7 Νοεμβρίου 2024

5η Διάλεξη
Συστήματα Μηχανής Παιχνιδιών - Ο Βρόχος Παιχνιδιού

ΔΙΔΑΣΚΩΝ: ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΙΧΛΑΣ

ΕΠΙΜΕΛΕΙΑ ΣΗΜΕΙΩΣΕΩΝ: ΠΟΤΟΣ ΒΑΣΙΛΕΙΟΣ, ΚΑΡΥΩΤΗ ΒΑΡΒΑΡΑ

Περιεχόμενα

Συστήματα Μηχανής Παιχνιδιών
Ο Βρόχος Παιχνιδιού

Εισαγωγή

Για την ανάπτυξη βιντεοπαιχνιδιών είναι εξαιρετικά σημαντικό να κατανοήσουμε την αρχιτεκτονική και τη λειτουργία μιας μηχανής παιχνιδιών, αναλύοντας βασικά στοιχεία και συστήματα που απαιτούνται για την ανάπτυξή της. Ξεκινώντας από την αρχιτεκτονική, θα παρουσιαστεί αναλυτικά η δομή της μηχανής, η οποία περιλαμβάνει εργαλεία και συστήματα εκτέλεσης, από το επίπεδο του υλικού έως εφαρμογές υψηλού επιπέδου. Επίσης, αντίστοιχη βαρύτητα πρέπει να δοθεί σε ορισμένα συστήματα που συμβάλουν στην βελτίωση της απόδοσης και της σταθερότητας, όπως τα χαμηλού επιπέδου SDKs, οι διαχειριστές πόρων, η φυσική, οι συγκρούσεις, και η απόδοση γραφικών και ήχου.

Η λειτουργία του παιχνιδιού ελέγχεται μέσω του κεντρικού βρόχου (Game Loop), ο οποίος διαχειρίζεται τον χρόνο, την εισαγωγή χρηστών και τη ροή των γραφικών σε πραγματικό χρόνο, χρησιμοποιώντας τα απαραίτητα εργαλεία, τα οποία θα δούμε αναλυτικότερα στη συνέχεια. Θα κατανοήσουμε καλύτερα όσα αναφέρθηκαν μέσα από παραδείγματα βρόχων, όπως το Pong και το Asteroids. Τέλος, σημαντική είναι η αναφορά μας στην διαχείριση του χρόνου στις μηχανές παιχνιδιών, καθώς έχουμε έλεγχο, όχι μόνο του χρόνου του παιχνιδιού, αλλά και του πραγματικού χρόνου και του τοπικού χρόνου, για συγχρονισμό και σταθερότητα. Σημαντική είναι η συμβολή και η χρήση της έννοιας του deltaTime, με απευθείας ανάγνωση του χρόνου ή με χρήση τρέχοντα μέσου όρου για να ρυθμίσουμε τα fps (frames per second).

Μηχανή Παιχνιδιών

Ορισμός

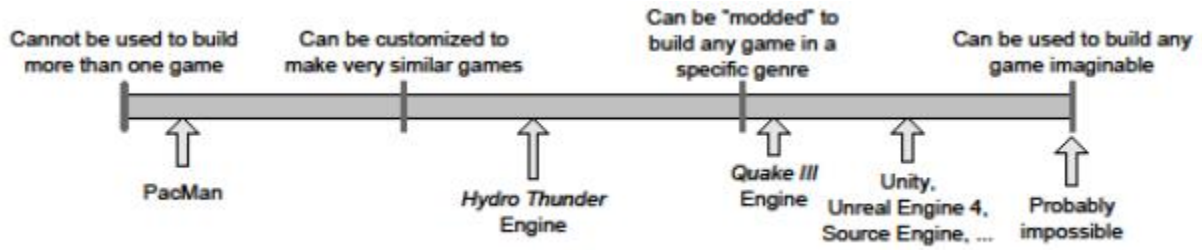
Οι σύγχρονες μηχανές παιχνιδιών είναι κατά βάση καθοδηγούμενες από τα δεδομένα με βασικό στόχο την επαναχρησιμοποιησιμότητα, επιτρέποντας έτσι την κατασκευή νέων παιχνιδιών χωρίς να χρειάζεται η ανάπτυξη εκ νέου βασικών συστημάτων.

Βασικά στοιχεία και Παιχνίδια

Ο όρος "μηχανή παιχνιδιών" εμφανίστηκε τη δεκαετία του 1990, με το Doom να είναι το πρώτο παιχνίδι που καθιέρωσε σαφή διαχωρισμό ανάμεσα τόσο στα βασικά και στα καλλιτεχνικά στοιχεία του παιχνιδιού, όσο και στους κανόνες του. Αναλυτικότερα, στο Doom, τα βασικά στοιχεία της μηχανής παιχνιδιών περιλαμβάνουν το σύστημα γραφικών, το σύστημα ανίχνευσης σύγκρουσης και το σύστημα ήχου, δηλαδή τους τεχνικούς μηχανισμούς που εξασφαλίζουν την ομαλή λειτουργία του παιχνιδιού. Τα καλλιτεχνικά στοιχεία αφορούν τα μοντέλα, τις υφές και την κίνηση, δηλαδή το οπτικό και ακουστικό περιεχόμενο που διαμορφώνει τον κόσμο του παιχνιδιού και την αισθητική του. Τέλος, οι κανόνες του παιχνιδιού αποτελούν τους μηχανισμούς που καθορίζουν τη ροή και τη λογική του.

Στη συνέχεια, ορισμένα παιχνίδια ακολούθησαν τις ίδιες αρχές, όπως τα Quake III και Unreal, και οι εταιρείες άρχισαν να πωλούν άδειες χρήσης των μηχανών και εργαλείων τους, επιτρέποντας την χρήση τους από το κοινό που είχε τις απαραίτητες άδειες.

