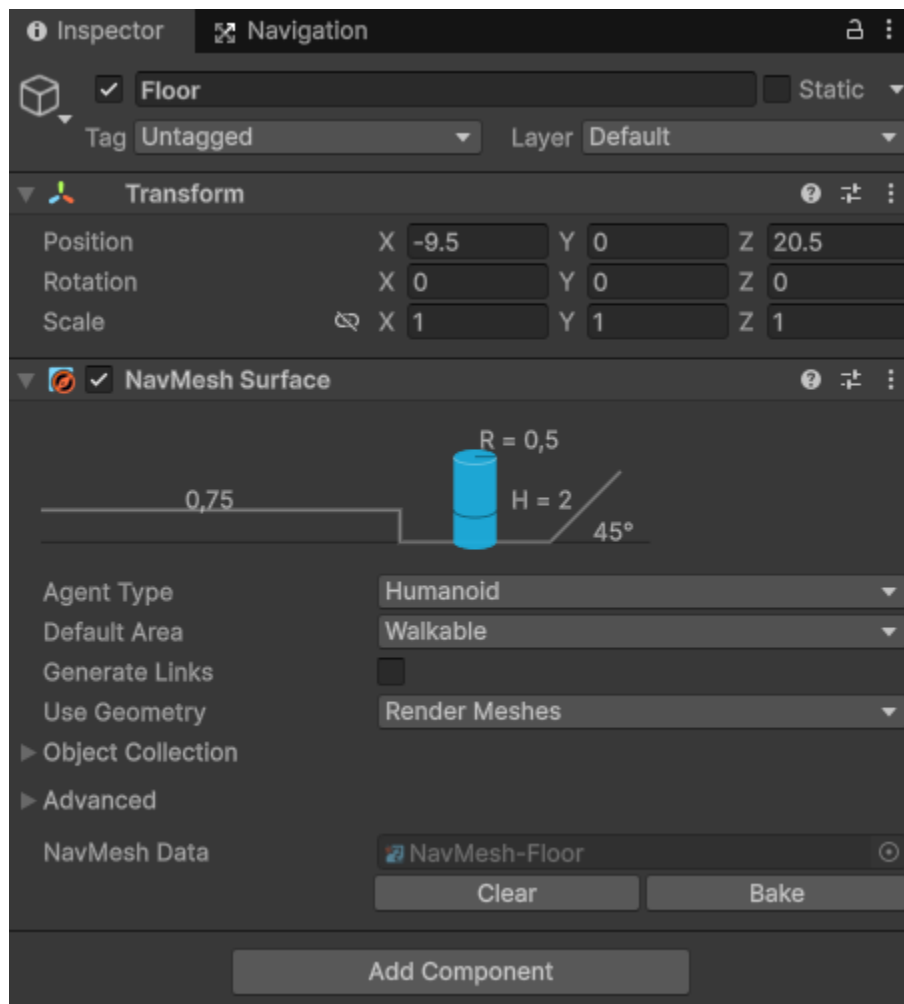
Για να ελεγχθεί η κίνηση των εχθρών προς τον παίκτη, χρησιμοποιήθηκε η βιβλιοθήκη NavMesh που εκτελεί χωρικά ερωτήματα, όπως δοκιμές εύρεσης μονοπατιού και βατότητας. Επιτρέπει επίσης τον ορισμό του κόστους εύρεσης μονοπατιών για συγκεκριμένους τύπους περιοχών και την τροποποίηση της καθολικής συμπεριφοράς της εύρεσης μονοπατιών και της αποφυγής.

Παρακάτω δίνεται ο πίνακας των συστατικών (Components) της βιβλιοθήκης

Component	Περιγραφή
Nav Mesh Agent	Επιτρέπει τον χαρακτήρα να κινείται στην σκηνή και ορίζεται ως πράκτορας
NavMeshObstacle	Έχει κυλινδρικό σχήμα και μπορεί να κινείται γύρω από την επιφάνεια του NavMesh με καθορισμένη ταχύτητα. Από προεπιλογή, το εμπόδιο θα επηρεάσει μόνο τη συμπεριφορά αποφυγής του πράκτορα και όχι την εύρεση διαδρομής. Αυτό σημαίνει ότι ο πράκτορας θα αγνοήσει το εμπόδιο όταν σχεδιάζει ένα μονοπάτι, αλλά θα το παρακάμψει ενώ κινείται κατά μήκος του μονοπατιού. Εάν το

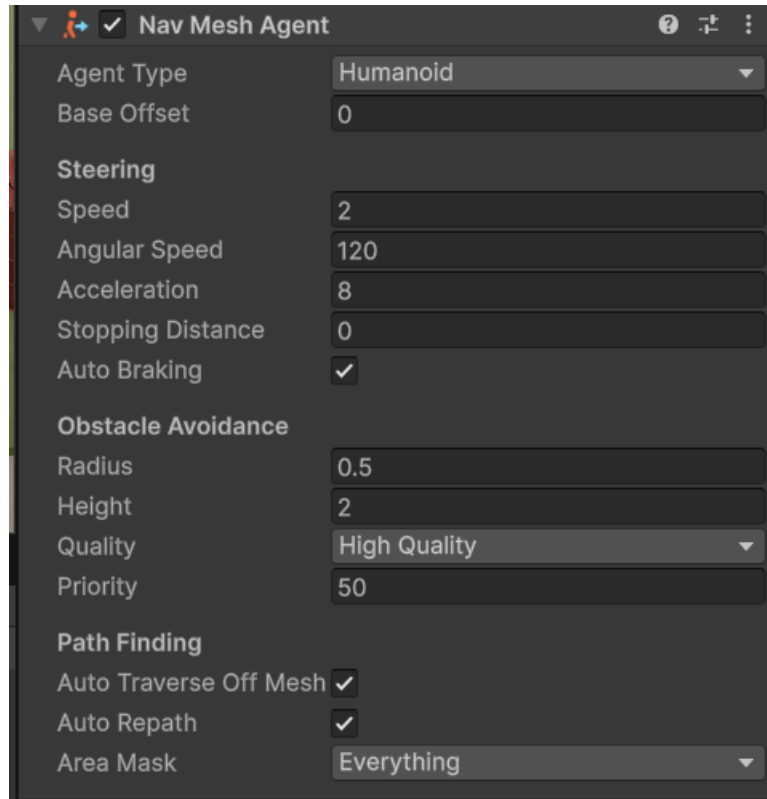
	σκάλισμα είναι ενεργοποιημένο, το εμπόδιο θα δημιουργήσει μια προσωρινή "τρύπα" στο NavMesh. Η τρύπα θα αναγνωριστεί από τον εντοπισμό του μονοπατιού, επομένως θα σχεδιαστούν μονοπάτια για να αποφευχθεί το εμπόδιο. Αυτό σημαίνει ότι εάν, για παράδειγμα ένα εμπόδιο μπλοκάρει ένα στενό χάσμα, το μονοπάτι θα αναζητήσει μια εναλλακτική διαδρομή προς τον στόχο. Χωρίς σκάλισμα, ο πράκτορας θα κατευθυνθεί προς το κενό, αλλά δεν θα μπορεί να περάσει μέχρι να ξεκαθαρίσει το εμπόδιο.
NavMesh Surface	Αναπαριστά μία περιοχή για περπάτημα για έναν συγκεκριμένο τύπο πράκτορα NavMesh και ορίζει ένα τμήμα της σκηνής όπου θα πρέπει να κατασκευαστεί ένα NavMesh

Πίνακας 3-2 Πίνακας συστατικών του NavMesh



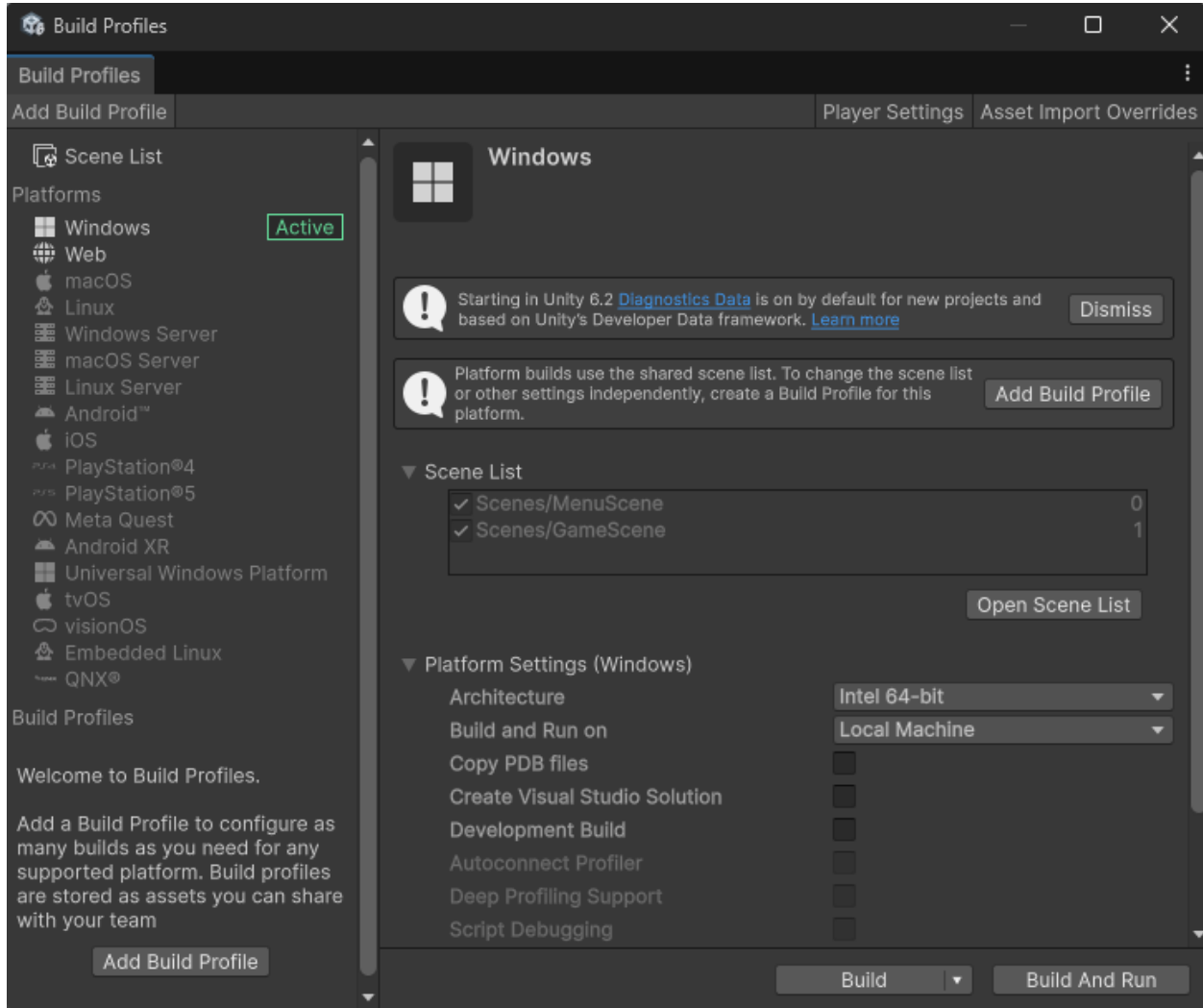
Εικ. 3-22 Παράθυρο NavMesh Surface

Στην καρτέλα του NavMesh surface βλέπουμε τις παραμέτρους που ορίζουν την επιφάνεια που θα πατήσει ο παίκτης, τον τύπο, την χρήση της γεωμετρίας και την καθορισμένη περιοχή



Εικ. 3-23 Παράθυρο NavMesh Surface

3.1.13 Έκδοση (Build)



Εικ. 3-24 Παράθυρο Build Profiles

Ένα προφίλ έκδοσης ή αλλιώς Build Profiles είναι ένα σύνολο ρυθμίσεων διαμόρφωσης που χρησιμοποιείται στην δημιουργία την εφαρμογή σε μια συγκεκριμένη πλατφόρμα. Χρησιμοποιώντας το παράθυρο «Προφίλ έκδοσης» δημιουργούμε πολλά προφίλ έκδοσης για κάθε πλατφόρμα στην οποία εργαζόμαστε, αποθηκεύοντας διαφορετικές διαμορφώσεις για εκδόσεις κυκλοφορίας και ανάπτυξης

Υπάρχουν δύο τύποι προφίλ διαθέσιμοι στο παράθυρο Προφίλ κατασκευής, οι πλατφόρμες και προφίλ.

Στο παράθυρο «Πλατφόρμες» εμφανίζει μια λίστα με τις τρέχουσες εγκατεστημένες πλατφόρμες που υποστηρίζει το Unity. Ένα προφίλ πλατφόρμας περιλαμβάνει ορισμένες κοινές ρυθμίσεις που ισχύουν για όλες τις πλατφόρμες. Για παράδειγμα, αν ενεργοποιήσετε τη ρύθμιση Κατασκευή ανάπτυξης για ένα προφίλ πλατφόρμας, το Unity θα ενεργοποιήσει τη ρύθμιση σε όλα τα διαθέσιμα προφίλ πλατφόρμας. Οι πλατφόρμες μοιράζονται επίσης τα ίδια δεδομένα σκηνής

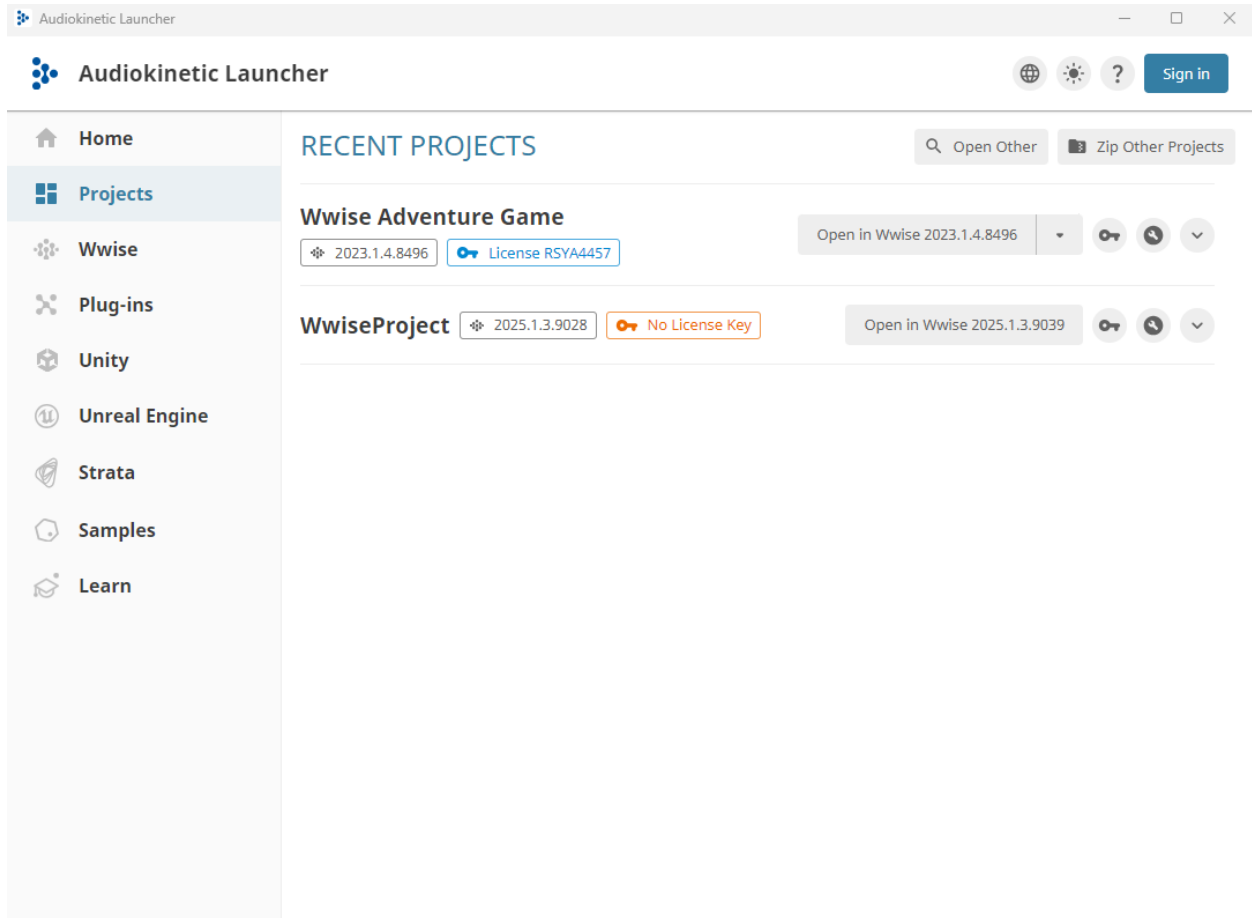
Σε αντίθεση με τις πλατφόρμες, οι ρυθμίσεις που αποθηκεύονται στα προφίλ κατασκευής δεν κοινοποιούνται σε όλες τις πλατφόρμες. Μπορείτε να αντιστοιχίσετε συγκεκριμένες σκηνές σε κάθε προφίλ κατασκευής. Τα προφίλ κατασκευής επιτρέπουν την αποθήκευση ανεξάρτητων διαμορφώσεων κατασκευής. Μπορούμε να αποθηκεύσουμε όσα προφίλ κατασκευής χρειαζόμαστε, χρησιμοποιώντας ένα προσαρμοσμένο όνομα για κάθε προφίλ. Το Unity αποθηκεύει το προφίλ κατασκευής ως αρχείο στοιχείου που είναι έτοιμο για χρήση με τον έλεγχο εκδόσεων

(Unity, 2026)

3.2 Wwise

Σε αυτήν την υποενότητα γίνεται η εξήγηση των υπομερούς τμημάτων του Wwise για καλύτερη κατανόηση και εξοικείωση το οποίο λειτουργεί θετικά εις πέρας της υλοποίησης της διπλωματικής. Όλα τα επιμερούς αντικείμενα που αναφέρονται παρακάτω καθώς και οι εικόνες και τα σχεδιαγράμματα παραπέμπουν στην επίσημη τεκμηρίωση (Documentation) του Wwise.

