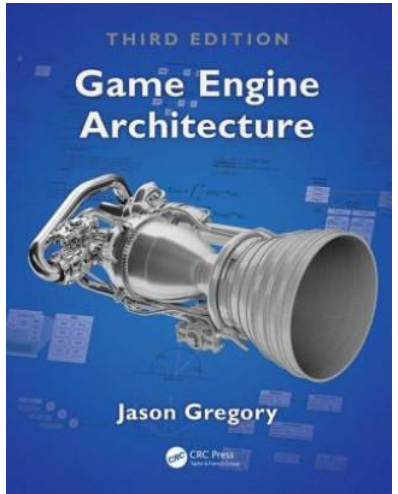
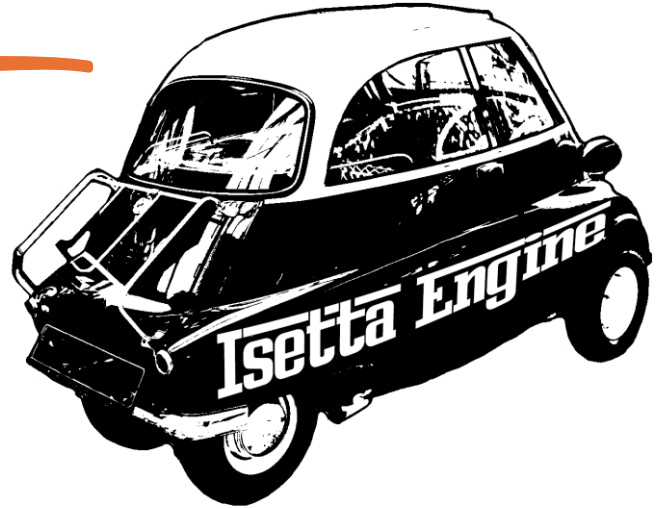
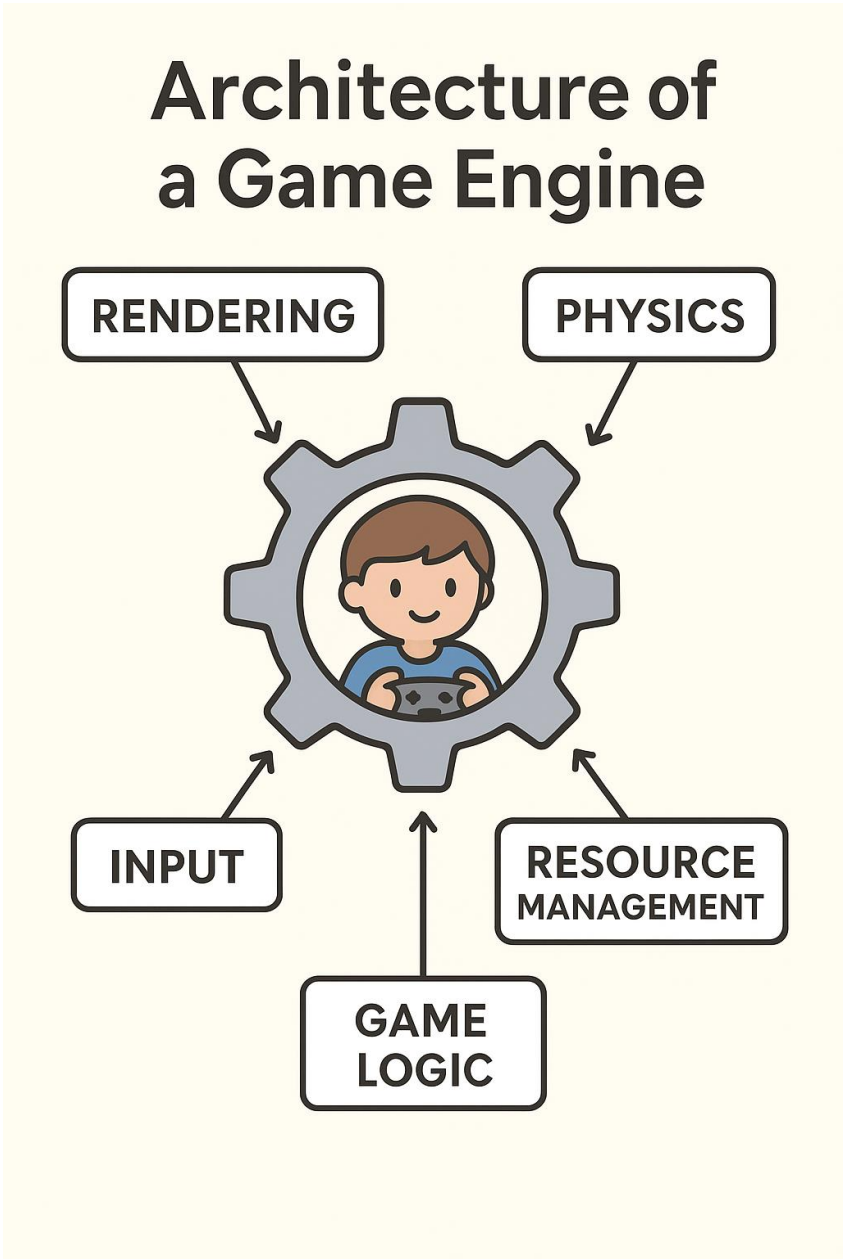
Βασικά Στοιχεία Μηχανών Βιντεοπαιχνιδών