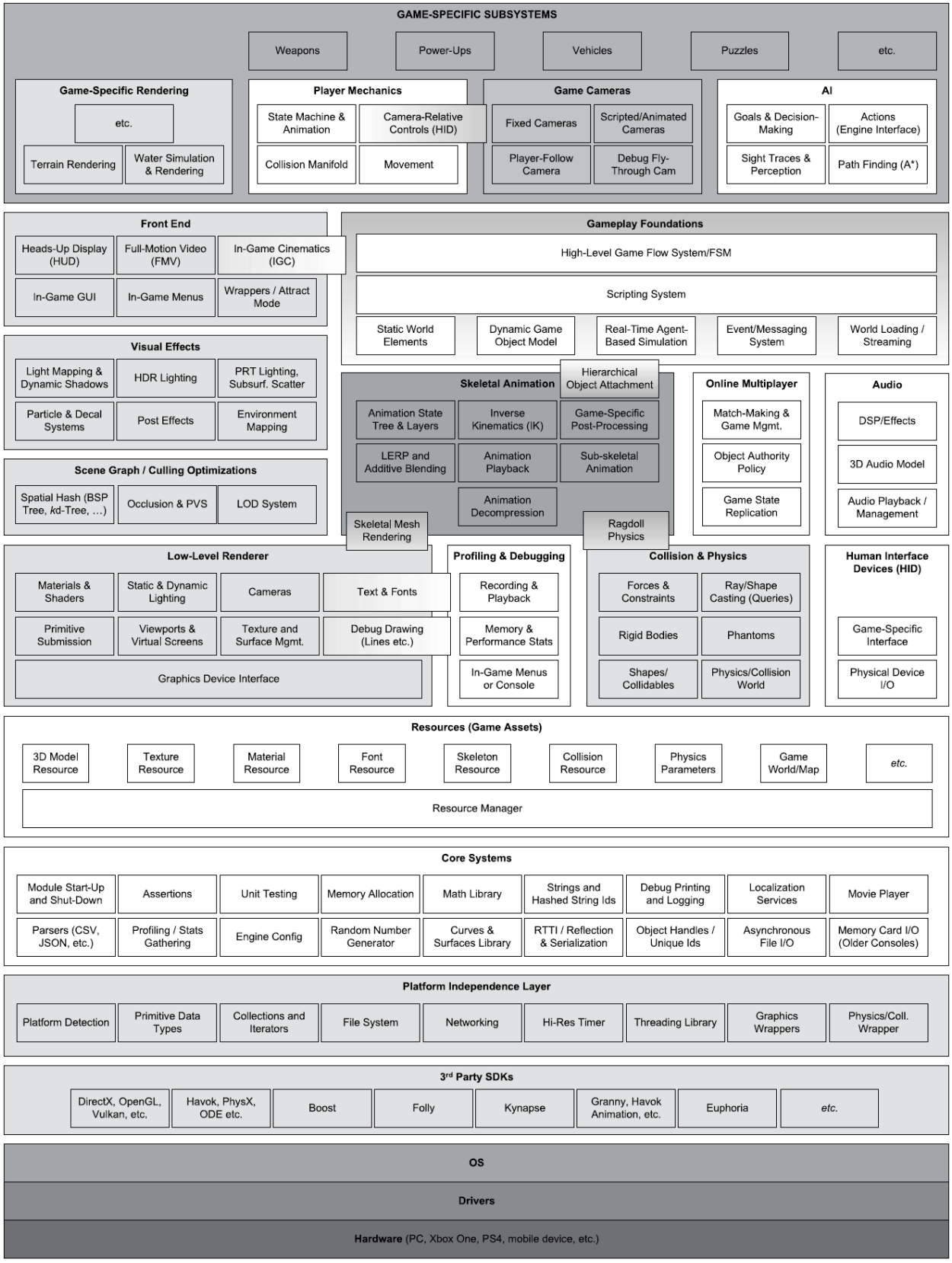
Ευελιξία μηχανών



Ουσιαστικά, ανάλογα την μηχανή, μπορεί αυτή είτε να χρησιμοποιηθεί για μεγάλο εύρος παιχνιδιών είτε να είναι σχεδιασμένη αποκλειστικά για κάποια ή και για ένα συγκεκριμένο είδος. Όπως βλέπουμε στην παραπάνω εικόνα, οι μηχανές ορισμένων πολύ γνωστών παιχνιδιών στο ευρύ κοινό, μπορούν να επαναχρησιμοποιηθούν, άλλες ευκολότερα και άλλες δυσκολότερα. Στην αριστερή πλευρά, η μηχανή του PacMan είναι σχεδιασμένη αποκλειστικά για το συγκεκριμένο παιχνίδι και δεν επαναχρησιμοποιείται, ενώ η μηχανή του Hydro Thunder μπορεί να προσαρμοστεί για πολύ παρόμοια παιχνίδια. Στο κέντρο του φάσματος, η μηχανή του Quake III προσφέρει δυνατότητα τροποποίησης για την ανάπτυξη παιχνιδιών συγκεκριμένου είδους. Προς τη δεξιά πλευρά, μηχανές όπως το Unity, το Unreal Engine 4 και το Source Engine είναι εξαιρετικά ευέλικτες και μπορούν να χρησιμοποιηθούν για τη δημιουργία παιχνιδιών κάθε είδους. Στην άκρα δεξιά, ένας υποθετικός τύπος μηχανής θα ήταν απόλυτα γενικός και θα μπορούσε να υποστηρίξει αποδοτικά κάθε πιθανό είδος παιχνιδιού, κάτι που θεωρείται σχεδόν αδύνατο να επιτευχθεί (ως προς την αποδοτικότητα).

Αρχιτεκτονική

Σημαντική είναι η αναφορά της αρχιτεκτονικής μιας μηχανής παιχνιδιών, η οποία περιλαμβάνει μια σουίτα εργαλείων και συστημάτων εκτέλεσης που εκτείνονται από το επίπεδο υλικού μέχρι τις υψηλού επιπέδου εφαρμογές. Η αρχιτεκτονική σχεδιάζεται σε επίπεδα για να διευκολύνει την επαναχρησιμοποίηση και τη δοκιμή, αποφεύγοντας κυκλικές εξαρτήσεις που μπορεί να δημιουργούν προβλήματα. Ο πολυεπίπεδος σχεδιασμός συμβάλλει στη διασφάλιση ότι τα διάφορα συστήματα μπορούν να λειτουργούν ανεξάρτητα, επιτρέποντας παράλληλη ανάπτυξη και εύκολη τροποποίηση των επιμέρους τμημάτων της μηχανής. Τα επιμέρους επίπεδα μπορούμε να τα δούμε και στην παρακάτω εικόνα.


GAME-SPECIFIC SUBSYSTEMS
Weapons
Power-Ups
Vehicles
Puzzles
etc.
Game-Specific Rendering
etc.
Terrain Rendering
Water Simulation & Rendering
Player Mechanics
State Machine & Animation
Camera-Relative Controls (HID)
Collision Manifold
Movement
Game Cameras
Fixed Cameras
Scripted/Animated Cameras
Player-Follow Camera
Debug Fly-Through Cam
AI
Goals & Decision-Making
Actions (Engine Interface)
Sight Traces & Perception
Path Finding (A*)
Front End
Heads-Up Display (HUD)
Full-Motion Video (FMV)
In-Game Cinematics (IGC)
In-Game GUI
In-Game Menus
Wrappers / Attract Mode
Gameplay Foundations
High-Level Game Flow System/FSM
Scripting System
Static World Elements
Dynamic Game Object Model
Real-Time Agent-Based Simulation
Event/Messaging System
World Loading / Streaming
Visual Effects
Light Mapping & Dynamic Shadows
HDR Lighting
PRT Lighting, Subsurf. Scatter
Particle & Decal Systems
Post Effects
Environment Mapping
Skeletal Animation
Hierarchical Object Attachment
Animation State Tree & Layers
Inverse Kinematics (IK)
Game-Specific Post-Processing
LERP and Additive Blending
Animation Playback
Sub-skeletal Animation
Animation Decompression
Skeletal Mesh Rendering
Ragdoll Physics
Online Multiplayer
Match-Making & Game Mgmt.
Object Authority Policy
Game State Replication
Audio
DSP/Effects
3D Audio Model
Audio Playback / Management
Scene Graph / Culling Optimizations
Spatial Hash (BSP Tree, kd-Tree, …)
Occlusion & PVS
LOD System
Low-Level Renderer
Materials & Shaders
Static & Dynamic Lighting
Cameras
Text & Fonts
Primitive Submission
Viewports & Virtual Screens
Texture and Surface Mgmt.
Debug Drawing (Lines etc.)
Graphics Device Interface
Profiling & Debugging
Recording & Playback
Memory & Performance Stats
In-Game Menus or Console
Collision & Physics
Forces & Constraints
Ray/Shape Casting (Queries)
Rigid Bodies
Phantoms
Shapes/ Collidables
Physics/Collision World
Human Interface Devices (HID)
Game-Specific Interface
Physical Device I/O
Resources (Game Assets)
3D Model Resource
Texture Resource
Material Resource
Font Resource
Skeleton Resource
Collision Resource
Physics Parameters
Game World/Map
etc.
Resource Manager
Core Systems
Module Start-Up and Shut-Down
Assertions
Unit Testing
Memory Allocation
Math Library
Strings and Hashed String Ids
Debug Printing and Logging
Localization Services
Movie Player
Parsers (CSV, JSON, etc.)
Profiling / Stats Gathering
Engine Config
Random Number Generator
Curves & Surfaces Library
RTTI / Reflection & Serialization
Object Handles / Unique Ids
Asynchronous File I/O
Memory Card I/O (Older Consoles)
Platform Independence Layer
Platform Detection
Primitive Data Types
Collections and Iterators
File System
Networking
Hi-Res Timer
Threading Library
Graphics Wrappers
Physics/Coll. Wrapper
3rd Party SDKs
DirectX, OpenGL, Vulkan, etc.
Havok, PhysX, ODE etc.
Boost
Folly
Kynapse
Granny, Havok Animation, etc.
Euphoria
etc.
OS
Drivers
Hardware (PC, Xbox One, PS4, mobile device, etc.)

Αυτό το σχήμα δείχνει ξεκάθαρη τη πολυεπίπεδη αρχιτεκτονική μιας μηχανής παιχνιδιών, από το χαμηλότερο επίπεδο υλικού μέχρι τα εργαλεία και τα συστήματα που υποστηρίζουν την ανάπτυξη και εκτέλεση παιχνιδιών. Η αρχιτεκτονική αυτή εξασφαλίζει ότι οι επιμέρους λειτουργίες της μηχανής παιχνιδιών είναι διακριτές, επιτρέποντας τη συντήρηση, την αναβάθμιση, και την ανεξάρτητη ανάπτυξη. Ας δούμε συνοπτικά το κάθε επίπεδο:

1. **Hardware**: Στο κάτω μέρος, το επίπεδο υλικού περιλαμβάνει τις υπολογιστικές πλατφόρμες όπως προσωπικούς υπολογιστές, κονσόλες βιντεοπαιχνιδιών, κινητά, κ.ο.κ..
2. **SDKs τρίτων:** Περιλαμβάνει διάφορα εργαλεία που αφορούν τις συγκεκριμένες πλατφόρμες όπως βιβλιοθήκες δομών δεδομένων και αλγορίθμων (π.χ., STL), βιβλιοθήκες γραφικών (π.χ., OpenGL) και άλλα.
3. **Platform Independence Layer**: Αυτό το επίπεδο παρέχει πρόσβαση σε βασικές υπηρεσίες όπως είσοδο/έξοδο, επεξεργασία, και διαχείριση αρχείων και ανεξαρτητοποιεί τα παραπάνω επίπεδα από την αρχιτεκτονική της υπολογιστικής πλατφόρμας.
4. **Core Systems**: Αυτά τα συστήματα περιλαμβάνουν βασικές λειτουργίες όπως η διαχείριση μνήμης, οι βιβλιοθήκες μαθηματικών, ο έλεγχος μονάδων, και η υποστήριξη για debugging.
5. **Resources Manager**: Το επίπεδο αυτό διαχειρίζεται τα στοιχεία του παιχνιδιού (assets) όπως αρχεία ήχου, μοντέλα, γραφικά, και άλλα δεδομένα που απαιτούνται για τη λειτουργία του παιχνιδιού. Ο διαχειριστής πόρων είναι κρίσιμος για την αποδοτική φόρτωση και εκφόρτωση δεδομένων κατά τη διάρκεια του παιχνιδιού.
6. **Gameplay Foundation**: Το επίπεδο αυτό αποτελεί τη βάση για τα συστήματα παιχνιδιού και περιλαμβάνει διαχείριση αρχείων, τη διαχείριση της ροής παιχνιδιού, και βασικές δομές δεδομένων.
7. **Gameplay Systems**: Περιέχει βασικά συστήματα για τον έλεγχο του gameplay, όπως φυσική, γραφικά, ήχο, και AI.
8. **Game-Specific Layer**: Στο ανώτερο επίπεδο, υπάρχουν τα συστήματα και τα εργαλεία που σχετίζονται με συγκεκριμένα χαρακτηριστικά του παιχνιδιού. Αυτά μπορεί να περιλαμβάνουν rendering για μοναδικά γραφικά, συγκεκριμένες επιλογές AI, και ρυθμίσεις gameplay που προσαρμόζονται στις ανάγκες του εκάστοτε παιχνιδιού.