Η ενσωμάτωση του γίνεται αρχικά μέσω του Launcher της Audiokinetic και συγκεκριμένα στην καρτέλα της υποστηριζόμενης μηχανής παιχνιδιού με την επιλεγμένη έκδοση του Wwise και μετά την αφετηρία της στην καρτέλα των Projects.



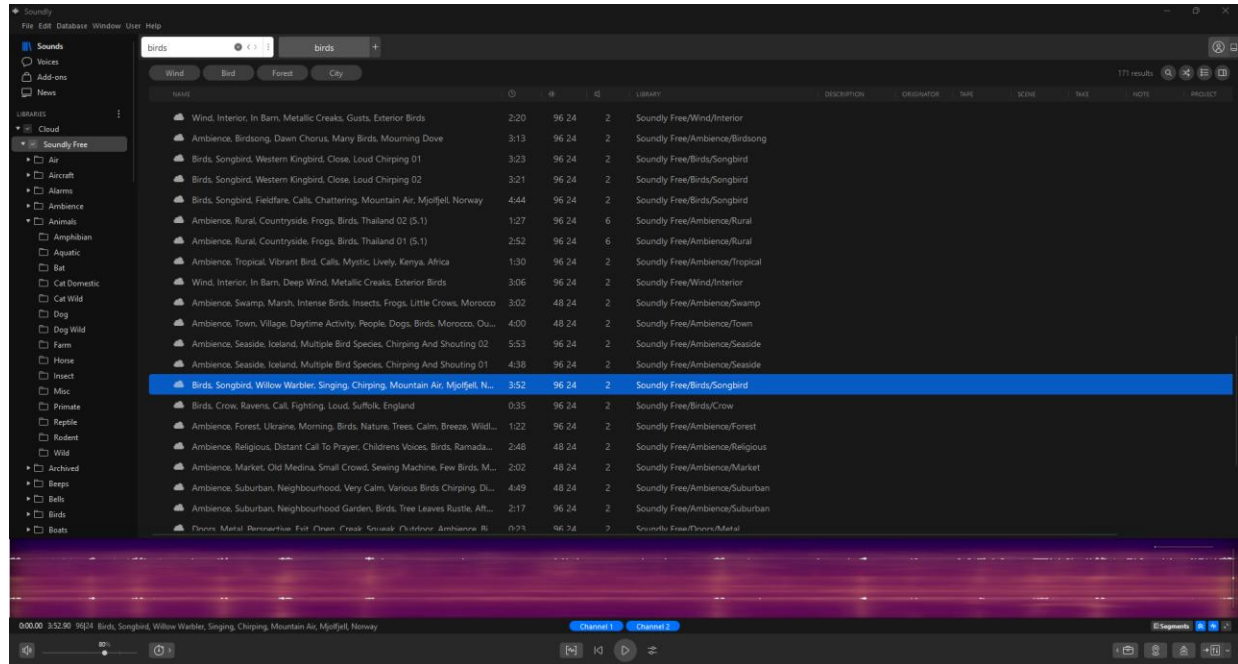
Εικ. 3-25 Παράθυρο εκκίνησης Audiokinetic Launcher



Εικ. 3-26 Audiokinetic Wwise

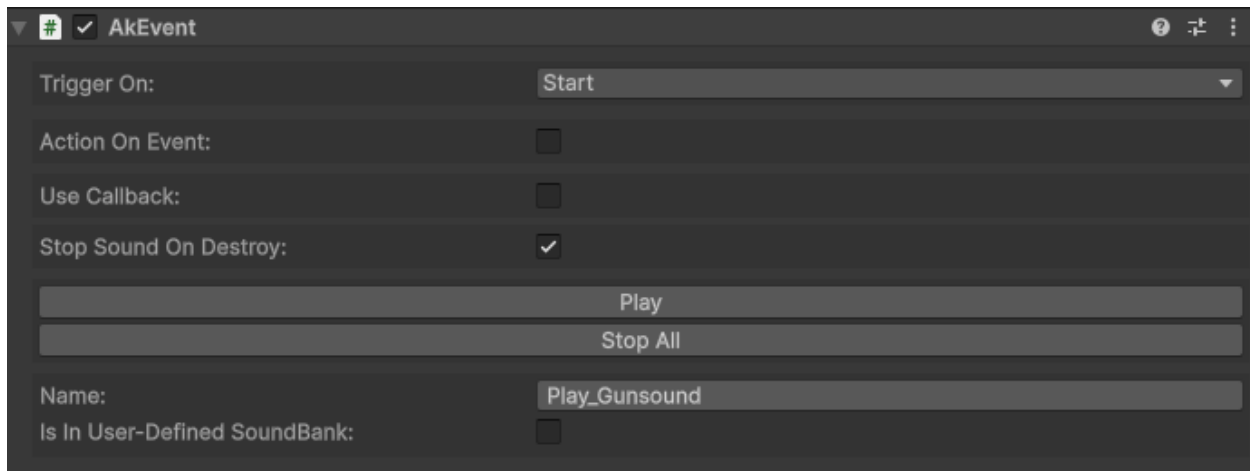
3.2.1 Ήχοι παιχνιδιού

Για την επιλογή των ήχων για το παιχνίδι χρησιμοποιήθηκε η πλατφόρμα Soundly λόγω της ποιότητας των ήχων, παρέχει μεγάλη ποικιλία ήχου μέσω cloud και μπορεί εύκολα να κόψει κανείς ένα μικρό κομμάτι από το αρχείο ήχου και να ενσωματωθεί εύκολα στο Wwise. Επίσης έχουν χρησιμοποιηθεί και ήχους που διατίθενται δωρεάν αν και η ποιότητα τους δεν είναι η ιδανική.

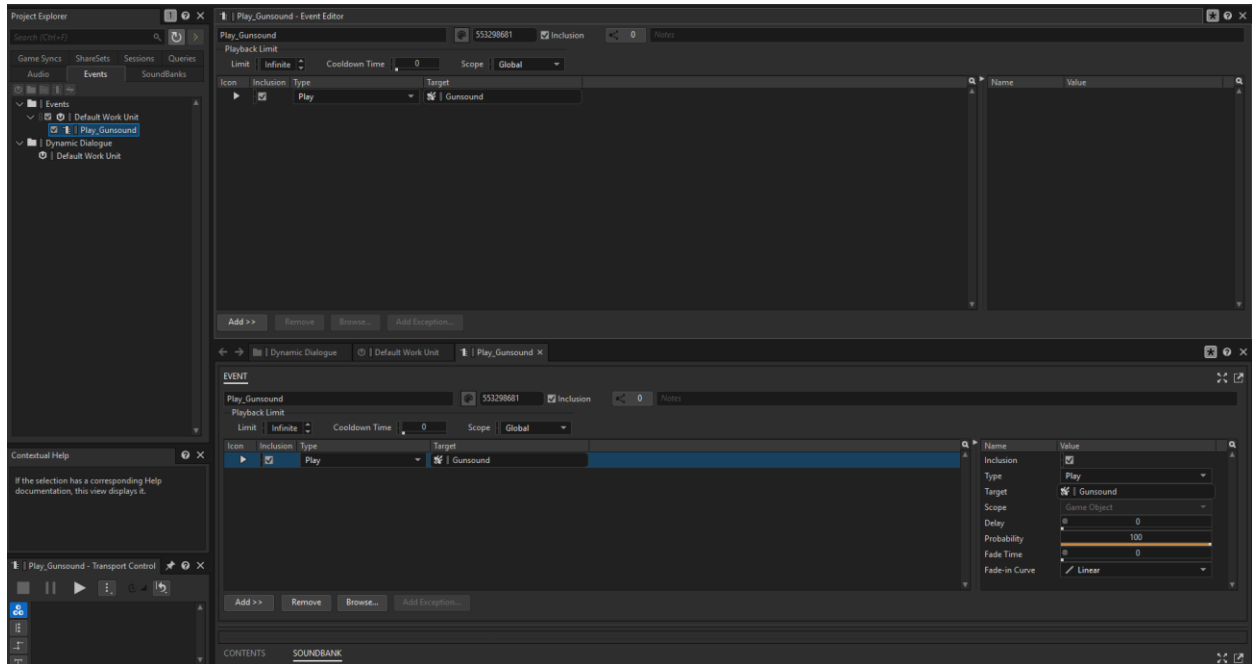


Εικ. 3-27 Παράθυρο Soundly.

Για την φόρτωση των ήχων στο παιχνίδι, χρησιμοποιούνται Events, στο παράθυρο Event Editor γίνεται η προσθήκη των ηχητικών εφέ όπως για παράδειγμα χαμηλοπερατά και υψηλοπερατά φίλτρα στους ήχους και φορτώνονται στο Unity μέσω του Component AkEvent ή του AkAmbient. Για να γίνει η εκκίνηση για παράδειγμα του ήχου του όπλου, θα χρειαστεί εκκίνηση του παίκτη όταν πυροβολεί, δηλαδή το κατάλληλο AkEvent.



Εικ. 3-28 AkEvent Component



Εικ. 3-29 Παράθυρο Event Editor στο Wwise

Για την δημιουργία των ήχων των τεράτων στο παιχνίδι, δημιουργήθηκε ένα Event Random Sequencer που αναπαράγει τυχαία σε σειρά τους ήχους μαζί με επεξεργασία για να επιτευχθεί ένα πιο φυσικό αποτέλεσμα.

3.2.2 Αντικείμενα παιχνιδιού (Game Objects)

Τα αντικείμενα παιχνιδιού είναι το κυρίως αντικείμενο στο Wwise, επειδή κάθε συμβάν που ενεργοποιείται στην μηχανή σχετίζεται με ένα αντικείμενο παιχνιδιού. Ένα αντικείμενο παιχνιδιού αντιπροσωπεύει γενικά ένα συγκεκριμένο αντικείμενο ή στοιχείο στο παιχνίδι σας που μπορεί να εκπέμπει ήχο, συμπεριλαμβανομένων χαρακτήρων, όπλων, αντικειμένων περιβάλλοντος, όπως όπλα, κ.λπ. Σε ορισμένες περιπτώσεις, ωστόσο, μπορεί να αντιστοιχούν σε αντικείμενα παιχνιδιού σε διαφορετικά μέρη ενός στοιχείου του παιχνιδιού. Για παράδειγμα, μπορεί να αντιστοιχίσει ένα διαφορετικό αντικείμενο παιχνιδιού σε διαφορετικά μέρη ενός γιγαντιαίου χαρακτήρα, έτσι ώστε οι ήχοι των βημάτων και η φωνή του χαρακτήρα να προέρχονται από διαφορετικές θέσεις εντός του τρισδιάστατου ηχητικού χώρου.

Για κάθε αντικείμενο του παιχνιδιού, το Wwise αποθηκεύει μια ποικιλία πληροφοριών που θα χρησιμοποιηθούν για να καθορίσει τον τρόπο αναπαραγωγής κάθε ήχου στο παιχνίδι. Οποιοδήποτε από τα παρακάτω είδη πληροφοριών μπορεί να συσχετιστεί με το αντικείμενο του παιχνιδιού:

- Θέση και προσανατολισμός 3D.
- Πληροφορίες συγχρονισμού παιχνιδιού, συμπεριλαμβανομένων καταστάσεων, Switches και RTPC (Real Time Parameter Controls).

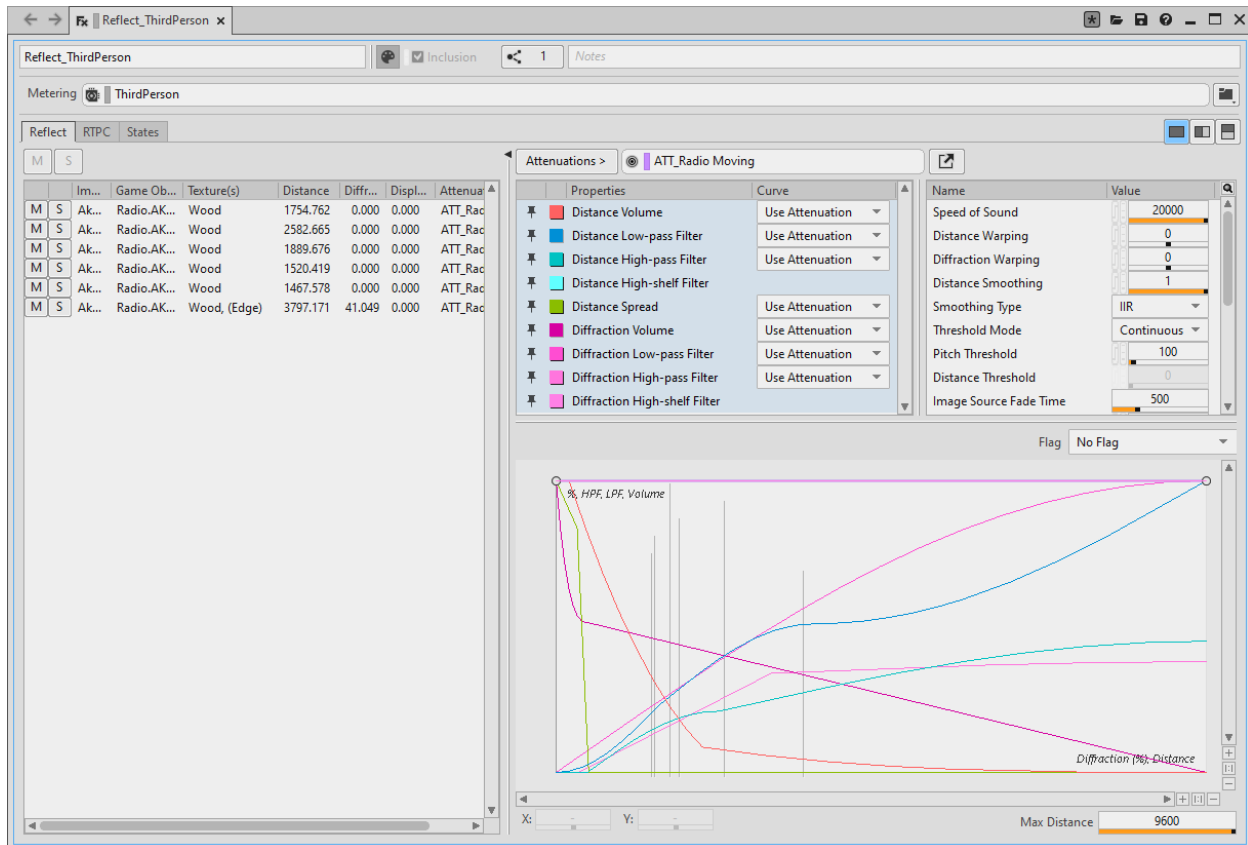
- Περιβαλλοντικά εφέ.
- Εμπόδια και απόκρυψη τυχών εμποδίων.

Χρησιμοποιώντας αντικείμενα παιχνιδιού, η διαχείριση του ήχου απλοποιείται, καθώς ακολουθούνται μόνο τα αντικείμενα παιχνιδιού και όχι οι μεμονωμένοι ήχοι.

Μόλις δημιουργηθούν τα αντικείμενα του παιχνιδιού, οι προγραμματιστές πρέπει μόνο να δημοσιεύσουν τα συμβάντα, να ρυθμίσουν τους συγχρονισμούς του παιχνιδιού, συμπεριλαμβανομένων των διακοπών, των καταστάσεων και των RTPC, καθώς και τα περιβάλλοντα εντός του παιχνιδιού. Οι συγκεκριμένες λεπτομέρειες σχετικά με το ποιος ήχος θα αναπαραχθεί και πώς θα αναπαραχθεί καθορίζονται από τον σχεδιαστή ήχου στο Wwise. Χρησιμοποιώντας αυτήν την προσέγγιση, εξοικονομείται χρόνος όταν οι προγραμματιστές ασχολούνται με το πλήθος των ήχων που σχετίζονται με τις διάφορες οντότητες εντός του παιχνιδιού.