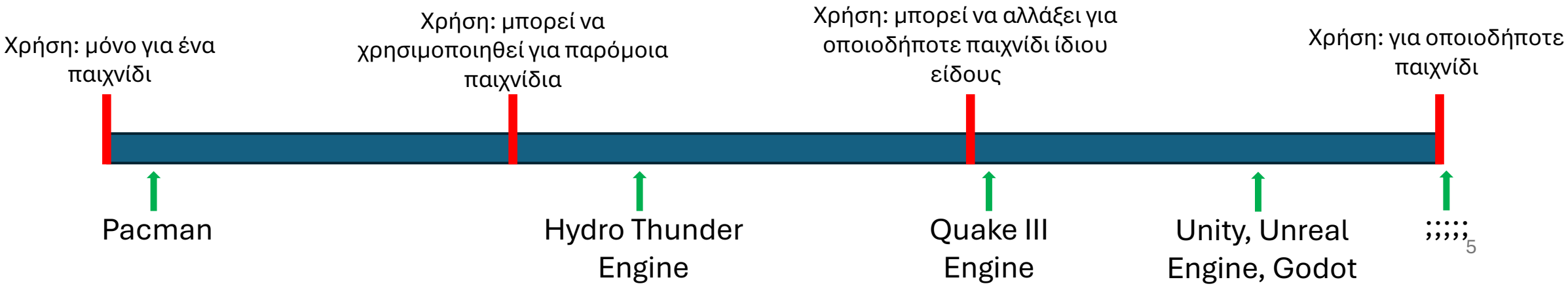
Ο Προγραμματισμός Παιχνιδιών είναι Ζόρικος

- Οι παίκτες θέλουν όμορφα γραφικά
- Το παιχνίδι πρέπει να τρέχει γρήγορα (30fps+)
- Το AI δεν είναι ακριβώς τετριμμένο
- Θέλουμε δικτύωση αλλά χωρίς καθυστέρηση
- Η φυσική είναι ήδη δύσκολη. Τώρα, κάντε τη σε πραγματικό χρόνο…