Στη συνέχεια θα δούμε αναλυτικότερα τα συστήματα μηχανής τα οποία απεικονίζονται στην παραπάνω εικόνα.

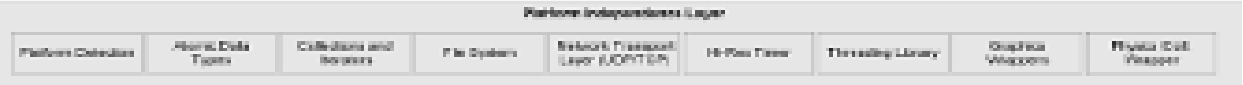
Συστήματα χαμηλού επιπέδου

Τα συστήματα χαμηλού επιπέδου περιλαμβάνουν το υλικό, το οποίο αποτελεί το σύστημα στο οποίο το παιχνίδι τρέχει. Οι οδηγοί συσκευών (Device Drivers) διαχωρίζουν το λειτουργικό σύστημα και τα ανώτερα συστήματα από τις χαμηλού επιπέδου λεπτομέρειες επικοινωνίας με τις συσκευές. Το λειτουργικό σύστημα διαχειρίζεται την εκτέλεση πολλαπλών προγραμμάτων σε μία μηχανή και στις κονσόλες είναι εξαιρετικά λιτό.

SDKs(Software Development Kit) τρίτων

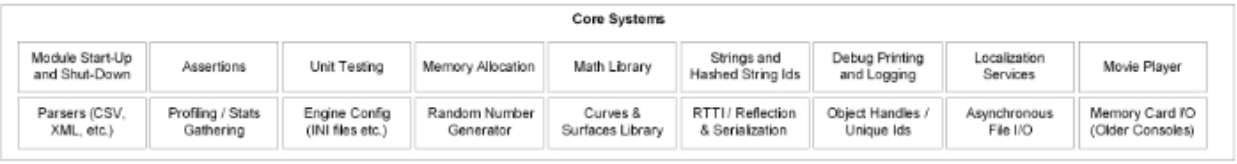
Τα SDKs τρίτων περιλαμβάνουν διάφορες βιβλιοθήκες και εργαλεία που χρησιμοποιούνται για την ανάπτυξη λογισμικού. Οι δομές δεδομένων και οι αλγόριθμοι είναι θεμελιώδη για την αποδοτική επεξεργασία δεδομένων. Η STL (Standard Template Library) της C++ παρέχει έτοιμες υλοποιήσεις για κοινές δομές δεδομένων και αλγόριθμους, ενώ η βιβλιοθήκη Boost επεκτείνει τις δυνατότητες της C++ με πρόσθετες λειτουργίες. Στα γραφικά, εργαλεία όπως το OpenGL και το DirectX παρέχουν τη δυνατότητα δημιουργίας 3D γραφικών. Για συγκρούσεις και φυσική, εργαλεία όπως τα Havok, PhysX, ODE και Bullet προσφέρουν λύσεις προσομοίωσης φυσικών φαινομένων. Στην κίνηση χαρακτήρων, το Granny 3D χρησιμοποιείται για τον ορθό χειρισμό των κινούμενων χαρακτήρων. Τέλος, η τεχνητή νοημοσύνη υποστηρίζεται από εργαλεία όπως το Unity ML-Agents Toolkit και το Unreal's AI Toolkit για τη δημιουργία εξελιγμένων AI συστημάτων.

Επίπεδο ανεξαρτησίας πλατφόρμας



Αυτό το επίπεδο αποτελεί τον πυρήνα της προσαρμοστικότητας της μηχανής παιχνιδιών σε διαφορετικές πλατφόρμες, εξασφαλίζοντας ότι οι βασικές λειτουργίες του παιχνιδιού, όπως η διαχείριση μνήμης, ο χρονομετρητής και η είσοδος/έξοδος δεδομένων, είναι ανεξάρτητες από το υλικό και το λειτουργικό σύστημα. Παρέχει μία ενιαία διεπαφή που καλύπτει όλες τις βασικές ανάγκες του συστήματος και καθιστά τη μηχανή παιχνιδιών συμβατή με διαφορετικά περιβάλλοντα χωρίς να απαιτούνται αλλαγές σε χαμηλού επιπέδου κώδικα. Επίσης, περιλαμβάνει λειτουργίες όπως όπως θεμελιώδης τύπους, υποστήριξη δικτύου, συστήματα αρχείων και άλλα.

Συστήματα Πυρήνα



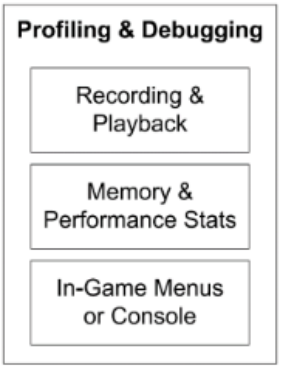
Ο πυρήνας ενός συστήματος είναι υπεύθυνος για την αποτελεσματική διαχείριση πόρων και τη διασύνδεση του υλικού με το λογισμικό. Ο πυρήνας διαχειρίζεται τη μνήμη για την αποθήκευση δεδομένων και εντολών, χρησιμοποιεί βιβλιοθήκες μαθηματικών για υπολογισμούς με διανύσματα και μητρώα (π.χ., φυσική κίνηση, σύγκρουση), και παρέχει προσαρμοσμένες δομές δεδομένων για την αποδοτική αποθήκευση και επεξεργασία πληροφοριών παιχνιδιού. Έτσι, εξασφαλίζεται ομαλή λειτουργία και αλληλεπίδραση μεταξύ των στοιχείων του παιχνιδιού, προσφέροντας μια συνεχή εμπειρία για τον παίκτη.

Διαχειριστής πόρων (resources manager)

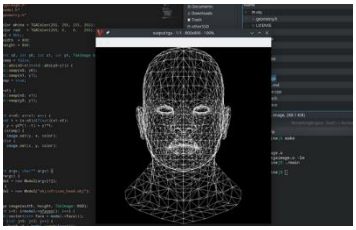
Ο διαχειριστής πόρων σε ένα παιχνίδι παρέχει μια ενιαία διεπαφή για την πρόσβαση στα στοιχεία του παιχνιδιού (assets), όπως 3D μοντέλα, υφές, ήχους και γραμματοσειρές. Το επίπεδο πολυπλοκότητας της διαχείρισης εξαρτάται από τις απαιτήσεις του παιχνιδιού: οι προγραμματιστές συχνά χρειάζεται να φορτώνουν άμεσα πόρους, ενώ σε άλλες περιπτώσεις, μηχανές παιχνιδιών διαχειρίζονται πλήρως την αποθήκευση και την απόδοση των στοιχείων, μειώνοντας την πολυπλοκότητα για τους προγραμματιστές.

Profiling/Εντοπισμός σφαλμάτων

Το profiling και ο εντοπισμός σφαλμάτων σε ένα παιχνίδι περιλαμβάνουν τη χρονομέτρηση του κώδικα για ανάλυση απόδοσης, την εμφάνιση στατιστικών στην οθόνη για παρακολούθηση σε πραγματικό χρόνο, και την αποθήκευση δεδομένων απόδοσης για μελλοντική ανάλυση. Η διαδικασία αυτή βοηθά στη βελτιστοποίηση της χρήσης μνήμης και την εξάλειψη μη αποδοτικών τμημάτων του κώδικα. Επιπλέον, επιτρέπει την καταγραφή και αναπαραγωγή συγκεκριμένων γεγονότων για ενδελεχή ανάλυση, διευκολύνοντας την ανίχνευση και επιδιόρθωση σφαλμάτων.