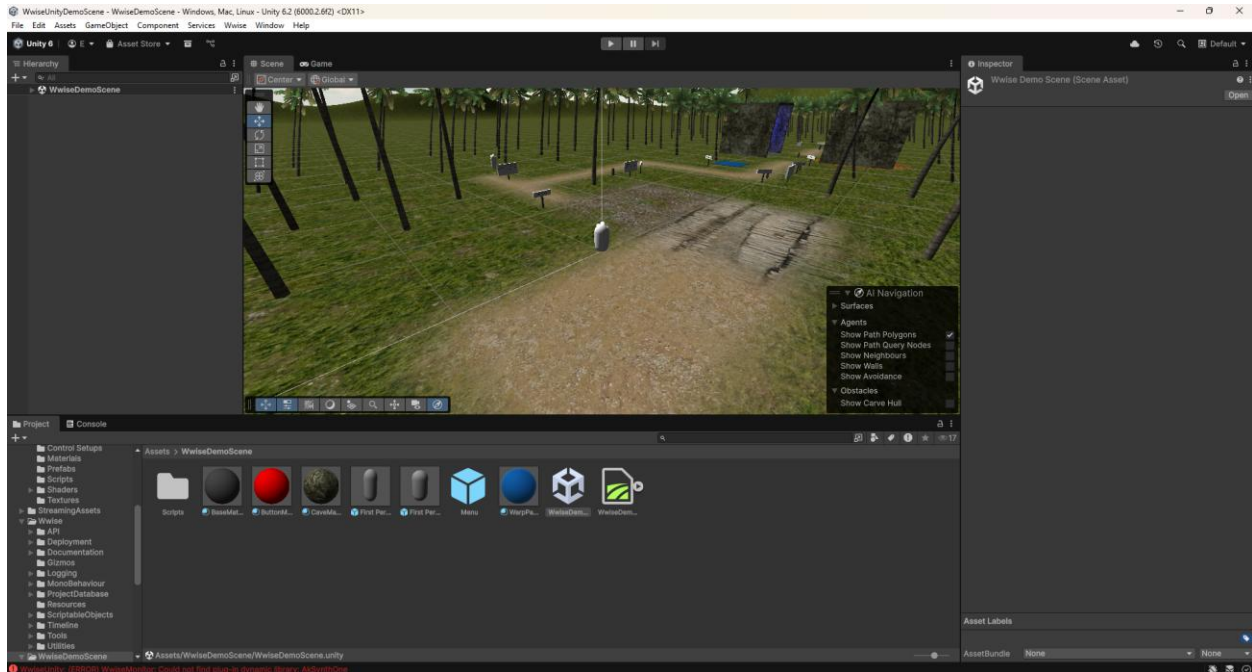
(Audiokinetic inc., 2025)

Το Wwise παρέχει και διάφορα πρόσθετα (plugins) όπως για παράδειγμα το Reflect που χρησιμοποιείται στην δημιουργία ανακλάσεων στον χώρο με βάση την εγγύτητα του ακροατή σε επιφάνειες όπως τοίχοι, δάπεδα και οροφές, αλλά οι άδειες για το καθένα κοστίζουν έξτρα και ξεφεύγουν από τους σκοπούς της διπλωματικής εργασίας.



Εικ. 3-30 Παράθυρο Wwise reflect

πηγή: www.audiokinetic.com/en/wwise/plugins/reflect/



Εικ. 3-31 Παράθυρο Unity Editor

Εδώ βλέπουμε το παράθυρο του Unity 6 που τρέχει το δοκιμαστικό project για τις δυνατότητες του Wwise. Η Audiokinetic παρέχει μία σειρά από μαθήματα για την χρήση του Wwise με δυνατότητα απόκτησης πιστοποίησης με εξετάσεις. Ένα από αυτά είναι το Wwise 301 για την ενσωμάτωση του Wwise στο Unity που μας βοηθάει στην καλύτερη κατανόηση να ενσωματώσουμε το Wwise στο Unity αλλά χρειάζεται ειδική άδεια (License) για να δουλέψει κανονικά το πρότζεκτ. Το δοκιμαστικό παιχνίδι λέγεται “The Wwise adventure game”. Σε αυτό το υποθετικό παιχνίδι ένας παίκτης βρίσκεται σε ένα χωριό και παρέχεται μία λίστα από εκπληρώσεις (Quest) που πρέπει να εκπληρωθούν. Μπορεί και ο παίκτης να τηλεμεταφερθεί και σε άλλες τοποθεσίες.



Εικ. 3-32 Εικόνα The Wwise adventure game

πηγή: <https://www.audiokinetic.com/en/blog/wwise-301-certification-is-live/>

3.2.3 Τράπεζες ήχων (Soundbanks)

Το Wwise παρέχει έναν τρόπο σχετικά με την διαχείριση των ήχων σε ένα παιχνίδι και λέγονται Soundbanks. Υπάρχουν δύο τύποι, το Initialization (Init) bank και το Soundbank. Το Initialization (Init) bank είναι ειδική τράπεζα που δημιουργείται αυτόματα κατά την αρχικοποίηση του project και περιέχει πληροφορίες σχετικά με την ιεραρχία του Bus, τις καταστάσεις (States), τους διακόπτες (Switches) και τα RTPC, επίσης μπορεί να αποθηκεύσει δεδομένα συσκευών ήχου (Audio Device Sharesets). Φορτώνεται μία φορά στην αρχή του παιχνιδιού.

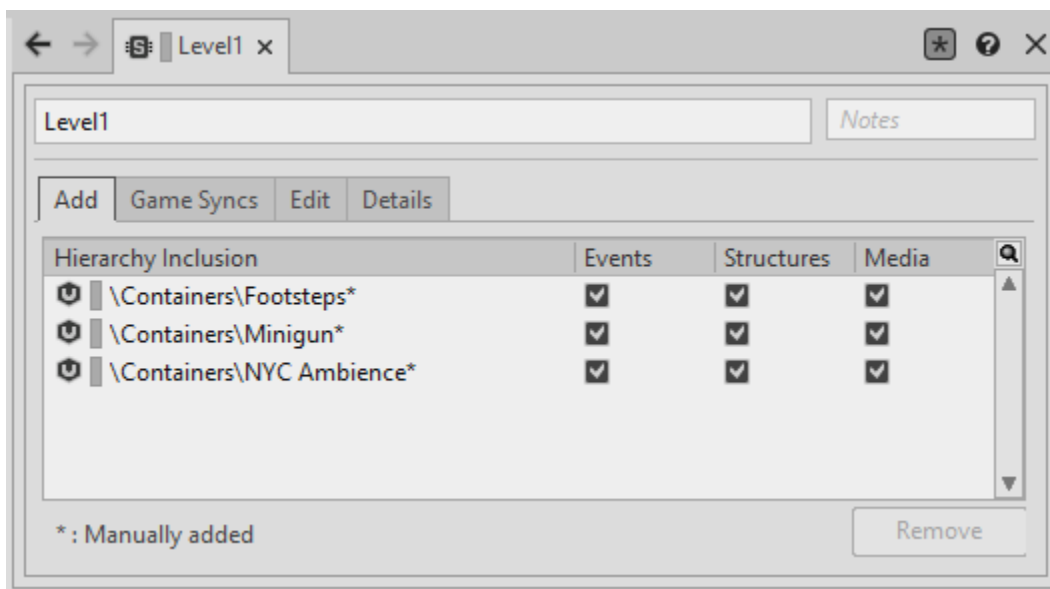
(Audiokinetic inc., 2025)

Ενώ τα Soundbanks είναι αρχεία που περιέχουν πληροφορίες γεγονότος, αρχεία ήχου και μουσικής και δεδομένα δομής κίνησης, φορτώνονται και εκφορτώνονται σε διαφορετικά σημεία του παιχνιδιού για καλύτερη αξιοποίηση της μνήμης της πλατφόρμας.

Η επεξεργασία και η διαχείριση τους γίνεται μέσω του παραθύρου “SoundBank Editor” όπου παρακάτω γίνεται η επεξήγηση των λειτουργιών σε καρτέλες (Tabs)

- Προσθήκη (Add): Εμφανίζει μόνο τα πραγματικά Events(Συμβάντα), τις ιεραρχίες αντικειμένων, τις Μονάδες Εργασίας(Work Units) και τους φακέλους που προστέθηκαν στο SoundBank. Οποιαδήποτε αντίστοιχα θυγατρικά αντικείμενα που προστίθενται επίσης αυτόματα στο SoundBank εμφανίζονται μόνο στην καρτέλα Edit (επεξεργασία). Στην καρτέλα Προσθήκη, καθορίζονται ποιοι τύποι πληροφοριών ή/και πολυμέσων περιλαμβάνονται στο SoundBank ανά ιεραρχικό στοιχείο.

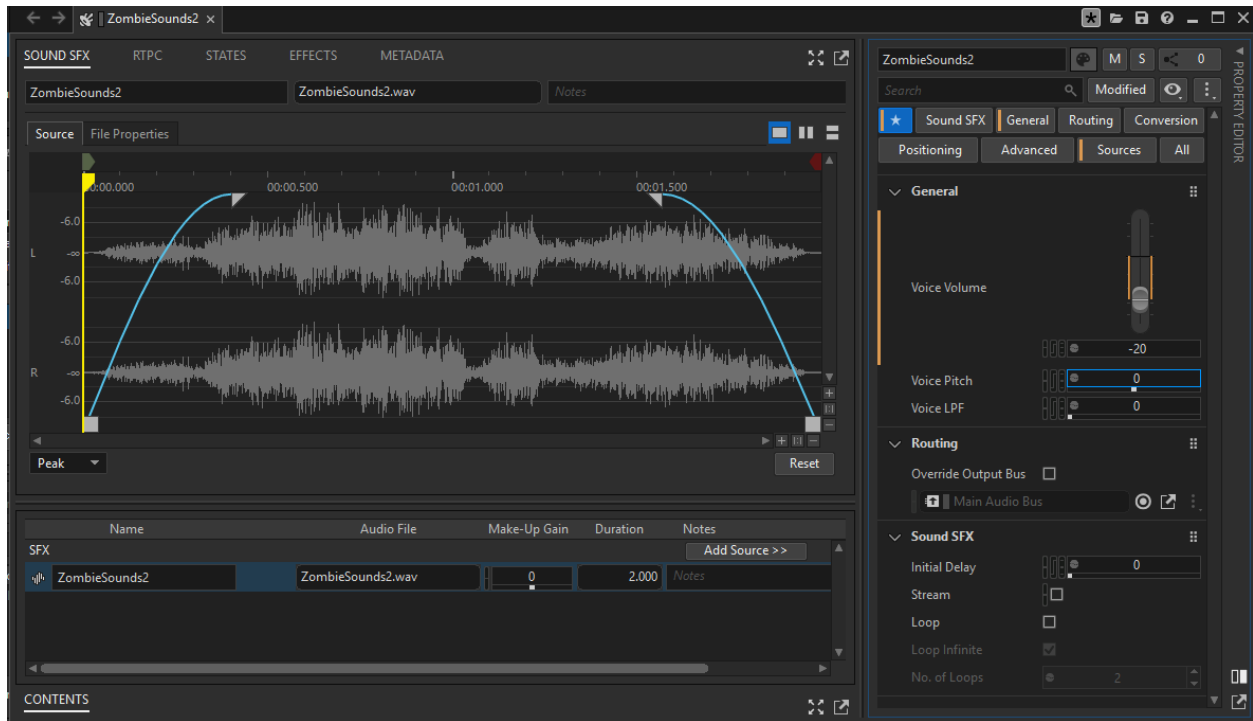
- **Game Syncs:** Εμφανίζει μία λίστα States(Καταστάσεις), Switches(Διακόπτες) και Triggers(Ενεργοποιητές): σχετιζόμενες με τα γεγονότα και δομές ήχου που έχουν προστεθεί στην καρτέλα της προσθήκης.
- **Επεξεργασία (Edit):** Εμφανίζει μία λίστα για κάθε ξεχωριστό γεγονός (Event), δομές αντικειμένων και αρχεία μέσων συμπεριλαμβανομένων όλων των θυγατρικών αντικειμένων σχετιζόμενα με τα στοιχεία του Project στην καρτέλα προσθήκης(Add). Επιπλέον πληροφορίες εμφανίζονται όπως μορφή αρχείου ήχου, μέγεθος αρχείου και ρυθμός δειγματοληψίας.
- **Λεπτομέρειες (Details):** Εμφανίζει αναλυτικές πληροφορίες σχετικά με το μέγεθος των διαφορετικών στοιχείων μέσα από το επιλεγμένο Soundbank καθώς και τυχόν αρχεία που λείπουν ή έχουν αντικατασταθεί.



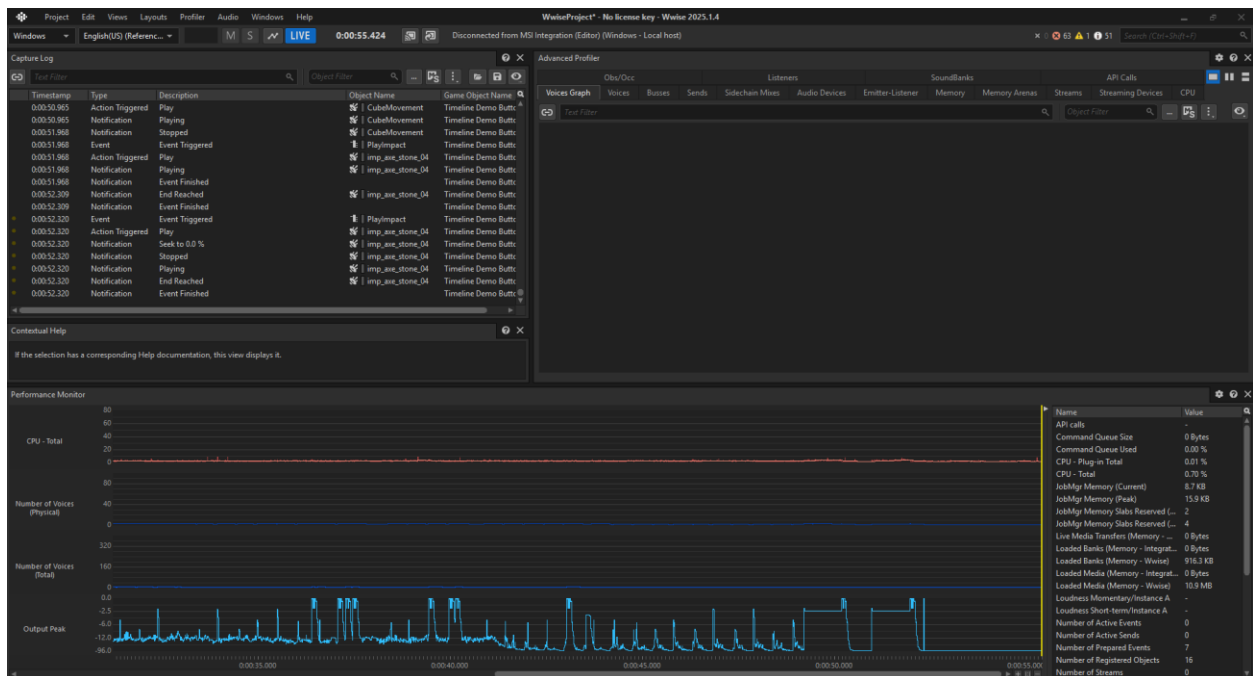
Εικ. 3-33 Παράθυρο Soundbank με την καρτέλα Add.

Πηγή: https://www.audiokinetic.com/en/public-library/2025.1.4_9062/?source=Help&id=managing_soundbanks

Παρακάτω βλέπουμε ένα παράθυρο που μπορούμε να κάνουμε επεξεργασία σε έναν ήχο και να ρυθμίσουμε διάφορες παραμέτρους όπως καμπύλη ξεθωριάσματος και δρομολόγηση σχετιζόμενη με την θέση του ακροατή.



Εικ. 3-34 Παράθυρο επεξεργασίας ήχου



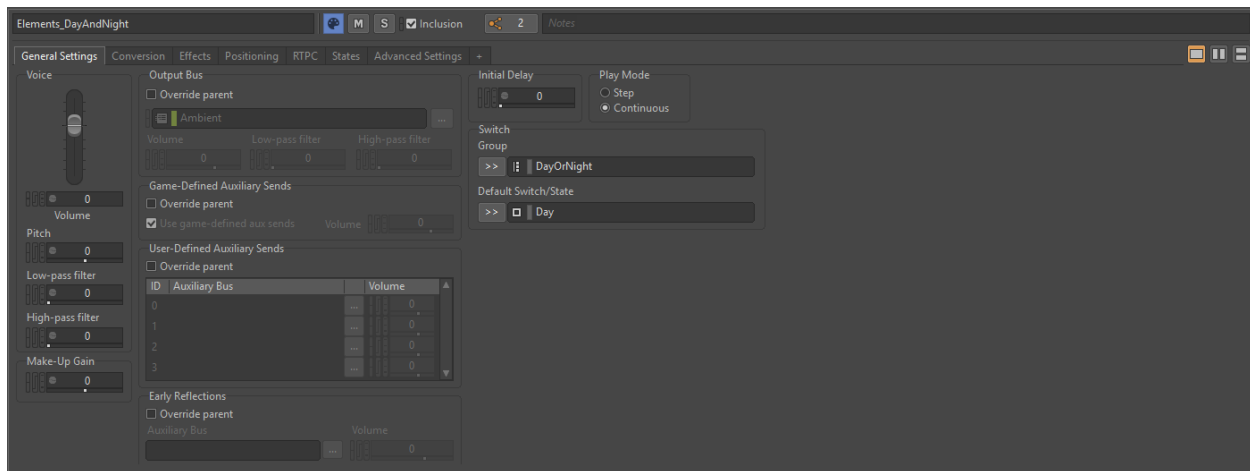
Εικ. 3-35 Παράθυρο profiler του Wwise

3.2.4 Δίαυλος ήχου (Bus)

Αποτελείται από μία ομάδα αντικειμένων που συνδέει πηγές ήχου σε προορισμούς που με ενσωματωμένο δείκτη που γίνεται η επεξεργασία των ήχων μέσω του Wwise. Χωρίζονται σε δύο κατηγορίες τα Audio Bus και τα Auxillary Bus.