Και κάντε τα όλα έγκαιρα για τα Χριστούγεννα (ή για τις εξετάσεις αν θέλετε ☺)…

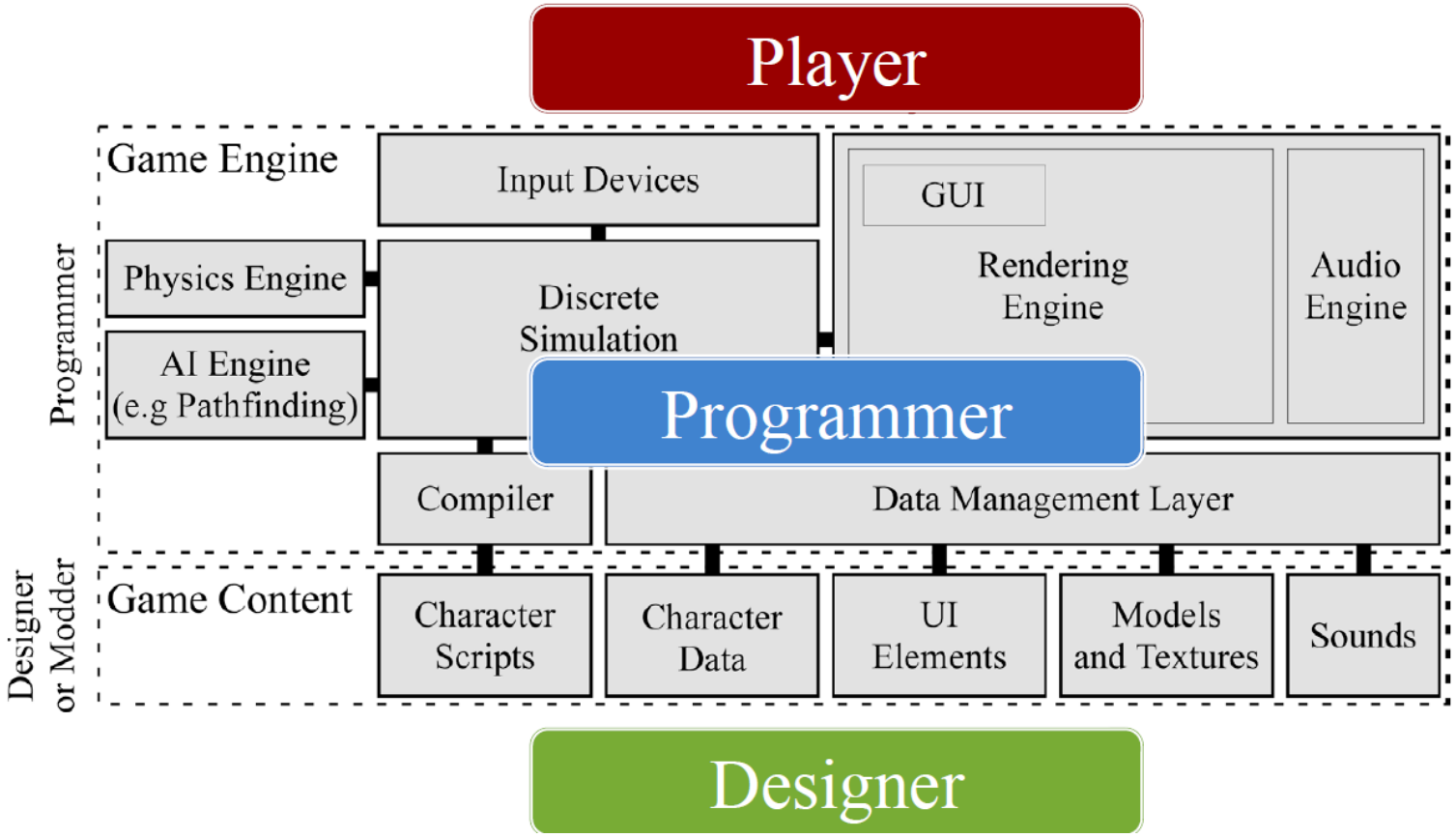
Τί Είναι η Μηχανή Παιχνιδιών;

- Ο όρος εμφανίστηκε τη δεκαετία του 1990
- Το Doom ήταν το πρώτο παιχνίδι που είχε ένα σαφή διαχωρισμό μεταξύ των:
  - Βασικών στοιχείων του παιχνιδιού (όπως το σύστημα απόδοσης γραφικών, το σύστημα ανίχνευσης σύγκρουσης, το σύστημα ήχου)
  - Καλλιτεχνικών στοιχείων (μοντέλα, υφές, κίνηση)
  - Κανόνων του παιχνιδιού
- Ακολούθησαν αυτές τις αρχές τα Quake III και Unreal (πούλησαν άδειες για τη μηχανή και τα εργαλεία τους)
- Η μηχανή παιχνιδιών αφορά μία οικογένεια αρχιτεκτονικών που βασίζεται σε δεδομένα (data-driven) και είναι επαναχρησιμοποιήσιμη και επομένως δεν έχουν περιεχόμενο παιχνιδιού (ως επί το πλείστο, αληθής πρόταση)