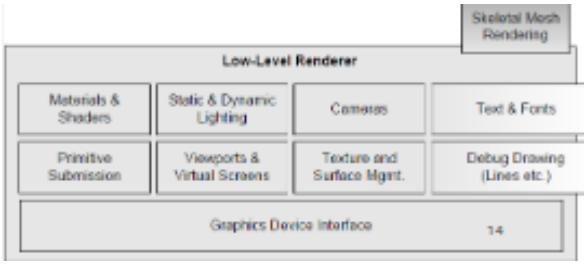


Μηχανή Απόδοσης Γραφικών



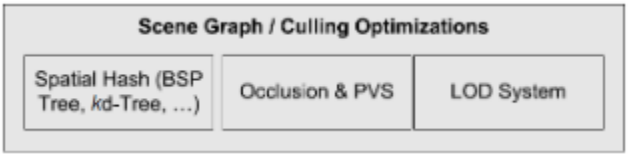
Η μηχανή απόδοσης γραφικών σε ένα παιχνίδι είναι υπεύθυνη για τη χαμηλού επιπέδου απόδοση των γραφικών, δηλαδή τη μετάφραση των εντολών του προγράμματος σε εικόνες που προβάλλονται στην οθόνη. Περιλαμβάνει τη διαχείριση του γραφήματος σκηνής, όπου οργανώνονται τα αντικείμενα του παιχνιδιού, και υποστηρίζει την εφαρμογή οπτικών εφέ, όπως φωτισμούς, σκιές και αντανακλάσεις, που βελτιώνουν την αισθητική του παιχνιδιού. Το front end της μηχανής επιτρέπει στους προγραμματιστές να αλληλεπιδρούν με τα στοιχεία της σκηνής, εξασφαλίζοντας ευκολία στη ρύθμιση και προσαρμογή της γραφικής απόδοσης.

Χαμηλού Επιπέδου Απόδοση Γραφικών (Κυρίως η GPU)

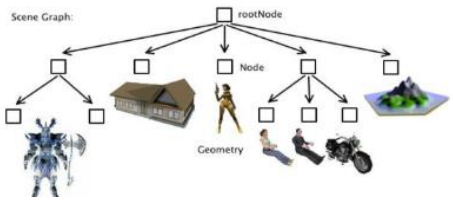


Η χαμηλού επιπέδου απόδοση γραφικών, η οποία υλοποιείται κυρίως από την GPU, επικεντρώνεται στη βέλτιστη απόδοση θεμελιωδών στοιχείων χωρίς να εξετάζει αν είναι ορατά στον χρήστη. Περιλαμβάνει τη διαχείριση της διεπαφής με τις συσκευές γραφικών, την απαρίθμηση και την αρχικοποίησή τους, καθώς και τη ρύθμιση της προσωρινής μνήμης για αποδοτικότερη λειτουργία. Επιπλέον, υποστηρίζει την αναπαράσταση γεωμετρικών αντικειμένων, τη διεπαφή της κάμερας, το σύστημα υλικών (materials), στατικό και δυναμικό φωτισμό, καθώς και την απόδοση κειμένου και γραμματοσειρών, προσφέροντας έτσι τα βασικά εργαλεία για την απεικόνιση των γραφικών του παιχνιδιού.

Γράφημα Σκηνής

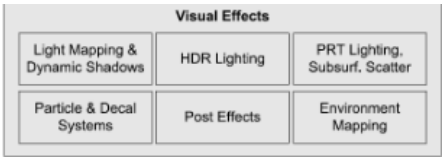


Το γράφημα σκηνής είναι μια δομή δεδομένων που οργανώνει και βελτιστοποιεί τα αντικείμενα σε μια σκηνή παιχνιδιού, μειώνοντας τον αριθμό των θεμελιωδών πράξεων που πρέπει να εκτελεστούν για την απόδοση. Χρησιμοποιεί τεχνικές όπως το frustum culling για να εξαιρεί αντικείμενα που βρίσκονται εκτός του οπτικού πεδίου της κάμερας, βελτιώνοντας έτσι την απόδοση. Επιπλέον, εφαρμόζει χωρική υποδιαίρεση μέσω δομών όπως BSP trees (Binary Space Partitioning), quadtrees, octrees και kd-trees για την οργάνωση των αντικειμένων σε μικρότερες περιοχές, διευκολύνοντας την αποτελεσματική διαχείριση και απόδοση των γραφικών.



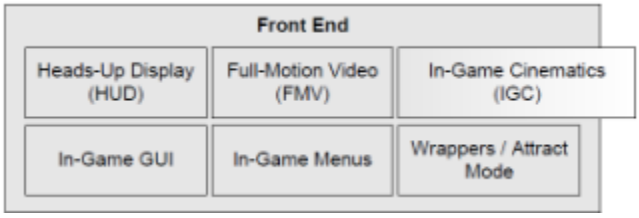
Για παράδειγμα, στο διπλανό σχήμα τα φύλλα του δέντρου περιέχουν τα πραγματικά δεδομένα της σκηνής, όπως γεωμετρία αντικειμένων, υλικά και άλλα χαρακτηριστικά, ενώ οι εσωτερικοί κόμβοι οργανώνουν και ομαδοποιούν αυτά τα δεδομένα, διευκολύνοντας τη διαχείριση και την απόδοσή τους. Η δομή αυτή επιτρέπει στο σύστημα να επεξεργάζεται αποδοτικά μεγάλες σκηνές, εφαρμόζοντας τεχνικές βελτιστοποίησης όπως το frustum culling και η χωρική υποδιαίρεση.

Οπτικά εφέ



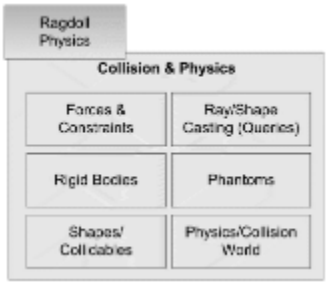
Τα οπτικά εφέ είναι βασικά στοιχεία για τη βελτίωση της οπτικής ποιότητας και της εμπειρίας σε ένα παιχνίδι. Τα συστήματα σωματιδίων (όπως για παράδειγμα η δημιουργία φωτιάς) χρησιμοποιούνται για να προσομοιώσουν φυσικά φαινόμενα και να προσθέσουν ρεαλιστικότητα. Τα decal συστήματα επιτρέπουν την προσθήκη επιφανειακών λεπτομερειών, όπως σημάδια ή γρατσουνιές, στα αντικείμενα του παιχνιδιού. Το light mapping χρησιμοποιείται για την αποθήκευση προ-υπολογισμένων πληροφοριών φωτισμού, βελτιώνοντας την απόδοση και τη ρεαλιστικότητα του φωτισμού σε στατικές σκηνές. Η δυναμική σκίαση επιτρέπει την παραγωγή ρεαλιστικών σκιών που προσαρμόζονται ανάλογα με τη θέση των αντικειμένων και του φωτός. Τέλος, τα μετα-εφέ πλήρους οθόνης εφαρμόζονται στο τελικό frame για επιπλέον εφέ, όπως θόλωση κίνησης ή αλλαγές στον κορεσμό των χρωμάτων, προσδίδοντας πιο κινηματογραφικό αποτέλεσμα.

Front end



Το Front End αναφέρεται στα οπτικά στοιχεία και στη διεπαφή του παιχνιδιού με τα οποία αλληλεπιδρά ο παίκτης. Περιλαμβάνει το HUD (Head-Up Display), το οποίο εμφανίζει κρίσιμες πληροφορίες όπως τη ζωή, τα πυρομαχικά ή την πρόοδο του παίκτη χωρίς να διακόπτει τη ροή του παιχνιδιού. Περιλαμβάνει επίσης το μενού, που δίνει στον παίκτη πρόσβαση στις βασικές ρυθμίσεις και επιλογές πλοήγησης, και το GUI για τη διαχείριση χαρακτήρα, μέσω του οποίου ο παίκτης μπορεί να προσαρμόζει και να παρακολουθεί τα χαρακτηριστικά του χαρακτήρα του. Τέλος, τα video για cut scenes προσφέρει αφηγηματικές σκηνές που εμβαθύνουν την ιστορία και εμπλουτίζουν την εμπειρία του παίκτη.

Συγκρούσεις και Φυσική



Η διαχείριση συγκρούσεων και η προσομοίωση φυσικής είναι κρίσιμα στοιχεία για τη ρεαλιστική αναπαράσταση της κίνησης και της αλληλεπίδρασης των αντικειμένων σε ένα παιχνίδι. Συνήθως, η φυσική αφορά τη δυναμική των άκαμπτων σωμάτων, δηλαδή αντικειμένων που δεν αλλάζουν σχήμα, όπως πτώσεις, αναπηδήσεις και συγκρούσεις. Η ανάπτυξη μιας μηχανής φυσικής είναι ένα πολύπλοκο και απαιτητικό έργο, για αυτό οι προγραμματιστές συχνά χρησιμοποιούν έτοιμες βιβλιοθήκες φυσικής για να διευκολύνουν τη διαδικασία. Μερικές από τις πιο γνωστές βιβλιοθήκες είναι το Havok, το PhysX της NVIDIA, το ODE (Open Dynamics Engine) και το Bullet, οι οποίες προσφέρουν εργαλεία και λειτουργίες για την προσομοίωση φυσικών φαινομένων και την αντιμετώπιση συγκρούσεων.