Τα Audio Bus (Δίαυλοι ήχου) χρησιμοποιούνται στην ομαδοποίηση των δομημένων αντικειμένων ήχων σε ένα παιχνίδι.

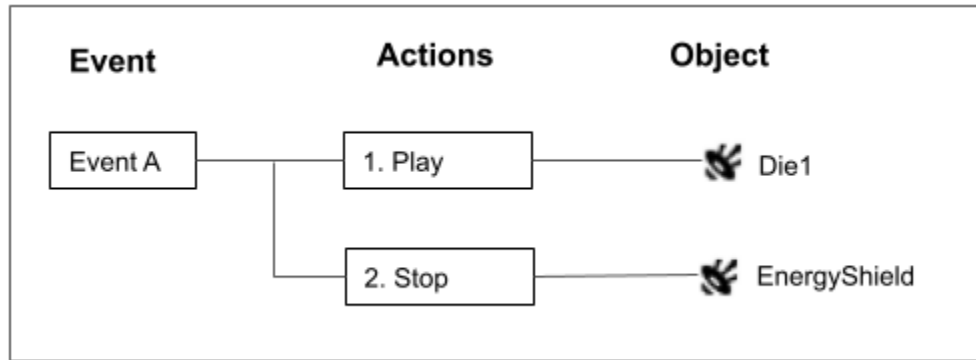
Τα Auxillary Bus (Βοηθητικοί δίαυλοι) χρησιμοποιούνται για την ρύθμιση της έντασης, της διαμόρφωσης του διαύλου, της θέσης και του RTPC, καθώς και για την εφαρμογή εφέ ή καταστάσεων πριν από την επιστροφή σε έναν γονικό δίαυλο ήχου. Οι ρυθμίσεις Ducking, HDR και μίξης φωνής δεν είναι δυνατές σε έναν βοηθητικό δίαυλο.



Εικ. 3-36 Παράθυρο Audio Bus

3.2.5 Γεγονότα (Events)

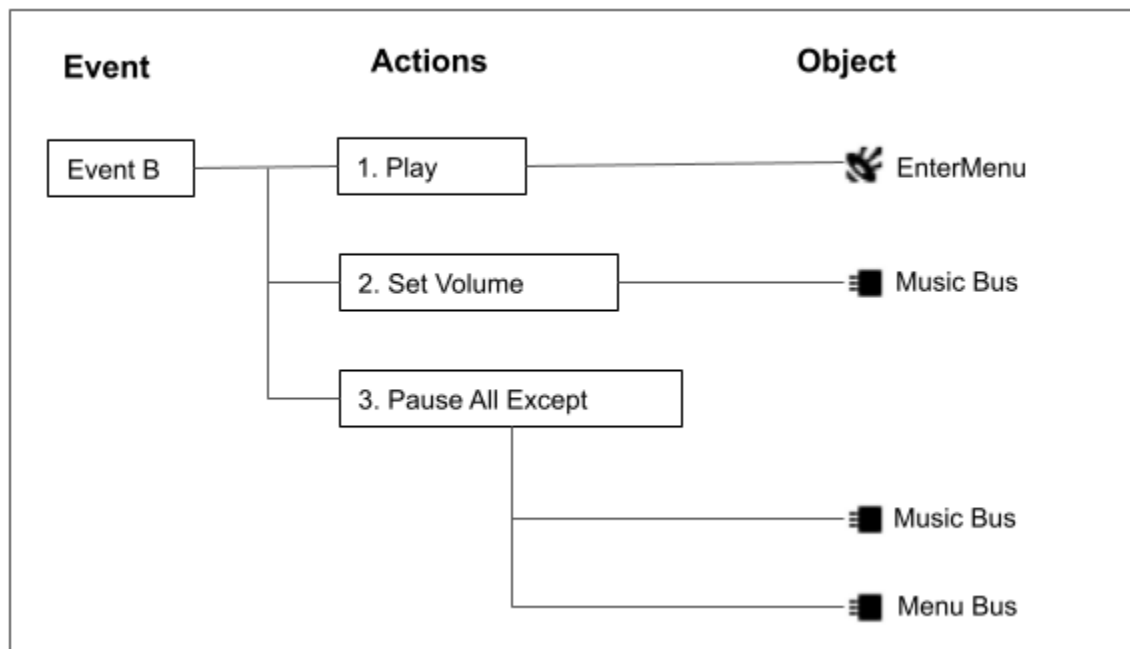
Τα Events είναι μία σειρά από γεγονότα παιχνιδιού που το καθένα συνδυάζονται με ένα αντικείμενο ήχου ή με ένα Bus. Οι δράσεις που επιλέγονται καθορίζουν ποια αντικείμενα του Wwise θα εκτελέσουν, σταματήσουν ή κάτι άλλο. Για παράδειγμα δημιουργείται ένα γεγονός σε ένα παιχνίδι βολών (Shooter) με έναν παίκτη που σκοτώνεται. Θα παιχτεί το γεγονός που ο παίκτης πεθαίνει και θα σταματήσει να εκτελείται το αντικείμενο της ασπίδα του παίκτη.



Σχήμα 3-2 Σχεδιάγραμμα γεγονότος Event στο Wwise.

Πηγή: https://www.audiokinetic.com/en/public-library/2025.1.5_9095/?source=WwiseFundamentalApproach&id=understanding_events

Σε ένα άλλο γεγονός όταν ο παίκτης επιστρέφει πίσω στο μενού, εκτελείται μία σειρά εντολών. Παίζει το αντικείμενο ήχου, μειώνεται η ένταση του μουσικού bus και σταματάνε τα υπόλοιπα bus.

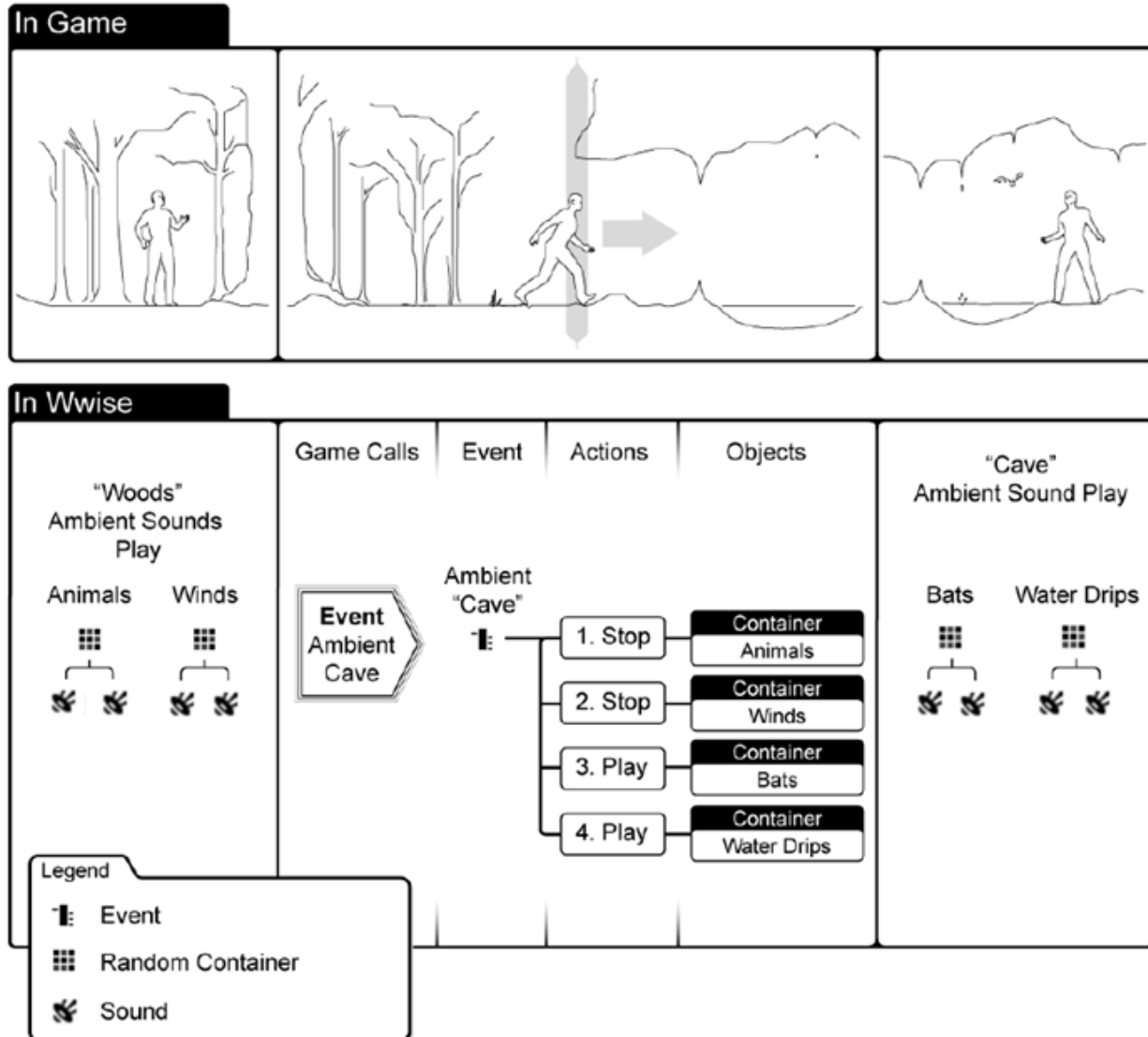


Σχήμα 3-3 Σχεδιάγραμμα γεγονότος για την επιστροφή του παίκτη στο μενού

Πηγή: https://www.audiokinetic.com/en/public-library/2025.1.5_9095/?source=WwiseFundamentalApproach&id=understanding_events

Υπάρχουν δύο τύποι γεγονότων στο Wwise, της δράσης (Actions) που εκτελούνται τα ηχητικά εφέ και του διαλόγου που παρέχει και δέντρο αποφάσεων με καταστάσεις (Decision tree) και διακόπτες (Switches).

Για παράδειγμα σε ένα γεγονός δράσης, ο παίκτης μπαίνει σε μία σπηλιά, σταματάει τα περιεχόμενα (Container) των ήχων των ζώων και των ανέμων και παίζει αυτά των νυχτερίδων και των σταγόνων νερών από τους σταλακτίτες όπως φαίνεται παρακάτω στο διάγραμμα.

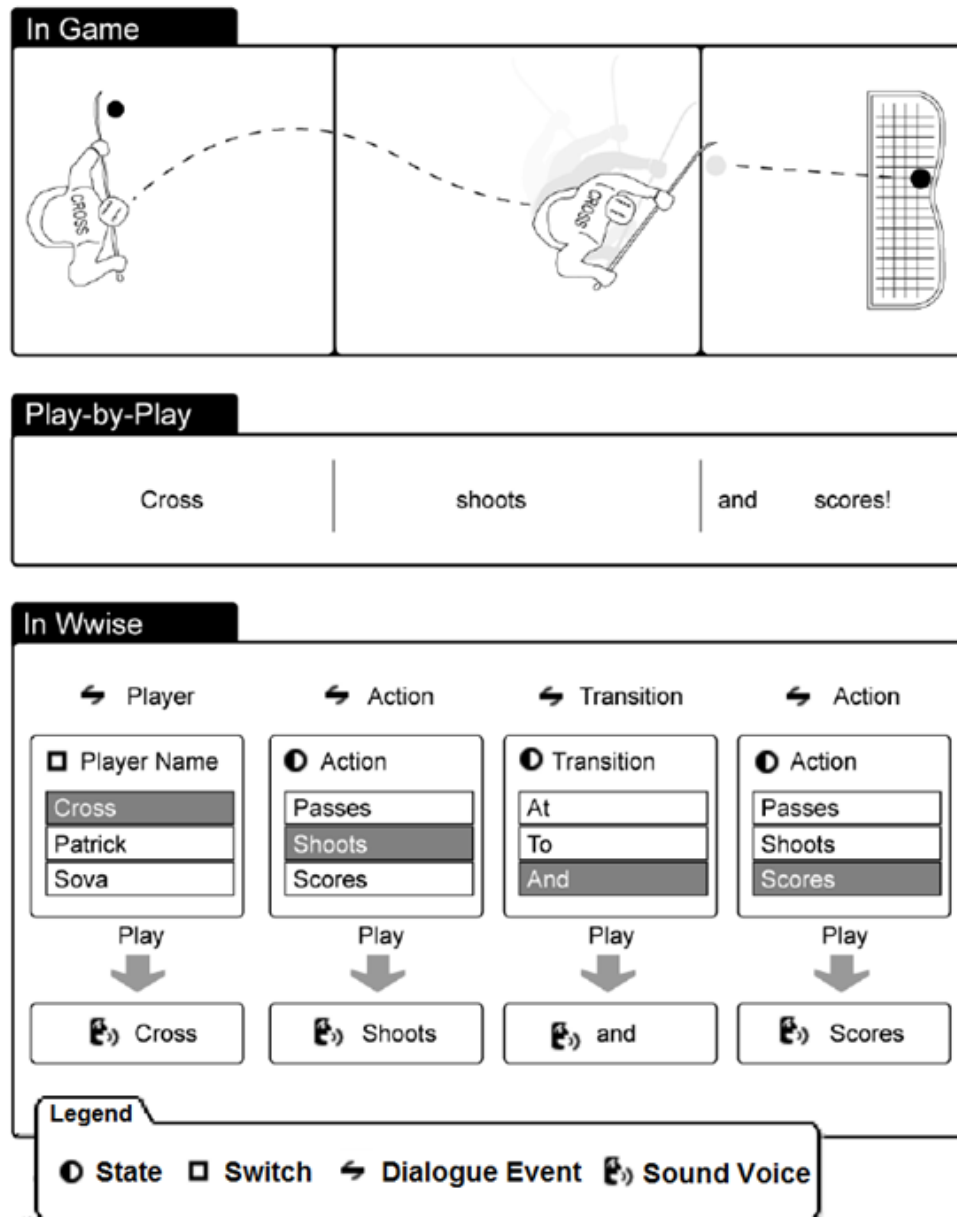


Σχήμα 3-4 Παράδειγμα γεγονότος δράσης στο Audiokinetic Wwise

Πηγή: https://www.audiokinetic.com/en/public-library/2025.1.5_9095/?source=WwiseFundamentalApproach&id=understanding_events

Στο παράδειγμα γεγονότος διαλόγου, όταν ένας παίκτης σουτάρει και σκοράρει, ο σχολιασμός κάθε φορά πρέπει να αντιστοιχεί στην ενέργεια του παιχνιδιού. Για να ρυθμίσετε τις διαφορετικές δυνατότητες και τα αποτελέσματα στο Wwise, θα χρειαστεί να δημιουργήσετε Συμβάντα Διαλόγου για παίκτες, ενέργειες, μεταβάσεις και ούτω καθεξής. Κάθε ένα από αυτά τα Συμβάντα περιέχει ένα σύνολο αντίστοιχων τιμών Κατάστασης και διακοπής για το παιχνίδι. Δημιουργείται μια διαδρομή Κατάστασης/Διακοπής που ορίζει κάθε συνθήκη ή αποτέλεσμα και

στη συνέχεια να αντιστοιχεί ένα κατάλληλο φωνητικό αντικείμενο σε κάθε διαδρομή Κατάστασης/Διακοπής. Κατά τη διάρκεια του παιχνιδιού, το παιχνίδι θα αντιστοιχίσει τις τρέχουσες τιμές Κατάστασης/Διακοπής με τις διαδρομές που ορίστηκαν στο Wwise για να καθορίσει ποιο φωνητικό αντικείμενο θα αναπαραχθεί.



Σχήμα 3-5 Παράδειγμα γεγονότος διαλόγου στο Wwise

Πηγή: https://www.audiokinetic.com/en/public-library/2025.1.5_9095/?source=WwiseFundamentalApproach&id=understanding_events

3.2.6 Έλεγχος παραμέτρων σε πραγματικό χρόνο (Real-Time Parameter Controls - RTPC)

Οι έλεγχοι παραμέτρων σε πραγματικό χρόνο (RTPC) είναι χρήσιμοι για τον έλεγχο στις ιδιότητες διαφόρων αντικειμένων του Wwise (ήχοι, κοντέινερ, διαύλοι ελέγχου, εφέ κ.λπ.) με βάση τις αλλαγές παραμέτρων που συμβαίνουν στο παιχνίδι σε πραγματικό χρόνο.

Οι ήχοι αυτοί μπορούν να τοποθετηθούν τυχαία σε πολλά ηχεία, να χρησιμοποιηθεί κώδικα τυχαίας τοποθέτησης για να οδηγηθεί το RTPC στην αμφιωτική πανοραμική προβολή των τρισδιάστατων ήχων που ορίζονται από τον χρήστη, αντί των προκαθορισμένων διαδρομών που έχουν δημιουργηθεί στο Wwise.

Ο σχεδιαστής ήχου ορίζει τα ονόματα και τα εύρη των παραμέτρων του παιχνιδιού στο Wwise Authoring. Οι σχεδιαστές ορίζουν επίσης τις καμπύλες RTPC. Η τιμή μιας παραμέτρου του παιχνιδιού αντιπροσωπεύει τον άξονα X μιας καμπύλης RTPC και το τύπο αντικείμενου σε άξονα Y.

Αφού ο σχεδιαστής ήχου ορίσει τις παραμέτρους του παιχνιδιού και τις καμπύλες RTPC στο Wwise Authoring, το παιχνίδι χρειάζεται μόνο να ενημερώσει την τιμή των διαφόρων παραμέτρων του παιχνιδιού κατά τη διάρκεια του παιχνιδιού. Η μηχανή ήχου υπολογίζει την τιμή της ιδιότητας αξιολογώντας την καμπύλη RTPC για την τρέχουσα τιμή της παραμέτρου του επιμέρους στοιχείου του παιχνιδιού.