Κινούμενο Σχέδιο (Animation)

Το Κινούμενο Σχέδιο (Animation) αποτελεί ένα βασικό κομμάτι της ανάπτυξης παιχνιδιών, επιτρέποντας τη δημιουργία ρεαλιστικών και εκφραστικών κινήσεων για χαρακτήρες και αντικείμενα. Οι πιο συνηθισμένοι τύποι κινούμενου σχεδίου περιλαμβάνουν την κίνηση sprite/texture, όπου αλλάζουν τα sprites ή τα textures για να δημιουργηθεί η αίσθηση της κίνησης, και την κίνηση ιεραρχίας άκαμπτου σώματος (Rigid body hierarchy animation), όπου τα τμήματα ενός αντικειμένου κινούνται σύμφωνα με μια ιεραρχική δομή. Η σκελετική κίνηση (skeletal animation), η πιο δημοφιλής μέθοδος, χρησιμοποιεί έναν σκελετό για να κατευθύνει τις κινήσεις των χαρακτήρων. Άλλες τεχνικές περιλαμβάνουν την κίνηση κόμβων (vertex animation), όπου κινούνται συγκεκριμένοι κόμβοι του μοντέλου, και τη μορφοποίηση (morphing), που επιτρέπει τη σταδιακή αλλαγή της μορφής του αντικειμένου.

HID

Το HID (Human Interface Device) λειτουργεί ως διασύνδεση που μετατρέπει τα ακατέργαστα δεδομένα εισόδου από συσκευές σε χρήσιμη πληροφορία για το παιχνίδι. Οι βασικές συσκευές εισόδου περιλαμβάνουν το πληκτρολόγιο και το ποντίκι, τα joypads και πιο εξειδικευμένα χειριστήρια. Επιπλέον, περιλαμβάνονται συσκευές όπως το Kinect, που επιτρέπει την ανίχνευση κίνησης, και τα VR headsets, τα οποία προσφέρουν πολύ ιδιαίτερη εμπειρία εικονικής πραγματικότητας. Αυτές οι συσκευές εισόδου διευκολύνουν την

αλληλεπίδραση του παίκτη με το παιχνίδι, παρέχοντας ένα πιο φυσικό και ακριβές σύστημα ελέγχου.

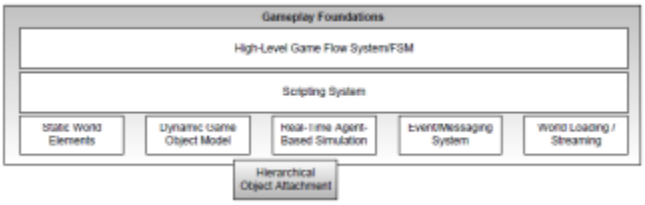
Ήχος

Ο ήχος στα παιχνίδια συχνά παραμελείται και αφήνεται για το τέλος, παρόλο που συμβάλλει καθοριστικά στη συνολική εμπειρία του παίκτη. Η πολυπλοκότητα του ήχου διαφέρει ανάλογα με τις ανάγκες του παιχνιδιού, από απλά ηχητικά εφέ μέχρι σύνθετες μουσικές επενδύσεις και περιβαλλοντικούς ήχους. Πολλά παιχνίδια βασίζονται σε έτοιμα εργαλεία για τη διαχείριση του ήχου, όπως το XACT και το Scream, που διευκολύνουν την ενσωμάτωση ηχητικών εφέ και μουσικής με ελάχιστη ανάγκη για πρόσθετο προγραμματισμό.

Δικτύωση/Πολλαπλοί παίκτες

Η δικτύωση για παιχνίδια πολλαπλών παικτών καλύπτουν διάφορους τρόπους με τους οποίους μπορούν να παίξουν μαζί πολλοί παίκτες. Σε ένα παιχνίδι με ενιαία οθόνη, όλοι οι παίκτες βλέπουν την ίδια προβολή στην ίδια οθόνη, ενώ στη διαχωρισμένη οθόνη, η οθόνη χωρίζεται για να προσφέρει διαφορετικές προβολές σε κάθε παίκτη. Τα δικτυωμένα παιχνίδια επιτρέπουν τη σύνδεση πολλών υπολογιστών, προσφέροντας multiplayer εμπειρία μέσω τοπικού δικτύου ή διαδικτύου. Σε μεγάλα διαδικτυακά παιχνίδια για πολλούς παίκτες, τα οποία συνήθως εκτελούνται σε κεντρικό διακομιστή, μπορεί να παρουσιαστούν ζητήματα όπως η καθυστέρηση (lag), η οποία μπορεί να αντιμετωπιστεί μέχρι ένα βαθμό με τεχνικές πρόβλεψης από τον client και με χρήση του πρωτοκόλλου UDP, το οποίο παρέχει ταχύτητα αλλά είναι λιγότερο αξιόπιστο. Επιπλέον, η δικτύωση απαιτεί προστασία από cheating, καθώς οι παίκτες μπορεί να εκμεταλλεύονται αδυναμίες για να αποκτήσουν πλεονέκτημα.

Θεμελιώδες Σύστημα Υποστήριξης Gameplay



Το Θεμελιώδες Σύστημα Υποστήριξης Gameplay περιλαμβάνει όλα τα απαραίτητα στοιχεία που καθιστούν ένα παιχνίδι διαδραστικό και ενδιαφέρον για τον παίκτη. Περιλαμβάνει τη φόρτωση κόσμων και τη δημιουργία μοντέλων αντικειμένων και στατικών στοιχείων του περιβάλλοντος, ενώ επιτρέπει την εκτέλεση προσομοιώσεων πρακτόρων σε πραγματικό χρόνο, προσθέτοντας βάθος και πολυπλοκότητα στη λειτουργία του παιχνιδιού.

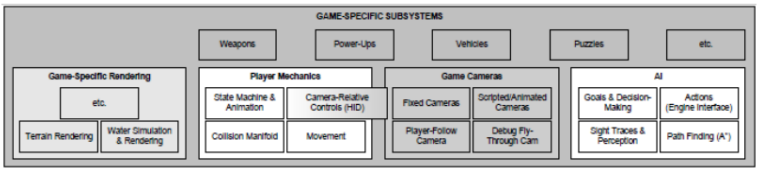
Σύστημα Γεγονότων & Scripting Σύστημα

Το σύστημα αυτό επιτρέπει την επικοινωνία μεταξύ των αντικειμένων μέσω ενός συστήματος γεγονότων (event system), όπου τα αντικείμενα στέλνουν μηνύματα προς έναν κεντρικό διαχειριστή συμβάντων. Παράλληλα, το scripting σύστημα προσφέρει τη δυνατότητα δημιουργίας νέας λογικής παιχνιδιού χωρίς την ανάγκη μεταγλώττισης, γεγονός που επιταχύνει την ανάπτυξη και τη δοκιμή του παιχνιδιού.

Βασικά Συστήματα AI

Τα Βασικά Συστήματα AI (Τεχνητής Νοημοσύνης) είναι υπεύθυνα για την αντίληψη και τη συμπεριφορά των εικονικών πρακτόρων του παιχνιδιού. Περιλαμβάνουν προσεγγίσεις και μεθόδους που αφορούν τον τρόπο αντίληψης του περιβάλλοντος, τον καθορισμό της μνήμης, την αντίδραση σε ερεθίσματα, καθώς και τον σχεδιασμό διαδρομής (path planning) με τη βοήθεια της δομής Navmesh για αποφυγή εμποδίων, προσφέροντας μια πιο ρεαλιστική αλληλεπίδραση.

Ειδικά υποσυστήματα που αφορούν το κάθε παιχνίδι ξεχωριστά



Τα Ειδικά Υποσυστήματα αφορούν χαρακτηριστικά που είναι μοναδικά για κάθε παιχνίδι και διαμορφώνονται ανάλογα με τις ανάγκες του. Αυτά τα συστήματα μπορεί να μην αποτελούν μέρος της βασικής μηχανής παιχνιδιού και συνήθως σχεδιάζονται για να υποστηρίξουν ιδιαίτερα χαρακτηριστικά του εκάστοτε παιχνιδιού.

Δημιουργία Ψηφιακού Περιεχομένου

Η Δημιουργία Ψηφιακού Περιεχομένου (Digital Content Creation – DCC) αποτελεί απαραίτητη διαδικασία, καθώς οι μηχανές παιχνιδιών χρησιμοποιούν δεδομένα σε πολλές μορφές, όπως 3D πλέγματα, υφές, ήχους και κινήσεις. Το περιεχόμενο αυτό συχνά δημιουργείται με εξωτερικά εργαλεία, όπως τα Maya, 3ds Max, Blender, Photoshop και SoundForge, και αποτελεί τη βάση για την ανάπτυξη του κόσμου του παιχνιδιού.

Διαχείριση Κόσμου Παιχνιδιού

Η διαχείριση και ο σχεδιασμός του κόσμου του παιχνιδιού ενσωματώνεται συνήθως στη μηχανή, επιτρέποντας στους σχεδιαστές να εργάζονται απευθείας με τα δεδομένα του παιχνιδιού. Αυτό το σύστημα διευκολύνει την οργάνωση, την τροποποίηση και τη διαχείριση του περιβάλλοντος, επιτρέποντας στους δημιουργούς να διαμορφώνουν τον κόσμο του παιχνιδιού σύμφωνα με τις ανάγκες του παίκτη.

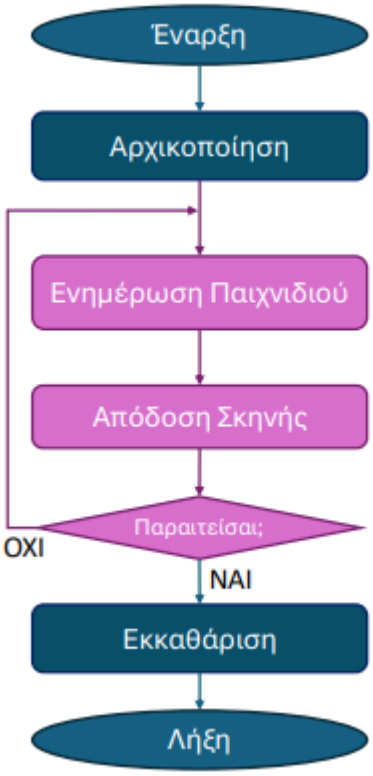
Βρόχος Παιχνιδιού

Εισαγωγή

Ο βρόχος του παιχνιδιού αφορά και περιγράφει όλες αυτές τις εργασίες που επαναλαμβάνονται για το σχεδιασμό κάθε frame. Βεβαίως, κάποιες εργασίες μπορεί να μην εκτελούνται σε κάθε επανάληψη (π.χ., AI) ενώ άλλες μπορεί να εκτελούνται περισσότερες από μία φορές σε κάθε επανάληψη (π.χ., προσομοίωση φυσικής).

Αρχικοποίηση/Εκκαθάριση

Το βήμα της αρχικοποίησης είναι υπεύθυνο για την προετοιμασία όσων στοιχείων είναι απαραίτητα για να ξεκινήσει ένα μέρος του παιχνιδιού. Τέτοια στοιχεία μπορεί να είναι το σύνολο των υφών που θα χρησιμοποιηθούν.



Το βήμα της εκκαθάρισης αναιρεί όσα έχει κάνει το βήμα προετοιμασίας, όμως με αντίστροφη σειρά.

Κύριος Βρόχος

Τα παιχνίδια οδηγούνται από έναν βρόχο παιχνιδιού, ο οποίος είναι υπεύθυνος για εκτέλεση μίας σειράς εργασιών εντός του κάθε frame. Σημαντικό είναι να αναφερθούμε στο γεγονός ότι κάποια παιχνίδια έχουν ξεχωριστούς πολλαπλούς βρόχους, ενώ άλλα παιχνίδια έχουν έναν ενοποιημένο βρόχο.

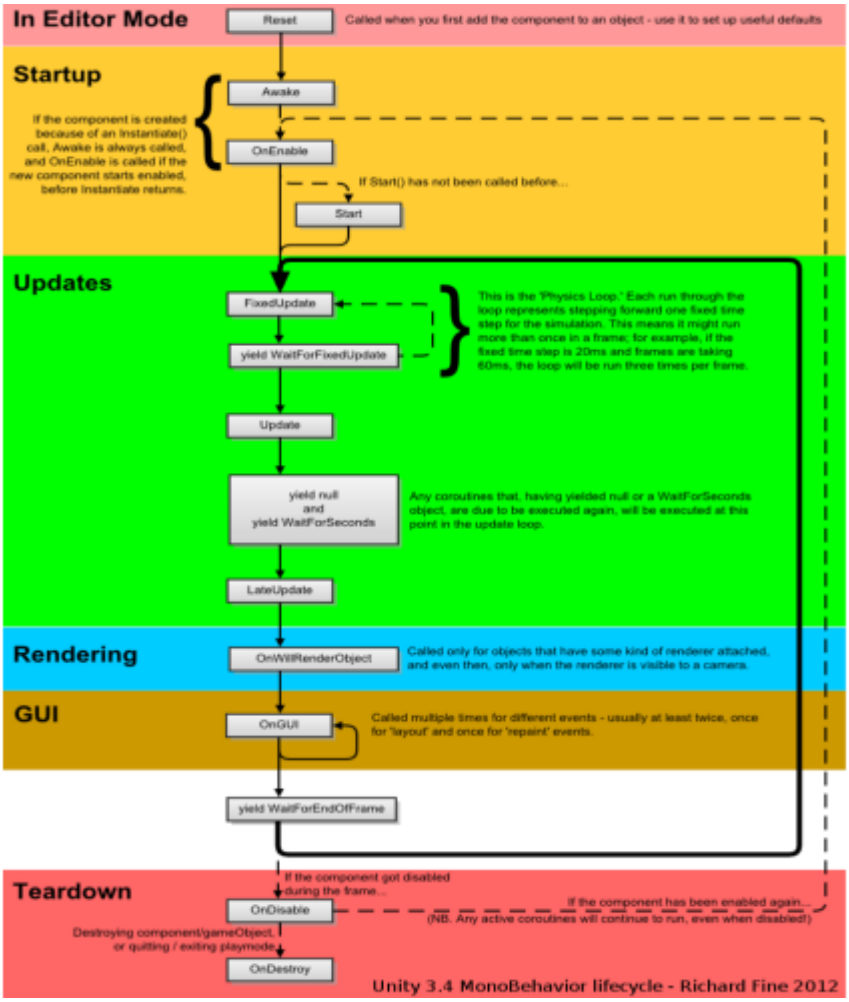
Ο κύριος βρόχος είναι υπεύθυνος για πολλές διαφορετικές εργασίες. Για την διαχείριση τόσο του χρόνου του παιχνιδιού, τον οποίο θα αναλύσουμε παρακάτω, όσο και για την διαχείριση του τρόπου που γίνεται η είσοδος του παίκτη. Εντός του κύριου βρόχου εκτελούνται όλες οι απαραίτητες λειτουργίες για την προσομοίωση του εικονικού κόσμου μας. Πράγματι, μέσα στον βρόχο ενημερώνονται τα αντικείμενα κατά τη διάρκεια του παιχνιδιού, ελέγχονται τα γραφικά και ο ήχος αλλά και ανιχνεύονται τυχόν συγκρούσεις που μπορεί να υπάρξουν και τις διαχειρίζεται κατάλληλα για την σωστή απόκριση του συστήματος. Οι παραπάνω είναι ορισμένες από τις πολλές εργασίες που λαμβάνουν χώρα εντός του κύριου βρόχου.

Βρόχος Παιχνιδιού και Ιστορία

Οι παλαιότερες κάρτες γραφικών είχαν περιορισμένες δυνατότητες και χαμηλές ταχύτητες, κάτι που ανάγκαζε τους προγραμματιστές να αναπτύσσουν τεχνικές για την βελτιστοποίηση της απόδοσης. Μία από τις τεχνικές αυτές ήταν η χρήση εξειδικευμένου υλικού που υποστήριζε την επικάλυψη ενός σταθερού αριθμού από sprites, επιτρέποντας έτσι την εμφάνιση χαρακτήρων και αντικειμένων χωρίς επιβάρυνση στον επεξεργαστή. Επιπλέον, οι προγραμματιστές αντέγραφαν ένα τμήμα του φόντου πριν σχεδιάσουν το sprite πάνω του, έτσι ώστε όταν χρειαζόταν να αφαιρέσουν ή να μετακινήσουν το sprite, επανέφεραν απλώς το αρχικό φόντο χωρίς να ανανεώσουν την οθόνη ολόκληρη. Μία ακόμα σημαντική τεχνική ήταν η χρήση της λειτουργίας XOR, η οποία επέτρεπε την εύκολη εμφάνιση και αφαίρεση των sprites χωρίς να επηρεάζουν το υπόβαθρο, κάτι ιδιαίτερα χρήσιμο για την ταχύτητα της απόδοσης. Επιπλέον, η χρήση περιορισμένων χρωματικών παλετών και χαμηλότερων αναλύσεων μείωνε την κατανάλωση μνήμης και βελτίωνε την απόδοση. Χρησιμοποιώντας αυτές τις τεχνικές, οι προγραμματιστές κατάφερναν να επιτύχουν εντυπωσιακά αποτελέσματα, παρά τους περιορισμούς του υλικού των παλαιότερων συστημάτων.

Στη σύγχρονη εποχή, η κίνηση της κάμερας απαιτεί τον πλήρη επαναϋπολογισμό της ορατής σκηνής, καθώς όλα τα αντικείμενα που βρίσκονται μέσα σε αυτήν ανανεώνονται. Αυτό σημαίνει ότι, αντί να προσπαθούμε να εντοπίσουμε ποια αντικείμενα πρέπει να ξανασχεδιαστούν ή ποιες τροποποιήσεις χρειάζονται, είναι πιο αποδοτικό να ξανασχεδιάζουμε ολόκληρη τη σκηνή από την αρχή σε κάθε frame. Αυτή η προσέγγιση επιτρέπει έναν πιο συνεπή και αποδοτικό χειρισμό της απόδοσης, ιδιαίτερα σε δυναμικά περιβάλλοντα όπου η κάμερα και τα αντικείμενα κινούνται συνεχώς, εξασφαλίζοντας ομαλή και υψηλής ποιότητας απεικόνιση της σκηνής.