Υπάρχουν δύο είδη μεταβλητών, οι καθολικοί (Global) και οι τοπικοί (local). Οι καθολικοί παράμετροι ορίζουν τιμές που επηρεάζει όλους τους ήχους στο παιχνίδι ομοιόμορφα και οι τοπικοί παράμετροι για έναν επιμέρους αντικείμενο σε ένα παιχνίδι.

Οι ενσωματωμένοι παράμετροι είναι οι:

Παράμετρος	Περιγραφή
Distance (απόσταση)	Η απόσταση μεταξύ του αντικείμενου του παιχνιδιού και του ακροατή. Σε περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί περισσότεροι από ένας ακροατές ή/και περισσότερες από μία θέσεις, η τιμή που λαμβάνεται υπόψη είναι η μικρότερη απόσταση μεταξύ όλων των συνδυασμών ακροατή και θέσης ήχου.
Azimuth (Αζιμούθιο)	Η γωνία, σε μοίρες, μεταξύ του ακροατή και του αντικείμενου του παιχνιδιού, όπως προβάλλεται στο οριζόντιο επίπεδο. Μια τιμή 0 μοιρών υποδηλώνει ότι ο ήχος βρίσκεται ακριβώς μπροστά από τον ακροατή, -90 μοίρες ότι ο ήχος βρίσκεται αριστερά, 90 μοίρες ότι βρίσκεται δεξιά και +/- 180 μοίρες

	ότι ο ήχος βρίσκεται ακριβώς πίσω από τον ακροατή. Σε περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές και/ή θέσεις ήχου, η τιμή που λαμβάνεται είναι η γωνία μεταξύ του ακροατή και της θέσης ήχου που βρίσκονται πιο κοντά μεταξύ τους
Elevation (Ανύψωση)	Η κατακόρυφη γωνία, σε μοίρες, ως προς τον ορίζοντα (άξονα XY), μεταξύ του ακροατή και του αντικειμένου του παιχνιδιού. Μια τιμή 0 μοιρών υποδηλώνει ότι ο ήχος βρίσκεται στο ίδιο οριζόντιο επίπεδο με τον ακροατή· μια τιμή 90 μοιρών υποδηλώνει ότι ο ήχος βρίσκεται ακριβώς από πάνω, ενώ -90 μοίρες υποδηλώνει ότι ο ήχος βρίσκεται ακριβώς από κάτω. Σε περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές και/ή θέσεις ήχου, η τιμή που λαμβάνεται είναι η γωνία μεταξύ του ακροατή και της θέσης ήχου που βρίσκονται πιο κοντά μεταξύ τους.
Emitter Cone (Κώνος εκπομπής)	Αντιπροσωπεύει τη γωνία 3D μεταξύ του διανύσματος προσανατολισμού του εκπομπού και του διανύσματος που σχηματίζεται από την ευθεία μεταξύ του εκπομπού και του ακροατή. Μια τιμή 0 μοιρών υποδηλώνει ότι ο εκπομπός βρίσκεται ακριβώς απέναντι από τον ακροατή, ενώ μια τιμή 180 μοιρών υποδηλώνει ότι ο εκπομπός βρίσκεται ακριβώς στην αντίθετη κατεύθυνση από τον ακροατή. Στην περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές και/ή θέσεις ήχου, η τιμή που λαμβάνεται είναι η γωνία μεταξύ του ακροατή και της θέσης ήχου που βρίσκονται πιο κοντά μεταξύ τους.
Obstruction (Εμπόδιο)	Σε περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές, η τιμή παρεμπόδισης που

	λαμβάνεται είναι αυτή που έχει αντιστοιχιστεί στον ακροατή με την πλησιέστερη τρέχουσα θέση ήχου.
Occlusion (Απόκρυψη)	Σε περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές, η τιμή απόκρυψης που λαμβάνεται είναι αυτή που έχει αντιστοιχιστεί στον ακροατή με την πλησιέστερη τρέχουσα θέση ήχου.
Listener Cone (Κώνος ακροατή)	Αντιπροσωπεύει τη τρισδιάστατη γωνία μεταξύ του διανύσματος προσανατολισμού του ακροατή και του διανύσματος που σχηματίζεται από την ευθεία μεταξύ του εκπομπού και του ακροατή. Μια τιμή 0 μοιρών υποδηλώνει ότι ο ακροατής βρίσκεται ακριβώς απέναντι από τον εκπομπό, ενώ μια τιμή 180 μοιρών υποδηλώνει ότι ο ακροατής βρίσκεται ακριβώς στην αντίθετη κατεύθυνση από τον εκπομπό. Στην περίπτωση που στο αντικείμενο του παιχνιδιού έχουν αντιστοιχιστεί πολλοί ακροατές και/ή θέσεις ήχου, η τιμή που λαμβάνεται είναι η γωνία μεταξύ του ακροατή και της θέσης ήχου που βρίσκονται πιο κοντά μεταξύ τους.
Diffraction (Περίθλαση)	Με δωμάτια και πύλες, τα αντικείμενα παιχνιδιού εκπομπής λαμβάνουν μια τιμή που σχετίζεται με τη «ξηρή» διάθλασή τους: τη γωνία απόκλισης, σε μοίρες, από μια ευθεία διαδρομή μεταξύ του εκπομπού και του ακροατή. Τα αντικείμενα παιχνιδιού δωματίου που έχουν καταχωρηθεί εσωτερικά από το Spatial Audio λαμβάνουν επίσης μια τιμή διάθλασης, η οποία όμως σχετίζεται με τη «υγρή» διάθλασή τους — τη διάθλαση του πεδίου διάχυτης κράτησης των ήχων που βρίσκονται μέσα στο δωμάτιο. Η υγρή διάθλαση είναι η γωνία απόκλισης, σε μοίρες, από την κάθετο προς το άνοιγμα μιας Πύλης. Σε περίπτωση που υπάρχουν πολλαπλές Πύλες ή πολλαπλές διαδρομές που μπορεί να ακολουθήσει ο ήχος για να φτάσει στον ακροατή, επιλέγεται η μικρότερη γωνία

	διάθλασης μεταξύ των διαφόρων διαδρομών. Η τιμή διάθλασης μπορεί να κυμαίνεται από 0 έως 100, η οποία αντιπροσωπεύει ένα ποσοστό διάθλασης, αντί για μοίρες. Απαιτεί ενεργοποίηση λειτουργίας μετάδοσης στο Property Editor.
Απώλεια μετάδοσης	Η απώλεια μετάδοσης υπολογίζεται για την ακτίνα που συνδέει απευθείας τον πομπό και τον ακροατή. Η απώλεια μετάδοσης της γεωμετρίας ορίζεται στη γεωμετρία ενώ η απώλεια μετάδοσης των δωματίων ορίζεται στα δωμάτια. Η τιμή της απώλειας κυμαίνεται από 0 έως 100. Η υψηλότερη τιμή χρησιμοποιείται σε όλα τα δωμάτια του πομπού και του ακροατή, καθώς και σε όλες τις επιφάνειες γεωμετρίας που διαχέουν.

Πίνακας 3-3 Πίνακας ενσωματωμένων (Built in) παραμέτρων RTPC του Wwise

Παρακάτω βλέπουμε ένα παράδειγμα μεταβλητή RTPC σε ένα δοχείο (container) ήχων, όσο μεγαλώνει η απόσταση μεταξύ του παίκτη και του εχθρού, τόσο λιγότερο ακούγονται και μπορεί να χρησιμοποιηθεί ο κέρσορας για να παίξει ο ήχος. Στην συνάρτηση `distanceRTPC.SetValue(gameObject, distanceToPlayer)`, το αντικείμενο `distanceRTPC` ορίζεται ο ήχος από την απόσταση του παίκτη από τον εχθρό. Μπορούμε να ορίσουμε και άλλες μεταβλητές για να δούμε πως θα λειτουργήσει στο παιχνίδι.



Στο παράθυρο “Profiler” του Wwise βλέπουμε τους ήχους που εκτελούνται κατά την διάρκεια της εκτέλεσης του παιχνιδιού σε χρονικές στιγμές με την σειρά, την επίδοση της CPU (κεντρική μονάδα επεξεργασίας) και τις πληροφορίες των συσκευών που συνδέονται. Για να ενεργοποιηθεί το Profiler, θα πρέπει πρώτα το Remote Platform να συνδεθεί με τον υπολογιστή και συγκεκριμένα με την μηχανή παιχνιδιού που εκτελείται το παιχνίδι που καλεί το API του Wwise.



Πηγή: <https://www.audiokinetic.com/en/blog/profiling-troubleshooting-and-debugging-using-wwise/>

Το profiler αποτελείται από έναν πίνακα επίδοσης (Performance Monitor) που παρακολουθεί το φορτίο του CPU, τον αριθμό των φωνών (φυσικές και συνολικές) και την ένταση του Output (σε Decibel)

3.2.8 Πίεση και αποσυμπίεση αρχείων ήχων

Τα αρχεία ήχου των παιχνιδιών είχαν πάντα την ανάγκη για συμπίεση. Οφείλεται ότι δεν επαρκεί ο χώρος στο δίσκο ή στη μνήμη για να συμπεριληφθεί όλο το ηχητικό περιβάλλον που θεωρούμε ως ακατέργαστα, μη συμπιεσμένα ηχητικά δείγματα.

Κάθε κωδικοποιητής ήχου έχει τα πλεονεκτήματα και τα μειονεκτήματά του. Δεν είναι πάντα προφανές ποιος κωδικοποιητής πρέπει να χρησιμοποιηθεί σε κάθε περίπτωση. Το πρώτο πράγμα που πρέπει να ληφθεί υπόψιν είναι ότι δεν υπάρχει μία και μοναδική σωστή επιλογή για όλες τις πλατφόρμες όπως δεν υπάρχει καν μία και μοναδική σωστή επιλογή για όλους τους τύπους ήχων σε μία και μοναδική πλατφόρμα.

Το Wwise υποστηρίζει 4 κωδικοποιητές λογισμικού: PCM, ADPCM, Vorbis και Opus. Υπάρχει ένα κύριο πλεονέκτημα στη χρήση αυτών των κωδικοποιητών, που είναι η συνέπεια μεταξύ των πλατφορμών. Θα έχουν ακριβώς την ίδια ποιότητα, σχετική απόδοση και συμπεριφορά σε όλες τις πλατφόρμες. Όλοι υποστηρίζουν το πλήρες σύνολο χαρακτηριστικών του Wwise για πηγές ήχου, κάτι που δεν ισχύει για τους κωδικοποιητές υλικού.

Επιπλέον, κάθε κονσόλα παιχνιδιών που δημιουργήθηκε συνοδεύεται από έναν επεξεργαστή ήχου υλικού όπου το καθένα έχει διαφορετικές δυνατότητες, οι οποίες δεν είναι ισοδύναμες μεταξύ των πλατφορμών. Το Wwise υποστηρίζει την αποκωδικοποίηση υλικού συμπιεσμένων πηγών και μερικές φορές περισσότερη επεξεργασία υλικού, όταν είναι διαθέσιμη στην πλατφόρμα.

Ο λόγος συμπίεσης είναι απλά ο λόγος του τελικού μεγέθους του αρχείου προς το αρχικό μέγεθος του αρχείου. Για τους κωδικοποιητές ήχου, συμπεριλαμβάνεται στον υπολογισμό μόνο το τμήμα των δειγμάτων ήχου του αρχείου, δηλαδή χωρίς το κοντέινερ WEM ή τους δείκτες. Εξαιρούνται επίσης τυχόν προηγούμενες υποδειγματοληψίες. Ο παρακάτω πίνακας συγκρίνει την αποτελεσματικότητα συμπίεσης κάθε κωδικοποιητή. Για τους κωδικοποιητές μεταβλητής ποιότητας, το περιεχόμενο ήχου και οι ρυθμίσεις παίζουν αναμφίβολα τεράστιο ρόλο στα αποτελέσματα.

Κωδικοποιητής ήχου	Λόγος συμπίεσης	Μεταβλητός ή σταθερός ρυθμός μετάδοσης
PCM	1:1	Σταθερός
ADPCM	4:1	Σταθερός
Vorbis	2:1 - ~40:1	Σταθερός ή Μεταβλητός
Opus	2:1 - ~60:1	Σταθερός

Πίνακας 3-4 Πίνακας υποστηριζόμενων κωδικοποιητών ήχου για Wwise

Ο παρακάτω πίνακας δείχνει την απόδοση των κωδικοποιητών λογισμικού, όσον αφορά τη διακίνηση δεδομένων. Με άλλα λόγια, είναι ο μέγιστος αριθμός ροών που μπορούν να αποκωδικοποιηθούν σε πραγματικό χρόνο, σε έναν πυρήνα. Ο αριθμός αυτός περιλαμβάνει μόνο τον ίδιο τον αποκωδικοποιητή, χωρίς επαναδειγματοληψία, επιβάρυνση διαχείρισης φωνής, διακοπές, συμφόρηση διαύλου μνήμης κ.λπ. Εν ολίγοις, ο αποκωδικοποιητής είναι απομονωμένος στις καλύτερες δυνατές συνθήκες. Ο σκοπός είναι να δοθεί μια ιδέα της μέγιστης απόδοσης, αλλά τα ανώτατα όρια είναι απίθανο να επιτευχθούν σε ένα πραγματικό περιβάλλον παιχνιδιού. Η απόδοση μπορεί να συγκριθεί με τις μέγιστες υποστηριζόμενες ροές για τους κωδικοποιητές υλικού (παρακάτω). Για τα Vorbis και Opus, οι 3 αριθμοί αντιστοιχούν στις ρυθμίσεις χαμηλότερης, προεπιλεγμένης και υψηλότερης ποιότητας, δίνοντας το πλήρες φάσμα της απόδοσης.

Το Vorbis είναι ένα πλήρως ανοιχτό, μη ιδιόκτητο, χωρίς πατέντες και δικαιώματα, γενικής χρήσης συμπίεμένο format ήχου για ήχο και μουσική μέσης έως υψηλής ποιότητας (8kHz-48.0kHz, 16+ bit, πολυφωνικό) με σταθερά και μεταβλητά bitrates από 16 έως 128 kbps/κανάλι. Αυτό τοποθετεί το Vorbis στην ίδια ανταγωνιστική κατηγορία με άλλα πρότυπα ήχου όπως το MPEG-4 (AAC) και είναι παρόμοιο, αλλά με υψηλότερη απόδοση από το MPEG-1/2 audio layer 3, το MPEG-4 audio (TwinVQ), το WMA και το PAC.

Το Opus είναι ένας ανοιχτός, χωρίς δικαιώματα, εξαιρετικά ευέλικτος κωδικοποιητής ήχου, είναι μονόδρομος για διαδραστική μετάδοση ομιλίας και μουσικής μέσω του Διαδικτύου, αλλά προορίζεται επίσης για εφαρμογές αποθήκευσης και streaming.

Πλατφόρμα	ADPCM	Vorbis (χαμηλή-μεσαία-υψηλή ποιότητα)	WEM Opus (χαμηλή-μεσαία-υψηλή ποιότητα)
Mac(με Mac M1)	10700	7500 - 5900 – 3100	1600 - 1200 – 700
Mobile(με ARM64 Cortex A9)	8600	5200 - 4300 – 2100	1100 - 800 – 500
PC(με Intel Core i7 του 2018)	9500	5700 - 4600 – 2300	1300 - 1000 – 500
Playstation 4	3200	1200 - 1000 – 500	300 - 200 – 100
Playstation 5	7600	5900 - 4700 - 2500 (SW) 80 (HW)	80 (HW)
Nintendo Switch	2000	600 - 500 – 300	200 - 150 - 100 (SW) 20 (HW) ⁴
Xbox One	10000	1000 - 3400 – 1200	300 - 200 – 100
Xbox Series X	9200	6900 - 5400 – 2800	300 (HW)

Πίνακας 3-5 Πίνακας συμπίεσης ήχου ανά πλατφόρμα

PCM

Το PCM είναι ένα απλό μη συμπίεμένο μέσο, που το μόνο πλεονέκτημά του είναι η ταχύτητα και το μειονέκτημα είναι ότι καταλαμβάνει πολύ χώρο στο δίσκο και στη μνήμη, δεδομένου ότι

δεν είναι συμπίεσμένο. Η μόνη συνιστάμενη χρήση είναι για ήχους που χρησιμοποιούνται συχνά σε πλατφόρμες με λιγότερες δυνατότητες. Σήμερα, δεν χρησιμοποιείται εκτενέστερα ενώ υπάρχουν άλλες επιλογές κωδικοποιητή.

ADPCM

Το ADPCM είναι γνωστό στον κλάδο των βιντεοπαιχνιδιών ως ένας από τους πρώτους προσιτούς κωδικοποιητές (από άποψη CPU) στις παλιές κονσόλες. Έχει σταθερό λόγο συμπίεσης 4:1 και αποκωδικοποιείται πολύ γρήγορα. Το κύριο μειονέκτημά του είναι η αβέβαιη ποιότητα. Οι ήχοι με εμφανή μεταβατικά φαινόμενα ή πολύ υψηλές συχνότητες θα παρουσιάζουν ακουστά τεχνητά φαινόμενα. Ωστόσο, αυτές οι περιπτώσεις δεν είναι ο κανόνας και το ADPCM εξακολουθεί να χρησιμοποιείται συχνά.