Γενική μορφή βρόχου στην Unity



Στο παραπάνω σχήμα απεικονίζεται το εσωτερικό ενός βρόχου, δηλαδή την σειρά εκτέλεσης των μεθόδων για ένα αντικείμενο (GameObject) στη Unity. Αυτή η δομή διαχειρίζεται την αλληλουχία των γεγονότων που συμβαίνουν σε ένα αντικείμενο στην Unity, από τη δημιουργία του μέχρι την καταστροφή του, εξασφαλίζοντας ότι όλες οι σημαντικές μέθοδοι καλούνται με τη σωστή σειρά για την ομαλή λειτουργία του παιχνιδιού. Ας εξετάσουμε αναλυτικότερα την καθεμία από αυτές εντός του στάδιου εκτέλεσης που βρίσκεται.

1. **In Editor Mode**: Η μέθοδος Reset καλείται όταν προστίθεται το component σε ένα αντικείμενο, επιτρέποντας την αρχικοποίηση χρήσιμων προεπιλογών.
2. **Startup**:

   Awake: Εκτελείται όταν το component αρχικοποιείται, π.χ., κατά τη δημιουργία του αντικειμένου. Είναι η πρώτη μέθοδος που καλείται.

OnEnable: Καλείται αν το component είναι ενεργοποιημένο κατά τη δημιουργία του αντικειμένου.

Start: Αυτή η μέθοδος καλείται την πρώτη φορά που το component γίνεται ενεργό. Είναι σημαντικό ότι καλείται μόνο μία φορά, μετά το Awake και μόνο αν δεν έχει κληθεί προηγουμένως.

3. **Updates**:

   FixedUpdate: Εκτελείται σε τακτά διαστήματα και είναι ιδιαίτερα χρήσιμη για φυσική επειδή επιτρέπει ακριβή και προβλέψιμη προσομοίωση.

   Update: Καλείται μία φορά ανά frame και είναι η κύρια μέθοδος που χρησιμοποιείται για τις περισσότερες λειτουργίες που απαιτούν συνεχή εκτέλεση.

   LateUpdate: Καλείται μετά την Update και είναι χρήσιμη για εργασίες που πρέπει να γίνουν αφού ενημερωθούν όλα τα άλλα components, όπως η ενημέρωση της κάμερας αφού έχουν ενημερωθεί οι θέσεις των αντικειμένων.

4. **Rendering**:

   OnWillRenderObject: Εκτελείται για αντικείμενα που πρόκειται να αποδοθούν και είναι χρήσιμη για την προσαρμογή της εμφάνισης πριν από την απόδοση.

5. **GUI**:

   OnGUI: Καλείται πολλαπλές φορές για τη διαχείριση της γραφικής διεπαφής χρήστη (UI), συμπεριλαμβανομένων των γεγονότων που απαιτούν ανανέωση της εμφάνισης της διεπαφής.

6. **Teardown**:

   OnDisable: Καλείται όταν το component απενεργοποιείται και είναι κατάλληλη για την απελευθέρωση πόρων ή την ακύρωση εργασιών.

   OnDestroy: Καλείται όταν το αντικείμενο ή το component καταστρέφεται, π.χ., κατά την έξοδο από την εφαρμογή ή την αφαίρεση του αντικειμένου, επιτρέποντας τον καθαρισμό των πόρων.

Οντότητες παιχνιδιού

Ο βρόχος παιχνιδιού επιδρά στις οντότητες του παιχνιδιού, δηλαδή στα στοιχεία του εικονικού κόσμου με τα οποία ο παίκτης μπορεί να αλληλεπιδράσει. Οι οντότητες αποτελούν αυτοτελή κομμάτια διαδραστικού περιεχομένου, σχεδιασμένα έτσι ώστε να ενσωματώνουν λειτουργίες ή χαρακτηριστικά που επιτρέπουν την αλληλεπίδραση με τον παίκτη. Είναι

σημαντικό οι οντότητες να χρησιμοποιούνται μόνο για περιεχόμενο με το οποίο ο παίκτης μπορεί να αλληλοεπιδράσει λόγω επιδείνωσης της απόδοσης σε διαφορετική περίπτωση.

Επικοινωνία οντοτήτων

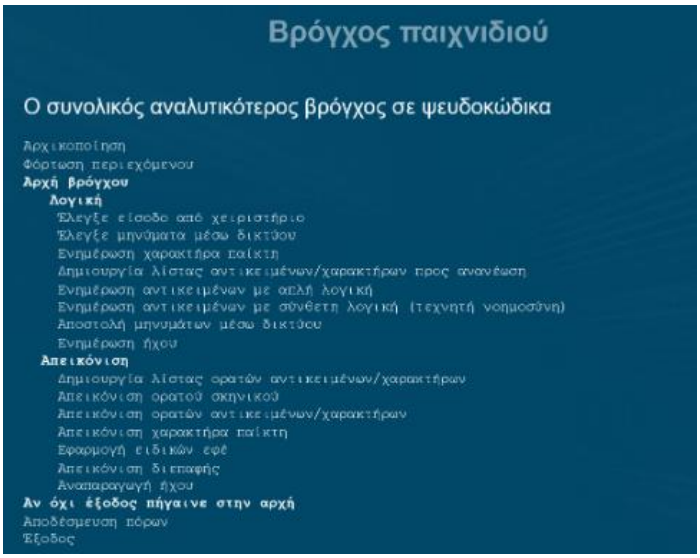
Η απλούστερη μέθοδος επικοινωνίας μεταξύ αντικειμένων σε ένα παιχνίδι είναι μέσω κλήσεων συναρτήσεων. Ωστόσο, πολλά σύγχρονα παιχνίδια χρησιμοποιούν ένα πλήρες σύστημα ανταλλαγής μηνυμάτων, όπου ένα μοναδικό αντικείμενο (συχνά με τη μορφή singleton – static class) αναλαμβάνει τη διαχείριση αυτών των μηνυμάτων. Είναι απαραίτητο να γίνεται προσεκτική διαχείριση των μηνυμάτων, τα οποία πρέπει να είναι μικρά και δομημένα σε συγκεκριμένα μοτίβα για να αποφεύγονται περιττές καθυστερήσεις. Επειδή σε κάθε frame μπορεί να αποστέλλονται χιλιάδες μηνύματα, η συχνή κλήση συναρτήσεων όπως new() και delete() αυξάνει σημαντικά τον υπολογιστικό φόρτο. Μια αποδοτικότερη προσέγγιση είναι η χρήση προκαθορισμένων δεξαμενών μνήμης, λειτουργώντας σαν «ταχυδρομικά κουτιά» για τα μηνύματα, ώστε να μειωθεί η ανάγκη για συχνές δεσμεύσεις και απελευθερώσεις μνήμης και να βελτιωθεί η απόδοση του συστήματος.

Αποσύνδεση της Απόδοσης Γραφικών

Η αποσύνδεση της απόδοσης γραφικών από τα βήματα προσομοίωσης και ενημέρωσης επιτρέπει στον βρόχο του παιχνιδιού να λειτουργεί σε διαφορετική τιμή fps από την απόδοση των γραφικών. Για παράδειγμα, ο βρόχος του παιχνιδιού μπορεί να εκτελείται με 30 fps, ενώ η απόδοση των γραφικών μπορεί να φτάνει τα 100 fps. Αυτό οδηγεί σε πιο ομαλή κίνηση και μεγαλύτερη απόκριση, βελτιώνοντας την εμπειρία του χρήστη. Ωστόσο, η εφαρμογή αυτής της προσέγγισης είναι πολύπλοκη και μπορεί να είναι επιρρεπής σε σφάλματα. Ο βρόχος που αναπαριστά αυτή την προσέγγιση περιλαμβάνει ενημέρωση χρόνου, έλεγχο για το πότε να εκτελεστεί η προσομοίωση, εκτέλεση της προσομοίωσης μόνο όταν είναι απαραίτητο, ενδιάμεση εκτίμηση της κατάστασης αν δεν εκτελεστεί η προσομοίωση και απόδοση του κόσμου.

```
while ( !IsDone() ) {
    UpdateTime();
    if(TimetoRunSimulation())
        RunSimulation();
    if(SimulationNotRun())
        InterpolateState();
    RenderWorld();
}
```

Λεπτομερής Βρόχος



Ο συνολικός βρόχος παιχνιδιού (για 60 fps) περιλαμβάνει δύο κύριες φάσεις: τη λογική και την απεικόνιση. Στη φάση της λογικής, ελέγχονται οι ενέργειες του παίκτη και τα μηνύματα δικτύου, ενημερώνεται ο χαρακτήρας του παίκτη και τα αντικείμενα (με απλή ή σύνθετη λογική, όπως η τεχνητή νοημοσύνη) και γίνεται διαχείριση του συστήματος ήχου. Στη φάση της απεικόνισης, δημιουργείται μια λίστα με τα ορατά αντικείμενα, απεικονίζεται το σκηνικό και οι χαρακτήρες, εφαρμόζονται ειδικά εφέ και αναπαράγεται ο ήχος. Ο βρόχος συνεχίζεται μέχρι να γίνει έξοδος του παίκτη από το παιχνίδι, οπότε αποδεσμεύονται οι πόροι του παιχνιδιού. Και όλα τα παραπάνω πρέπει να γίνουν μόνο σε 16.6 ms!!!

Παραδείγματα Βρόχων

Pong!

Το παιχνίδι "Pong" είναι ένα κλασικό παράδειγμα που χρησιμοποιεί έναν βασικό βρόχο παιχνιδιού (game loop). Ο βρόχος αυτός επαναλαμβάνεται συνεχώς μέχρι να τελειώσει το παιχνίδι. Στο παράδειγμα του Pong, η δομή του βρόχου έχει ως εξής:

1. **Initialize (Αρχικοποίηση):**

   Προετοιμάζει τις βασικές παραμέτρους του παιχνιδιού, όπως η θέση της μπάλας και της ρακέτας, και αρχικοποιεί το σύστημα γραφικών.

2. **Βασικός Βρόχος (do-until):**

Εκτελείται συνεχώς μέχρι το παιχνίδι να ολοκληρωθεί. Αναλυτικότερα, αποτελείται από τα εξής:

Update Ball (Ενημέρωση μπάλας - φυσική): Η θέση της μπάλας ενημερώνεται με βάση τις αρχές της φυσικής. Συνήθως χρησιμοποιούνται εξισώσεις για να κινηθεί η μπάλα. Σε περίπτωση που υπάρχει επιτάχυνση, προστίθεται στον ρυθμό αλλαγής ταχύτητας.

Update Paddle (Ενημέρωση ρακέτας - είσοδος χρήστη): Η θέση της ρακέτας ενημερώνεται σύμφωνα με την είσοδο του χρήστη (π.χ., αν πατηθεί κάποιο πλήκτρο). Χρησιμοποιείται για να μετακινηθεί η ρακέτα πάνω-κάτω και να διατηρείται εντός των ορίων.

Collision Detection (Ανίχνευση συγκρούσεων): Ελέγχει αν η μπάλα έχει χτυπήσει τη ρακέτα ή τον τοίχο. Αν ανιχνευθεί σύγκρουση, πραγματοποιείται κάποια ενέργεια, όπως αλλαγή κατεύθυνσης της μπάλας.

Draw Stuff (Σχεδίαση αντικειμένων): Εδώ ζωγραφίζεται η μπάλα, η ρακέτα και το υπόλοιπο σκηνικό του παιχνιδιού.

3. **Clean Up (Καθαρισμός):**

Όταν ο βρόχος τελειώσει, εκτελείται οποιαδήποτε διαδικασία καθαρισμού, όπως απελευθέρωση πόρων.

Asteroids

Στο παιχνίδι Asteroids, υπάρχουν πολλαπλά αντικείμενα που συνθέτουν τη σκηνή, όπως διαστημόπλοια, σφαίρες, αστεροειδείς και εχθρικά διαστημόπλοια. Το GUI (γραφικό περιβάλλον χρήστη) εμφανίζει σημαντικές πληροφορίες, όπως το σκορ και τις ζωές του παίκτη. Η σκηνή βασίζεται σε μια λίστα από οντότητες, όπου κάθε αντικείμενο είναι απλό χωρίς σύνθετη κίνηση. Ο βρόχος του παιχνιδιού ενημερώνει κάθε οντότητα, εκτελεί τις αλληλεπιδράσεις (εδώ χρειάζεται προσοχή μιας και μπορεί να είναι αρκετά απαιτητικό υπολογιστικά) και πραγματοποιεί τη γραφική απόδοση όλων των αντικειμένων. Αντί για λίστα αντικειμένων, θα μπορούσε να χρησιμοποιηθεί ένα γράφημα σκηνής για πιο σύνθετες σκηνές. Στο γράφημα σκηνής, οι δομές ακολουθούν ένα κατευθυνόμενο γράφημα, όπου σύνθετα αντικείμενα διαμοιράζονται σε επιμέρους τμήματα, βελτιώνοντας τη διαχείριση και την απόδοση στη σκηνή.

Χρόνος

Έννοιες Χρόνου

Στα παιχνίδια, ο πραγματικός χρόνος μπορεί να μετράται με το χρονόμετρο υψηλής ανάλυσης της CPU. Ο έλεγχος του παιχνιδιού συνήθως βασίζεται σε πραγματικό χρόνο, αλλά είναι πιθανό να επιβραδυνθεί, να επιταχυνθεί ή ακόμα και να διακοπεί. Υπάρχουν επίσης οι έννοιες του τοπικού και του καθολικού χρόνου, που αφορούν τη ροή των animations στο παιχνίδι, όπου ο καθολικός χρόνος μπορεί να αντιστοιχιστεί με οποιονδήποτε τρόπο στον τοπικό.

Χρόνος

1. **Χρόνος Frame (Frame Time)**: Ο χρόνος κάθε frame δεν είναι πάντα σταθερός. Σε κάθε frame εκτελούνται συγκεκριμένες διεργασίες που είναι κρίσιμες για την απόδοση της σκηνής.
2. **Χρόνος Προσομοίωσης Φυσικής**: Περιλαμβάνει τα βήματα της φυσικής προσομοίωσης, τα οποία μπορεί να εκτελούνται πιο γρήγορα από τον χρόνο του frame για ακριβή προσομοίωση. Είναι κρίσιμο να αποφεύγονται πολύ μεγάλα βήματα στη προσομοίωση για να μην υπάρχουν ασυνέπειες (για αυτό και η προσομοίωση μπορεί να εκτελείται περισσότερες από μία φορές σε κάθε βρόχο).
3. **Πραγματικός Χρόνος**: Είναι ο χρόνος του συστήματος που χρησιμοποιείται για τον συγχρονισμό μουσικής, βίντεο και άλλων στοιχείων που απαιτούν ακριβή συγχρονισμό.

Στην περίπτωση όπου ένα frame χάνεται (frame drop), παρατηρείται ότι η απόδοση του παιχνιδιού μπορεί να επηρεαστεί σημαντικά, καθώς κάποιες λειτουργίες καθυστερούν για να διατηρηθεί η συνέπεια της εμπειρίας του χρήστη.

Χρήση του χρόνου μεταξύ δύο Frames (deltaTime)

Στα παλαιότερα παιχνίδια δεν μετρούσαν τον πραγματικό χρόνο που περνούσε μεταξύ δύο frames. Αντί για αυτό, τα αντικείμενα του παιχνιδιού μετακινούνταν κατά μία σταθερή ποσότητα σε κάθε επανάληψη, καθορίζοντας έτσι τον ρυθμό κίνησης ανάλογα με την ταχύτητα της CPU. Αυτό είχε ως αποτέλεσμα, όταν οι υπολογιστές αναβαθμίζονταν, τα παιχνίδια να τρέχουν πολύ πιο γρήγορα. Το κουμπί "turbo" προστέθηκε σε κάποιες περιπτώσεις για να διορθώσει αυτό το πρόβλημα.

Η λύση σε αυτό το πρόβλημα προκύπτει κάνοντας μία κανονικοποίηση της μεταβολής της ταχύτητας χρησιμοποιώντας το χρόνο μεταξύ δύο frame (deltaTime). Αυτό σημαίνει ότι όσο περισσότερα frame αποδίδονται ανά δευτερόλεπτο τόσο πιο μικρός είναι ο χρόνος και τόσο πιο μικρή θα είναι η μεταβολή της κίνησης. Αυτό σημαίνει ότι η κίνηση είναι πιο ομαλή όταν

τα fps είναι υψηλά ενώ όταν τα fps είναι χαμηλά (π.χ., 20 fps) τότε η κίνηση μοιάζει «σπασμένη».

Υπολογισμός deltaTime

Το deltaTime είναι ένα κρίσιμο στοιχείο στον προγραμματισμό των παιχνιδιών. Συνήθως υπολογίζεται άμεσα διαβάζοντας τον χρόνο του τρέχοντος frame και τον υπολογισμό της διαφοράς από το προηγούμενο frame. Αυτός ο τρόπος όμως έχει προβλήματα, καθώς χρησιμοποιείται ο χρόνος του τρέχοντος frame ως εκτίμηση για τον χρόνο των επόμενων frame, κάτι που μπορεί να οδηγήσει σε καθυστερήσεις και ασταθή απόδοση. Μια πιο ακριβής μέθοδος είναι η χρήση ενός τρέχοντος μέσου όρου που "λειαίνει" τις διακυμάνσεις.

Ρύθμιση του Ρυθμού

Είναι προτιμότερο να ρυθμίζεται ο ρυθμός του παιχνιδιού ώστε να διατηρείται ο αριθμός FPS σε σταθερό επίπεδο. Έτσι, το παιχνίδι είναι συνεπές και διευκολύνεται η καταγραφή και αναπαραγωγή του.