Vorbis

Πρόκειται για έναν κωδικοποιητή ψυχοακουστικής γενικής χρήσης, που σημαίνει ότι χρησιμοποιεί τις ιδιαιτερότητες του ανθρώπινου αυτιού και εγκεφάλου για να συμπίεσει τον ήχο, ελαχιστοποιώντας ταυτόχρονα τα τεχνητά στοιχεία. Αυτός ο κωδικοποιητής εισήχθη αρχικά στον κόσμο των παιχνιδιών λόγω της καλής ποιότητας ήχου και της καλής συμπίεσης αρχείων που προσφέρει. Ο λόγος συμπίεσης είναι πολύ μεταβλητός και εξαρτάται από το ηχητικό σήμα. Μπορεί να κυμαίνεται από 2:1 (υψηλότερη ποιότητα ήχου, σε σήμα ευρείας ζώνης) έως 40:1 (χαμηλότερη ποιότητα, σε σήμα στενής ζώνης). Η εφαρμογή κωδικοποιητή που χρησιμοποιεί το Wwise έχει βελτιστοποιηθεί επανειλημμένα και το μέσο κόστος CPU είναι τώρα περίπου 1,5 έως 3 φορές υψηλότερο από αυτό του κωδικοποιητή ADPCM.

Το Vorbis είναι ένας κωδικοποιητής μεταβλητού ρυθμού bit, που σημαίνει ότι ο λόγος συμπίεσης εξαρτάται από το περιεχόμενο του αρχείου. Γενικά, όσο πιο θορυβώδης είναι ο ήχος, τόσο λιγότερη συμπίεση μπορεί να επιτευχθεί. Επίσης, το κόστος CPU συνδέεται άμεσα με τον παράγοντα ποιότητας που χρησιμοποιείται: περισσότερη ποιότητα σημαίνει περισσότερη χρήση CPU για την αποκωδικοποίηση.

Αυτή η μορφή απαιτεί έναν πίνακα αναζήτησης για να μπορεί να εκτελεί λειτουργίες αναζήτησης, να υποστηρίζει την επιλογή «Από το χρόνο που έχει παρέλθει» της εικονικής φωνής και το σύνολο χαρακτηριστικών της διαδραστικής μουσικής.

Επίσης, υπάρχει ένα μικρό επιπλέον κόστος μεταδεδομένων σε αυτή τη μορφή. Επομένως, πολύ μικρά αρχεία θα επηρεαστούν από αυτό. Κατά κανόνα, ήχοι μικρότεροι από 50 ms μπορεί να είναι μεγαλύτεροι από το απαραίτητο.

Opus

Το Opus είναι η καλύτερη λύση σε ό,τι αφορά την ποιότητα σε σχέση με το μέγεθος του αρχείου. Είναι ο διάδοχος του Vorbis και επωφελείται από σημαντικές βελτιώσεις στον λόγο συμπίεσης.

Η αντιληπτή ποιότητα είναι ελαφρώς υψηλότερη από αυτή του Vorbis, σύμφωνα με υποκειμενικές δοκιμές ακροατών. Επομένως, μπορεί να επιτύχει καλύτερους λόγους από το Vorbis, διατηρώντας παράλληλα την ποιότητα. Ωστόσο, έχει υψηλό κόστος CPU, συνήθως 5 έως 10 φορές πιο αργό από το ADPCM.

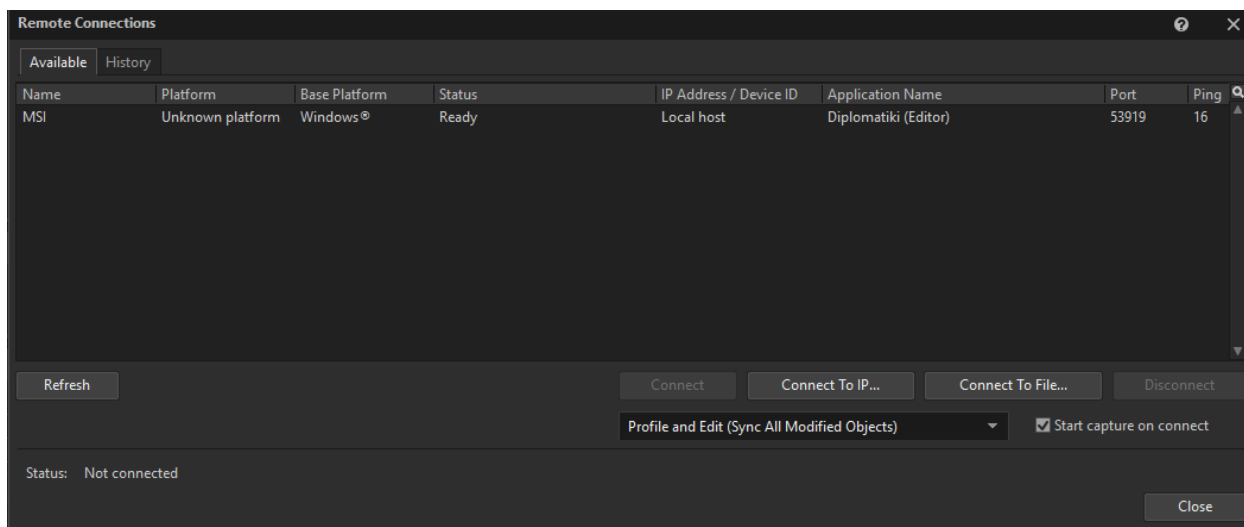
Ωστόσο δεν είναι ιδανικό για μικρά αρχεία. Ο αλγόριθμος χρειάζεται μερικά χιλιοστά του δευτερολέπτου δεδομένων για να ρυθμιστεί σωστά, επομένως προστίθενται περίπου 80 ms ήχου στο αρχείο. Επομένως, είναι σπατάλη να χρησιμοποιηθεί το Opus για μικρά κομμάτια ήχου περίπου αυτού του μήκους. Κατά κανόνα, αν ο ήχος είναι μικρότερος από 200 ms, απλά συνιστάται το Vorbis ή το ADPCM.

(Jean, 2023)

3.2.9 Ενσωμάτωση του AudioKinetic Wwise με το Unity

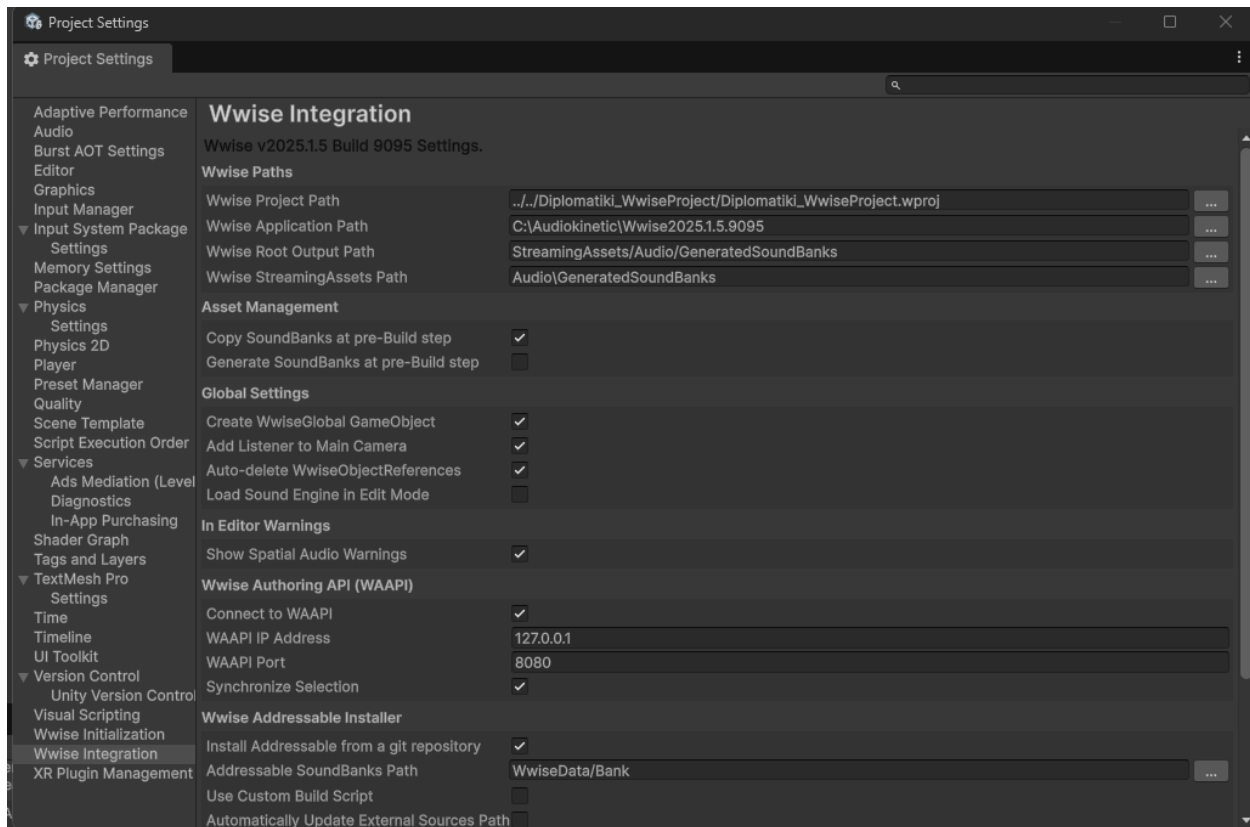
Όπως αναφέραμε παραπάνω, το Wwise παρέχει υποστήριξη σε διάφορες μηχανές παιχνιδιών εκ των οποίων το Unity. Η εγκατάσταση και η ενσωμάτωση του γίνεται μέσω του Launcher της AudioKinetic

Για να γίνει η επικοινωνία του, θα πρέπει το Unity να συνδεθεί μέσω Remote Platform στο Wwise για τον συγχρονισμό των Soundbanks.



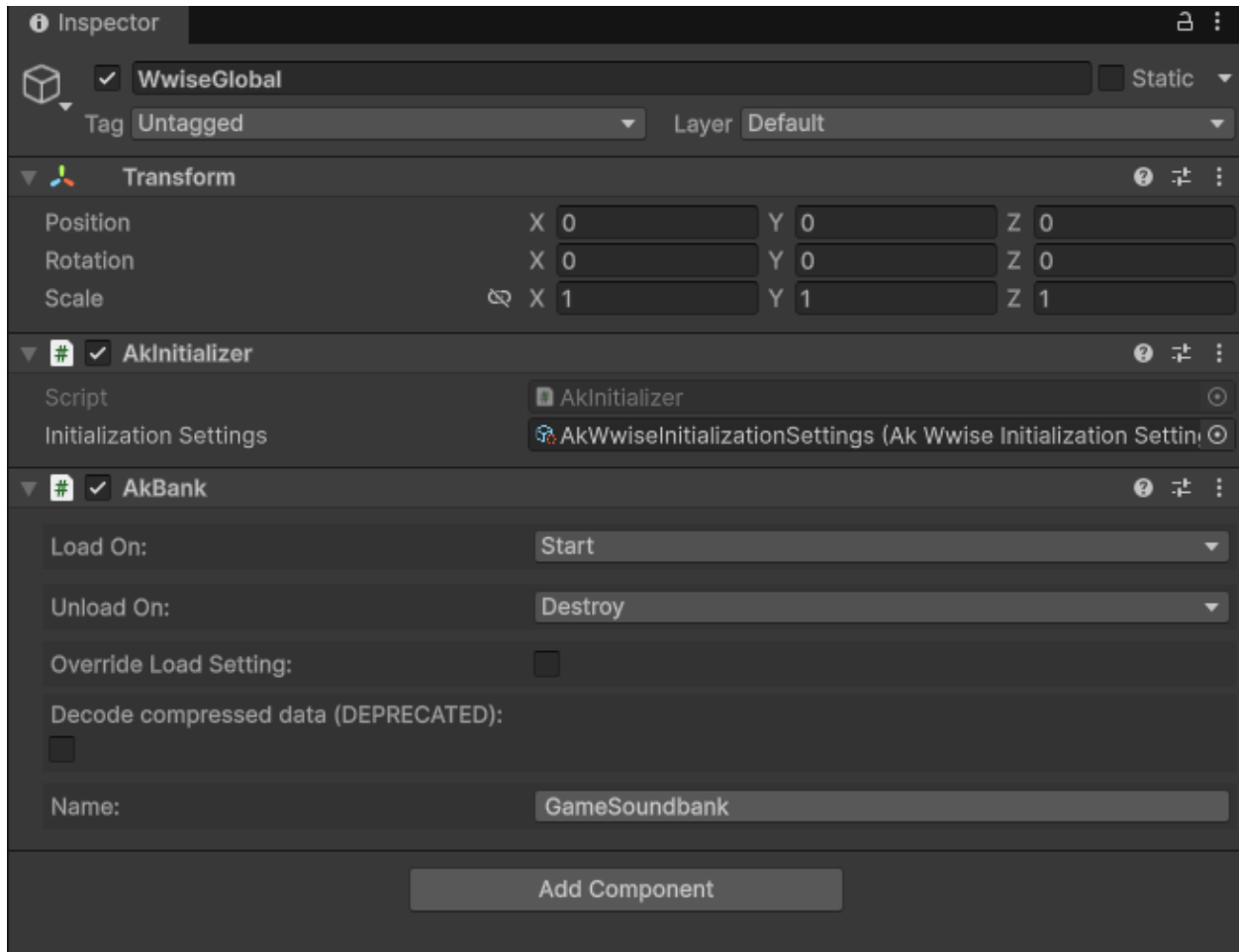
Εικ. 3-39 Παράθυρο Remote Connection του Wwise.

Στο παράθυρο Project Settings του Unity και συγκεκριμένα στο Wwise Integration, βλέπουμε τις διαδρομές αρχείων που είναι αποθηκευμένα τα αρχεία Soundbanks σε σχέση με το πρότζεκτ του Unity και την διεύθυνση IP και πύλη που συνδέεται με το Remote Connection του Wwise



Εικ. 3-40 Παράθυρο Project Settings Wwise Integration

Παρακάτω είναι ένας πίνακας από τα κύρια συστατικά (Components) που χρησιμοποιούνται στο Unity στα αντικείμενα παιχνιδιού (GameObjects). Μπορεί επίσης να ενσωματωθούν και στον πηγαίο κώδικα του παιχνιδιού. Το κύριο αντικείμενο παιχνιδιού που διαχειρίζεται αυτά είναι το WwiseGlobal.



Εικ. 3-41 Παράθυρο WwiseGlobal

Component	Περιγραφή
AkAmbient	Χρησιμοποιείται στους ατμοσφαιρικούς ήχους (Ambient). Μπορεί να χρησιμοποιηθούν σε πολλαπλά σημεία σε ένα αντικείμενο ή πολλαπλά αντικείμενα παιχνιδιού από ένα prefab με τον ίδιο ήχο για εξοικονόμηση στην CPU. Μπορεί να ρυθμιστεί η ακτίνα εξασθένησης του ήχου
AkBank	Φορτώνει και ξεφορτώνει μία τράπεζα ήχων (SoundBank) υπό συγκεκριμένες συνθήκες και γίνεται η κωδικοποίηση και αποκωδικοποίηση τους.
AkEnvironment	Χρησιμοποιείται στην διαμόρφωση χώρου αντήχησης μέσα σε ένα παιχνίδι
AkEvent	Προσθέτει γεγονότα ήχου σε αντικείμενα παιχνιδιού.
AkGameObj	Επικοινωνεί το αντικείμενο παιχνιδιού με τα αντικείμενα του Wwise.

AkInitializer	Ασχολείται με την αρχικοποίηση και τις αλλαγές στα καρέ της μηχανής ήχου Wwise. Παραμένει ενεργό για όλο τον κύκλο ζωής του παιχνιδιού και σε όλες τις σκηνές.
AkMicrophone	Επιτρέπει την αλληλεπίδραση μικροφώνου-υπολογιστή. Μπορεί να ρυθμιστεί ο ρυθμός δειγματολυσίας.
AkState	Καλεί τη συνάρτηση SetState της μηχανής ήχου αφού ενεργοποιηθεί ένα γεγονός πχ σύγκρουση με ένα άλλο αντικείμενο.
AkTrigger...	Χρησιμοποιείται ως εκκίνηση ενεργοποίησης του αντικείμενου ήχου στο παιχνίδι είτε με επαφή με άλλο αντικείμενο παιχνιδιού είναι από ενέργεια παίκτη.

Πίνακας 3-6 Πίνακας περιγραφής συστατικών του Wwise στο Unity.

3.3 Ανασκόπηση 3^{ου} κεφαλαίου

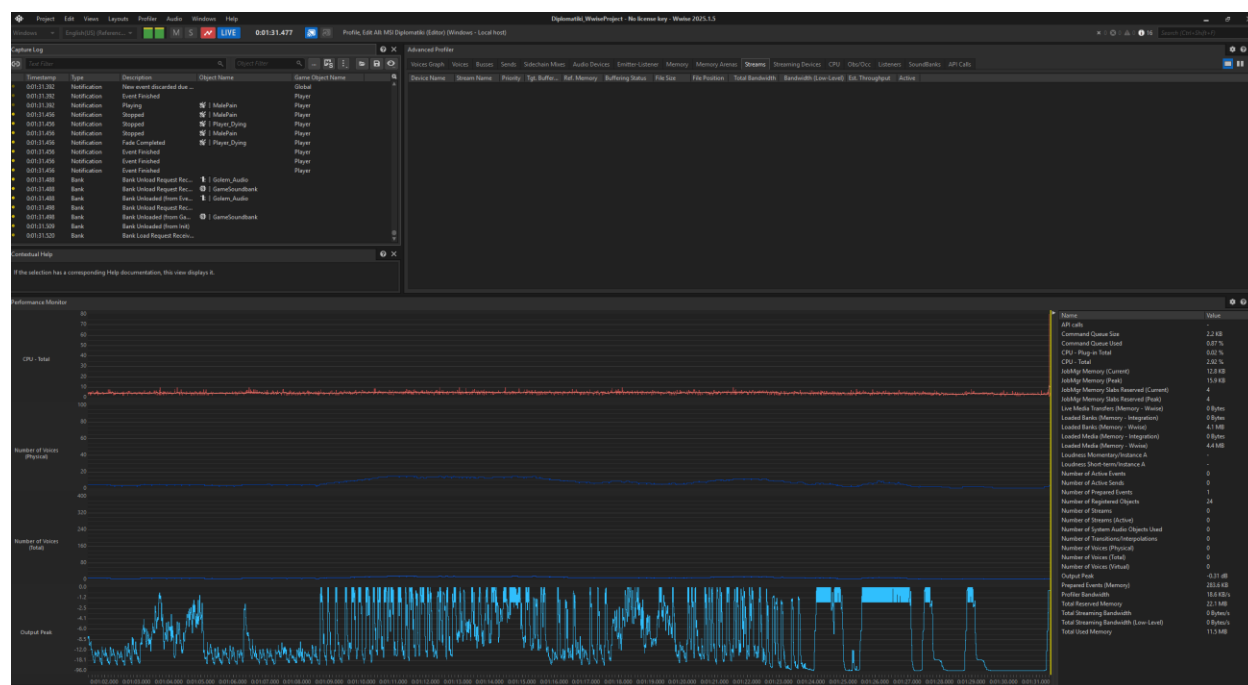
Σε αυτό το κεφάλαιο έγινε εκτενής ανάλυση των προγραμμάτων λογισμικού (Unity και Audiokinetic Wwise) που χρησιμοποιήθηκαν για την ανάπτυξη του παιχνιδιού με τα Zombie. Στη συνέχεια έγινε η ανάλυση του κώδικα και των επιμέρους στοιχείων των προγραμμάτων με έμφαση στην αλληλεπίδραση του χαρακτήρα με τους εχθρούς και τον ήχο. Τέλος έγινε αναφορά στην συμπίεση του ήχου και τον ρόλο της στις πλατφόρμες παιχνιδιών.

4 Αποτελέσματα

Η αξιολόγηση αποτελεί σημαντικό στάδιο για την λήψη πληροφοριών κατά την διάρκεια αλλά και μετά το πέρας της διαδικασίας. Δίνεται η δυνατότητα να αξιολογηθούν εάν συμβάλλει επιτυχώς το Audiokinetic Wwise και η ενσωμάτωση του με την μηχανή παιχνιδιού κατά την χωρική αντίληψη του ήχου στο παιχνίδι.

Ένα σημαντικό κομμάτι για την ανάπτυξη ενός παιχνιδιού αποτελεί η αξιολόγηση από δοκιμαστές. Βοηθάει στην βελτίωση του τελικού αποτελέσματος, τυχόν εντοπισμός λαθών που έχουν ξεφύγει κατά της διάρκεια ανάπτυξη της και διεξαγωγή συμπερασμάτων.

Η εξαγωγή των γραφημάτων ήχου (Voice graphs) και την επίδοση του υπολογιστή γίνεται μέσω του παραθύρου profiler στο Wwise και για κάθε αντικείμενο ήχου χωριστά και επίσης την σειρά των αντικειμένων ήχου που εκτελούνται κατά την διάρκεια του παιχνιδιού.



Εικ. 4-1 Παράθυρο profiler

4.1 Αξιολόγηση

Η αξιολόγηση του παιχνιδιού πραγματοποιήθηκε από τρία άτομα, τα οποία δοκίμασαν τη λειτουργία του φορώντας ακουστικά, ώστε να εξεταστεί η συμβολή του αμφιωτικού ήχου στην εμπειρία εμπύθισης και στη χωρική αντίληψη. Και οι τρεις συμμετέχοντες ανέφεραν ότι μπορούσαν να αντιληφθούν την κατεύθυνση από την οποία προέρχονταν οι βασικές ηχητικές πηγές, όπως οι εχθροί, οι πυροβολισμοί και οι συγκρούσεις, και ότι ο ήχος τούς βοήθησε να αντιδρούν στα γεγονότα του παιχνιδιού. Παράλληλα, αξιολόγησαν θετικά τη μεταβολή της έντασης ανάλογα με την απόσταση και τη συνοχή μεταξύ των ηχητικών συμβάντων και των

ενεργειών στην οθόνη. Ωστόσο, επισημάνθηκε ότι σε στιγμές ταυτόχρονης αναπαραγωγής πολλών ήχων η διάκριση ορισμένων πηγών γινόταν δυσκολότερη, γεγονός που υποδεικνύει την ανάγκη περαιτέρω εξισορρόπησης της μίξης και ρύθμισης των επιπέδων έντασης. Συνολικά, η δοκιμή έδειξε ότι η αξιοποίηση του αμφιωτικού ήχου ενίσχυσε την αίσθηση παρουσίας και υποστήριξε τον προσανατολισμό των παικτών μέσα στο εικονικό περιβάλλον. Παρότι τα αποτελέσματα είναι ενθαρρυντικά, το μικρό δείγμα των τριών ατόμων δεν επιτρέπει τη γενίκευσή τους και κρίνεται σκόπιμη η μελλοντική αξιολόγηση με περισσότερους συμμετέχοντες και τυποποιημένο ερωτηματολόγιο.

4.2 Προβλήματα

Κατά την διεξαγωγή της διπλωματικής αντιμετωπίσαμε διάφορα προβλήματα όπως για παράδειγμα την εύρεση κατάλληλων αρχείων ήχου για συγκεκριμένα σενάρια. Στο Unity αντιμετωπίσαμε δυσκολία εύρεσης των κατάλληλων Assets από το Unity Asset Store λόγω κόστους και τυχόν δυσλειτουργίες που προκύψανε σε σχέση με την ενσωμάτωση του Audiokinetic Wwise και μεγάλους χρόνους φόρτισης του project που μας επιβάρυνε και επιβράδυνε την διαδικασία.

5 Συμπεράσματα

Σκοπός αυτής της μεταπτυχιακής εργασίας ήταν να αναπτυχθεί ένα δοκιμαστικό παιχνίδι που δοκιμάζει τα αντανακλαστικά ήχου των παικτών. Για να επιτευχθεί ο σκοπός αυτός χρησιμοποιήσαμε το μοντέλο μεθοδολογίας ανάπτυξης Agile που χαρακτηρίζεται από την ταχύτητα και την εκτέλεση μικρότερων βημάτων για την επίτευξη του στόχου.

Τα αποτελέσματα της αξιολόγησης από τους τρεις συμμετέχοντες δείχνουν ότι η ενσωμάτωση του αμφιωτικού ήχου συνέβαλε ουσιαστικά στην επίτευξη του βασικού στόχου της εφαρμογής. Οι δοκιμαστές μπόρεσαν να εντοπίζουν την κατεύθυνση βασικών ηχητικών πηγών, όπως οι εχθροί, οι πυροβολισμοί και οι συγκρούσεις, ενώ η μεταβολή της έντασης σε συνάρτηση με την απόσταση υποστήριξε τον προσανατολισμό και την έγκαιρη αντίδρασή τους στα γεγονότα του παιχνιδιού. Η αντιστοίχιση των ηχητικών συμβάντων με τις ενέργειες στην οθόνη ενίσχυσε επίσης τη συνοχή της εμπειρίας και την αίσθηση παρουσίας στο εικονικό περιβάλλον. Παράλληλα, η δυσκολότερη διάκριση επιμέρους πηγών όταν αναπαράγονταν πολλοί ήχοι ταυτόχρονα αναδεικνύει την ανάγκη περαιτέρω εξισορρόπησης της μίξης, ρύθμισης των επιπέδων έντασης και καλύτερης ιεράρχησης των σημαντικών ηχητικών συμβάντων. Συνεπώς, η χρήση του Wwise σε συνδυασμό με το Unity κρίνεται λειτουργική και κατάλληλη για την ανάπτυξη διαδραστικού χωρικού ήχου, χωρίς όμως τα ευρήματα να μπορούν να γενικευθούν λόγω του περιορισμένου δείγματος. Μια μελλοντική αξιολόγηση με περισσότερους συμμετέχοντες, τυποποιημένο ερωτηματολόγιο και συγκριτική δοκιμή με και χωρίς αμφιωτική επεξεργασία θα μπορούσε να προσφέρει ασφαλέστερα και ποσοτικά τεκμηριωμένα συμπεράσματα.

Προσωπικά έμεινα ικανοποιημένος με το αποτέλεσμα και την διαδικασία της ανάπτυξης του παιχνιδιού και με τα νεότερα εργαλεία, μειώνεται ο χρόνος και αυξάνεται η παραγωγικότητα κατά την ανάπτυξη παιχνιδιού. Η ενσωμάτωση του χωρικού ήχου στα παιχνίδια αποτελεί ένα από τις πιο εντυπωσιακές λειτουργίες στα βιντεοπαιχνίδια που μπορεί να αποτελέσει.

5.1 Μελλοντικά πλάνα

Η εργασία αυτή μπορεί να αποτελέσει βοήθημα και έμπνευση για τους προγραμματιστές παιχνιδιών που θέλουν να εμπεδώσουν στο κομμάτι του διαδραστικού ήχου, ενός ανερχόμενου πεδίου του κλάδου της ακουστικής που στο μέλλον θα συναντηθεί σε ολοένα και περισσότερα παιχνίδια και διαδραστικές εφαρμογές.

Στη σημερινή εποχή, η ανάπτυξη των παιχνιδιών έχει γίνει πιο προσιτή από ποτέ με μικρότερες ομάδες παιχνιδιών να χρησιμοποιούν προχωρημένα εργαλεία για την δημιουργία ολοένα και πιο σύγχρονων παιχνιδιών με χαμηλό κεφάλαιο.

Στο μέλλον, αναμένονται να εμφανιστούν νέες τεχνολογίες που θα κάνει την ψηφιακή διασκέδαση πιο συναρπαστική ειδικά στην εικονική πραγματικότητα και η τεχνητή νοημοσύνη

θα μπορεί να σχεδιάσει διαφορετικούς ήχους σώζοντας χρόνο από τον προγραμματιστή και από τον συνθέτη μειώνοντας τον χρόνο ανάπτυξης του παιχνιδιού.

Το αρχείο του παιχνιδιού βρίσκεται στην εξής διεύθυνση:

https://drive.google.com/file/d/1Xx3hxcM971VM7Dpo_o08vx_9ii-oS16E/view?usp=sharing

6 Βιβλιογραφικές αναφορές

Ακολουθούν οι βιβλιογραφικές πηγές της εργασίας μας

Arm. (2021). *Glossary - Gaming Engines*. Ανάκτηση από Arm:
<https://www.arm.com/glossary/gaming-engines>

Audience Analysis. (χ.χ.). Ανάκτηση από University of Pittsburgh - Department of Communication : <https://www.comm.pitt.edu/oral-comm-lab/audience-analysis>

Audio Codec. (2025). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Audio_codec

Audiokinetic inc. (2025). *Wwise Help*. Ανάκτηση από Audiokinetic Wwise:
https://www.audiokinetic.com/en/public-library/2025.1.8_9170

Binaural recording. (2025, Δεκέμβριος 30). Ανάκτηση από Wikipedia:
https://en.wikipedia.org/wiki/Binaural_recording

Cale Plut, P. P. (2020). Generative music in video games: State of the art, challenges, and prospects. *Entertainment Computing, Volume 33*.

David Murphy, F. N. (2010, Ιανουάριος). Spatial Sound for Computer Games and Virtual Reality.

Dolby. (2025). *What is surround sound?* Ανάκτηση από Dolby:
<https://www.dolby.com/experience/home-entertainment/articles/what-is-surround-sound-and-how-it-works>

GameMaker. (2023, Μαΐου 10). *The Seven Stages Of Game Development*. Ανάκτηση από GameMaker: <https://gamemaker.io/en/blog/stages-of-game-development>

Gibbons, S. (2017, Νοέμβριος 5). *UX Mapping Methods Compared: A Cheat Sheet* . Ανάκτηση από Nielsen Norman Group: <https://www.nngroup.com/articles/ux-mapping-cheat-sheet/>

Introduction to Binaural Audio – The Science Behind Immersive Sound . (2025, Μάιος 20). Ανάκτηση από Brandenburg Labs: <https://brandenburg-labs.com/introduction-to-binaural-audio-the-science-behind-immersive-sound/>

- Jean, M. (2023, Οκτώμβριος 5). *A Guide for Choosing the Right Codec*. Ανάκτηση από Audiokinetic: <https://www.audiokinetic.com/en/blog/a-guide-for-choosing-the-right-codec/>
- Møller, H. (1992). Fundamentals of Binaural Technology. *Applied Acoustics, Volume 36, Issues 3–4*, σσ. 171-218.
- O'Dell, C. (2023, Αύγουστος). *From the National Recording Registry: "Super Mario" (1985)*. Ανάκτηση από Library of Congress Blogs: <https://blogs.loc.gov/now-see-hear/2023/08/from-the-national-recording-registry-super-mario-1985/>
- Unity. (2026). *Unity 6.3 User Manual*. Ανάκτηση από Unity: <https://docs.unity3d.com/6000.3/Documentation/Manual/index.html>
- YM2612: *The chip that powered music on the Mega Drive*. (2021). Ανάκτηση από Yamaha Australia: https://au.yamaha.com/en/news_events/2021/1203_YM2612.html
- Λάζαρος, Μ. (2023, Σεπτέμβριος). Τεχνικές σοβαρών παιχνιδιών στη συλλογή και συναισθηματική επισημείωση δειγμάτων ομιλίας . Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο.
- Χατζηαντωνίου, Π. (χ.χ.). Διαφάνειες ΑΣΠ61. Πάτρα: ΕΑΠ.

Παράρτημα Α: Αρχεία Κώδικα του παιχνιδιού

CountdownTimer.cs

```
using UnityEngine;
using TMPro;
using UnityEngine.InputSystem;

public class CountdownTimer : MonoBehaviour
{
    public float timeRemaining;
    public bool timerIsRunning = false;
    public TMP_Text timeText;
    public PlayerHealth playerHealth;
    public PlayerInput playerInput;

    private void Start()
    {
        timeRemaining = Random.Range(60f, 300f);
        timerIsRunning = true;
        playerInput = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerInput>();
        playerHealth = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerHealth>();
    }

    void Update()
    {
        if (timerIsRunning)
        {
            if (timeRemaining > 0)
            {
                timeRemaining -= Time.deltaTime;
                DisplayTime(timeRemaining);
            }
            else
            {
                playerHealth.Die();
            }
        }
    }

    void DisplayTime(float timeToDisplay)
    {
        timeToDisplay += 1;
        float minutes = Mathf.FloorToInt(timeToDisplay / 60);
        float seconds = Mathf.FloorToInt(timeToDisplay % 60);
        timeText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }
}
```

```
public void AddTime(float amountToAdd)
{
    timeRemaining += amountToAdd;
    Debug.Log("Added " + amountToAdd + " seconds! New total: " + timeRemaining);
}

}
```

BulletBehavior.cs

```
using UnityEngine;
```

```
public class BulletBehavior : MonoBehaviour
{
    [Header("Wwise Settings")]
    public AK.Wwise.Event fireSound;
    public AK.Wwise.Event explodeSound;
    public AK.Wwise.RTPC azimuthRTPC;
    public AK.Wwise.Event impactSound;

    public GameObject explosionParticlePrefab;
    public GameObject player;
    void Start()
    {
        player = GameObject.FindWithTag("Player");
    }

    void Update()
    {
        float distanceToPlayer = Vector3.Distance(player.transform.position, transform.position);
        azimuthRTPC.SetValue(gameObject, Mathf.Atan2(player.transform.position.y - transform.position.y,
distanceToPlayer));
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("Building") || other.gameObject.CompareTag("Tree"))
        {
            Vector3 hitPos = other.ClosestPoint(transform.position);
            PlayImpactAudio(hitPos);
            Destroy(gameObject);
        }
        else if (other.gameObject.CompareTag("Barrel"))
        {
            explodeSound.Post(gameObject);
            GameObject particleInstance = Instantiate(explosionParticlePrefab, gameObject.transform.position,
```

```
gameObject.transform.rotation);
    Destroy(particleInstance, 3.0f);
    Destroy(gameObject);
    Destroy(other.gameObject);
}
}

void PlayImpactAudio(Vector3 position)
{
    GameObject tempAudioObj = new GameObject("TempImpactAudio");
    tempAudioObj.transform.position = position;

    AkGameObj akObj = tempAudioObj.AddComponent<AkGameObj>();

    impactSound.Post(tempAudioObj, (uint)AkCallbackType.AK_EndOfEvent, (cookie, type, info) => {
        Destroy(tempAudioObj);
    });
}
}
```

EnemyBehavior.cs

```
using System;
using UnityEngine;
using UnityEngine.AI;
using System.Collections;
using UnityEngine.Serialization;

public class EnemyBehavior : MonoBehaviour
{
    public NavMeshAgent agent;
    public GameObject player;
    public Animator enemyAnimator;
    public bool isDead;
    public int health = 25;
    private CapsuleCollider capsule;
    public GameObject gameManager;
    public GameObject[] Pickups;
    public int points = 20;

    [Header("Wwise Settings")]
    public AK.Wwise.Event enemySound;
    public AK.Wwise.RTPC distanceRTPC;
    public AK.Wwise.RTPC azimuthRTPC;

    void Start()
    {
        isDead = false;
        GetComponent<CapsuleCollider>().enabled = true;
        gameManager = GameObject.Find("GameManager");
        agent = GetComponent<NavMeshAgent>();
        player = GameObject.FindGameObjectWithTag("Player");
        enemyAnimator = GetComponent<Animator>();
        AkBankManager.LoadBank("GameSoundbank", false, false);
    }

    void Update()
    {
        if (isDead) return;
        if (player == null) return;
        float distanceToPlayer = Vector3.Distance(player.transform.position, transform.position);
        enemyAnimator.SetFloat("playerDistance", distanceToPlayer);
        distanceRTPC.SetValue(gameObject, distanceToPlayer);
        azimuthRTPC.SetValue(gameObject, Mathf.Atan2(player.transform.position.y - transform.position.y,
distanceToPlayer));
        if (agent != null && agent.isActiveAndEnabled && agent.isOnNavMesh)
        {
            agent.SetDestination(player.transform.position);
        }
    }
}
```

```

    }
}

private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Bullet"))
    {
        enemySound.Post(gameObject);
        TakeDamage(5);
        Destroy(other.gameObject);
    }
}

public void TakeDamage(int amount)
{
    health -= amount;
    Debug.Log("Enemy hit! Health remaining: " + health);

    if (health <= 0)
    {
        EnemyDying();
    }
}

void EnemyDying()
{
    gameManager.GetComponent<GameManager>().AddPoints(points);
    gameManager.GetComponent<GameManager>().IncreaseKills();
    isDead = true;
    enemyAnimator.SetTrigger("OnDying");
    agent.isStopped = true;
    GetComponent<CapsuleCollider>().enabled = false;
    float prob = UnityEngine.Random.Range(0f, 1f);
    if (prob > 0.6f)
    {
        GameObject healthPickup = Instantiate(Pickups[0], new Vector3(transform.position.x, 2,
transform.position.z), Quaternion.identity);
        AkUnitySoundEngine.RegisterGameObj(healthPickup);
    }
    else if (prob > 0.8f)
    {
        GameObject timePickup = Instantiate(Pickups[1], new Vector3(transform.position.x, 2,
transform.position.z), Quaternion.identity);
        AkUnitySoundEngine.RegisterGameObj(timePickup);
    }
    AkGameObj wwiseObj = GetComponent<AkGameObj>();
    if (wwiseObj != null)
    {
        wwiseObj.enabled = false;
    }
}

```

```
}  
    Destroy(gameObject, 3.0f);  
}  
  
/*private void OnCollisionEnter(Collision other)  
{  
    if (other.gameObject.tag == "Player")  
    {  
  
    }  
}*/  
}
```

HealthBar.cs

```
public class HealthBar : MonoBehaviour
{

    public Slider slider;
    public Gradient gradient;
    public Image fill;

    public void SetMaxHealth(int health)
    {
        slider.maxValue = health;
        slider.value = health;

        fill.color = gradient.Evaluate(1f);
    }

    public void SetHealth(int health)
    {
        slider.value = health;

        fill.color = gradient.Evaluate(slider.normalizedValue);
    }
}
```

PlayerMovement.cs

```
using UnityEngine;  
using UnityEngine.InputSystem;  
using UnityEngine.Serialization;
```

```
public class PlayerMovement : MonoBehaviour
```

```
{
```

```
    public PlayerInput playerInput;  
    public CharacterController characterController;  
    public Animator characterAnimator;  
    public float speed = 5f;  
    private Vector3 playerMovement;
```

```
    [Header("Mouse positioning")]  
    private Vector2 mousePosition;  
    private bool _isFiring;
```

```
    public Transform bulletSpawnPoint;  
    public GameObject bulletPrefab;
```

```
    [Header("Shooting Settings")]  
    public float bulletSpeed = 20f;  
    private float timeBetweenShots = 0.2f;  
    public float fireDelay = 1f;
```

```
    [Header("Wwise settings")]  
    public AK.Wwise.Event footstepEvent;
```

```
    public PlayerHealth playerHealth;  
    public CountdownTimer countdownTimer;  
    private float lastFireTime;
```

```
    void Start()
```

```
    {  
        playerInput = GetComponent<PlayerInput>();  
        characterController = GetComponent<CharacterController>();  
        characterAnimator = GetComponent<Animator>();  
        AkBankManager.LoadBank("GameSoundbank", false, false);  
    }
```

```
    void Update()
```

```
    {  
        Move();  
        RotateToMouse();  
        UpdateAnimation();  
        /*if (_isFiring && Time.time >= lastFireTime + timeBetweenShots)  
        {
```

```
        Shoot();  
        lastFireTime = Time.time;  
    }*/  
}  
  
void Move()  
{  
    Vector3 playerMove = new Vector3(playerMovement.x, 0, playerMovement.z);  
    characterController.Move(playerMove * speed * Time.deltaTime);  
  
}  
  
void RotateToMouse()  
{  
    Ray ray = Camera.main.ScreenPointToRay(mousePosition);  
    Plane groundPlane = new Plane(Vector3.up, transform.position);  
  
    if (groundPlane.Raycast(ray, out float distance))  
    {  
        Vector3 worldPosition = ray.GetPoint(distance);  
  
        Vector3 direction = worldPosition - transform.position;  
        direction.y = 0;  
  
        if (direction.sqrMagnitude > 0.001f)  
        {  
            Quaternion targetRotation = Quaternion.LookRotation(direction);  
            transform.rotation = targetRotation;  
        }  
    }  
}  
  
void UpdateAnimation()  
{  
    bool isMoving = playerMovement.sqrMagnitude > 0.01f;  
    characterAnimator.SetBool("isRunning", isMoving);  
    characterAnimator.SetBool("isShooting", _isFiring);  
}  
  
private void OnMove(InputValue value)  
{  
    Vector3 input = value.Get<Vector3>();  
    playerMovement = new Vector3(input.x, 0, input.z);  
}  
  
private void OnLook(InputValue value)  
{  
    mousePosition = value.Get<Vector2>();
```

```
}

private void OnClick(InputValue value)
{
    _isFiring = value.isPressed;
}

private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.CompareTag("Pickup"))
    {
        Debug.Log("Player has gained health");
        Destroy(other.gameObject);
        playerHealth.GainHealth(10);
    }
    else if (other.gameObject.CompareTag("TimePickup"))
    {
        Debug.Log("Player has gained remaining time");
        Destroy(other.gameObject);
        countdownTimer.AddTime(10);
    }
}

void Shoot()
{
    GameObject bullet = Instantiate(bulletPrefab, bulletSpawnPoint.position, transform.rotation);
    Rigidbody rb = bullet.GetComponent<Rigidbody>();
    rb.AddForce(bulletSpawnPoint.forward * bulletSpeed, ForceMode.Impulse);
    Destroy(bullet, 3f);
}

void PlayFootstepSound()
{
    footstepEvent.Post(gameObject);
}

}
```

Instantiate.cs

```
using UnityEngine;  
using System.Collections;
```

```
public class Instantiate : MonoBehaviour  
{  
    public GameObject[] enemyPrefabs;  
    public GameObject player;  
  
    void Start()  
    {  
        if (player == null)  
        {  
            player = GameObject.FindGameObjectWithTag("Player");  
        }  
        InvokeRepeating("EnemySpawn", 1.0f, 2.0f);  
    }  
  
    void Update()  
    {  
        if (player != null && player.GetComponent<PlayerHealth>().isDead)  
        {  
            CancelInvoke("EnemySpawn");  
        }  
    }  
  
    void EnemySpawn()  
    {  
        Vector3 randomSpawnPosition = new Vector3(Random.Range(-25, 15), 0, Random.Range(-30, 30));  
        GameObject randomEnemy = enemyPrefabs[Random.Range(0, enemyPrefabs.Length)];  
        Instantiate(randomEnemy, randomSpawnPosition, Quaternion.identity);  
        AkUnitySoundEngine.RegisterGameObj(randomEnemy);  
    }  
}
```

PlayerHealth.cs

```
using UnityEngine;
using UnityEngine.InputSystem;
using UnityEngine.Serialization;

public class PlayerHealth : MonoBehaviour
{
    public int maxHealth = 100;
    public int currentHealth;
    public HealthBar healthBar;
    public GameObject gameOverCanvas;

    public bool isDead;
    public PlayerInput playerInput;
    public CapsuleCollider playerCollider;
    public GameManager gameManager;

    [Header("Wwise Settings")]

    public AK.Wwise.Event pickupSound;
    public AK.Wwise.Event hurtSound;
    public AK.Wwise.Event deathEvent;
    public AK.Wwise.State deathState;
    void Start()
    {
        playerInput = GetComponent<PlayerInput>();
        playerCollider = GetComponent<CapsuleCollider>();
        playerCollider.enabled = true;
        isDead = false;
        deathState.SetValue();
        currentHealth = maxHealth;
        healthBar.SetMaxHealth(maxHealth);
        AkUnitySoundEngine.RegisterGameObj(gameObject);
    }

    private void OnCollisionEnter(Collision other)
    {
        if(other.gameObject.CompareTag("Enemy"))
        {
            Debug.Log("Player hit enemy");
            TakeDamage(20);
        }
    }

    void TakeDamage(int damage)
    {

```

```
currentHealth -= damage;

healthBar.SetHealth(currentHealth);
hurtSound.Post(gameObject);
if (currentHealth <= 0)
{
    Die();
}
}

public void GainHealth(int Damage)
{
    currentHealth += Damage;
    healthBar.SetHealth(currentHealth);
    pickupSound.Post(gameObject);
}

public void Die()
{
    GameObject enemies = GameObject.FindWithTag("Enemy");
    GameObject spawner = GameObject.FindWithTag("Instantiator");
    isDead = true;

    if (playerInput != null)
    {
        playerInput.enabled = false;
    }

    PlayDeathSound();
    GetComponent<Animator>().SetBool("IsDying", true);
    gameOverCanvas.SetActive(true);
    Cursor.lockState = CursorLockMode.None;
    Cursor.visible = true;
    Destroy(enemies);
    Destroy(spawner);
    gameManager.SetHighScore();
}

void PlayDeathSound()
{
    deathEvent.Post(gameObject);
}
}
```

Explodable.cs

using UnityEngine;

```
public class Explodable : MonoBehaviour
{
    public float explosionRadius = 5.0f;
    public int explosionDamage = 50;
    public LayerMask enemyLayer;

    [Header("Effects & Audio")]
    public GameObject explosionParticlePrefab;
    public AK.Wwise.Event explosionSound;
    public void Explode()
    {
        if (explosionParticlePrefab != null)
        {
            GameObject particleInstance = Instantiate(explosionParticlePrefab, transform.position, transform.rotation);
            Destroy(particleInstance, 3.0f);
        }

        if (explosionSound != null)
        {
            explosionSound.Post(gameObject);
        }

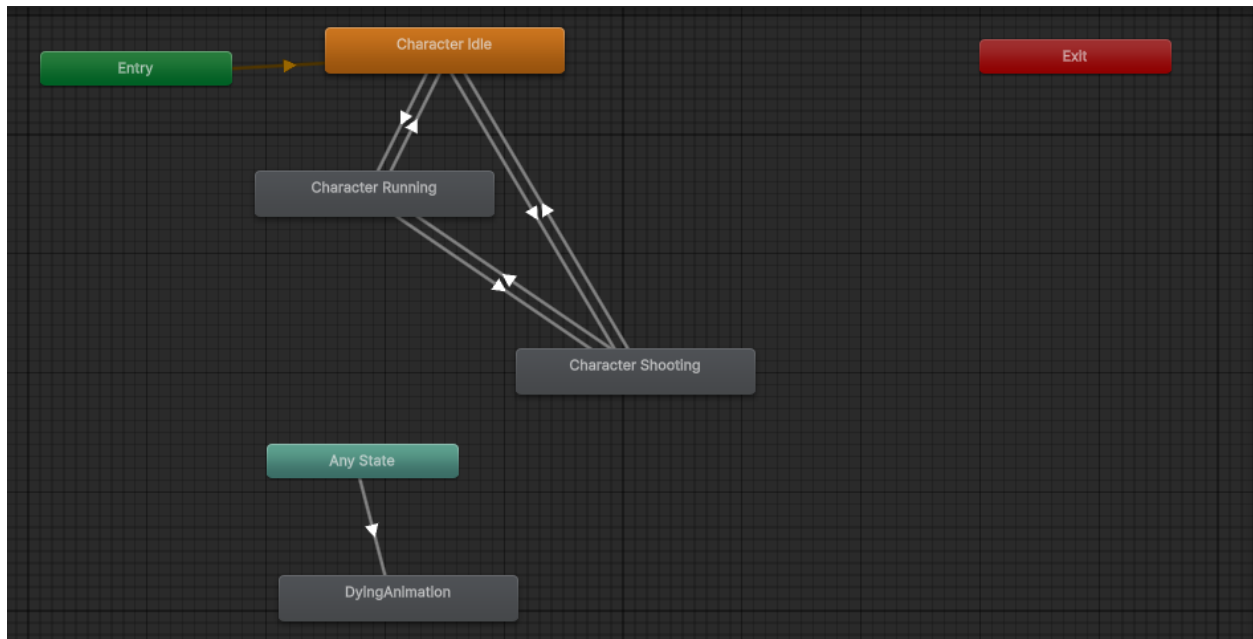
        Collider[] collidersToDamage = Physics.OverlapSphere(transform.position, explosionRadius, enemyLayer);

        foreach (Collider nearbyObject in collidersToDamage)
        {
            EnemyBehavior enemy = nearbyObject.GetComponent<EnemyBehavior>();

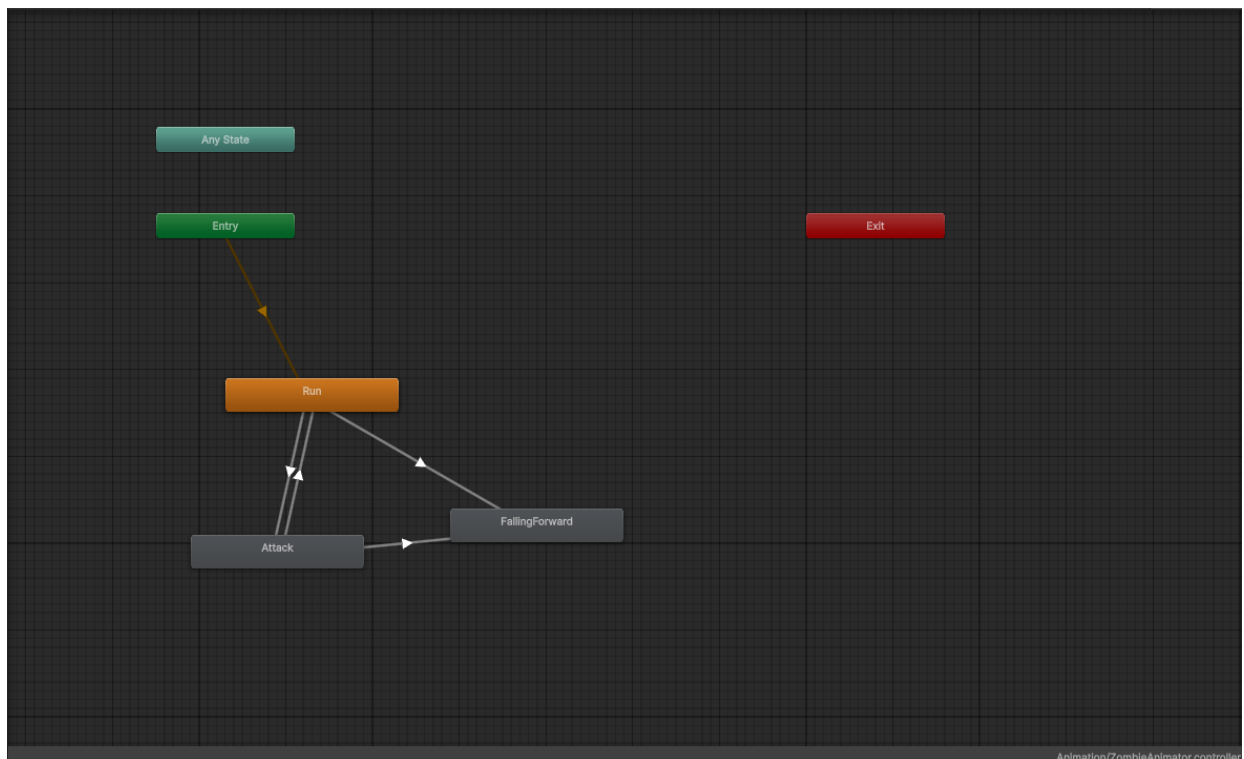
            if (enemy != null)
            {
                enemy.TakeDamage(explosionDamage);
            }
        }

        Destroy(gameObject);
    }
}
```

Παράρτημα Β: Διαγράμματα παιχνιδιού



Εικ. 0-1 Διάγραμμα Animator παίκτη



Εικ. 0-2 Διάγραμμα animation εχθρού