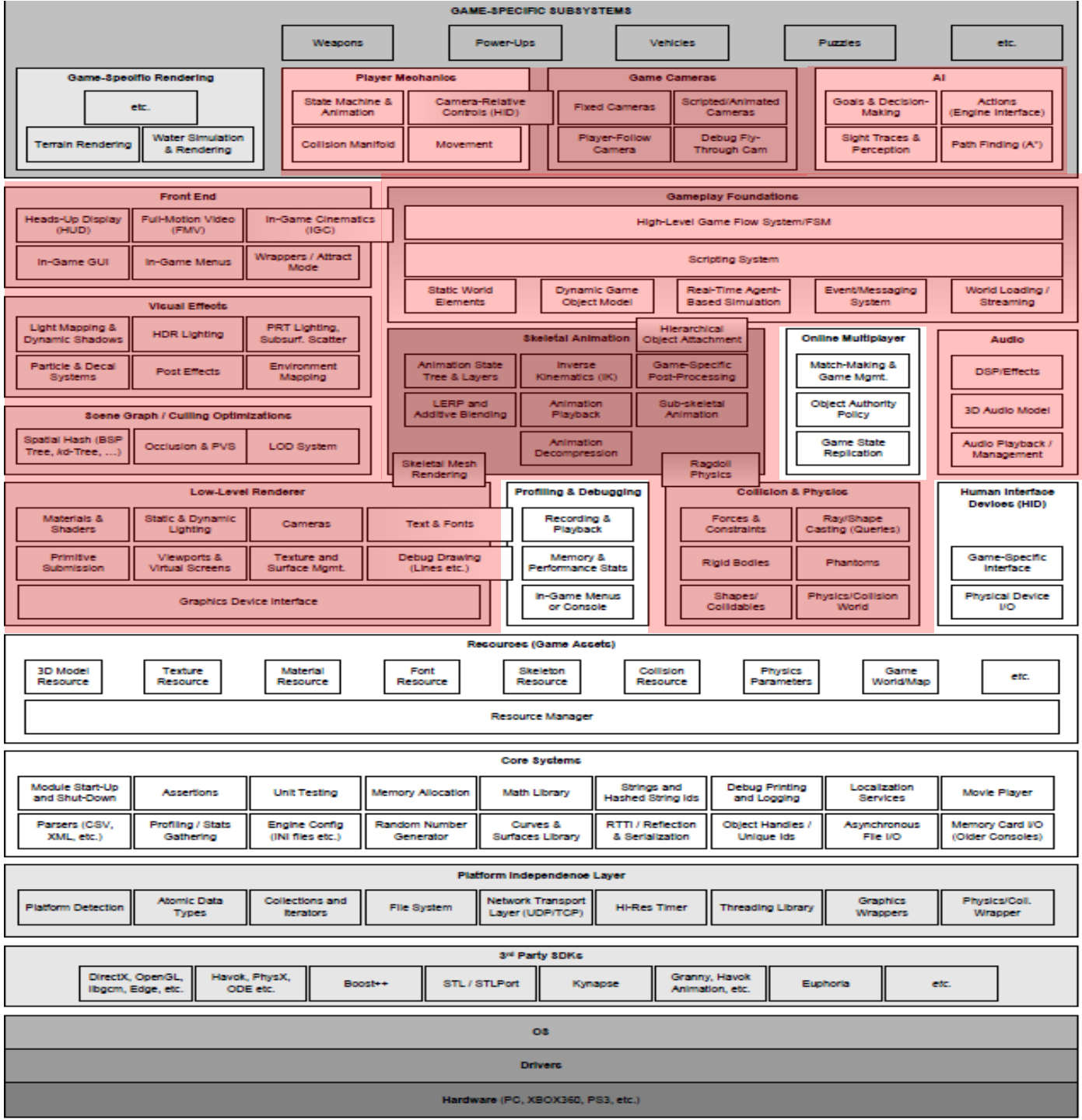
Game Engine Architecture

Αρχιτεκτονική Μηχανών Βιντεοπαιχνιδιών



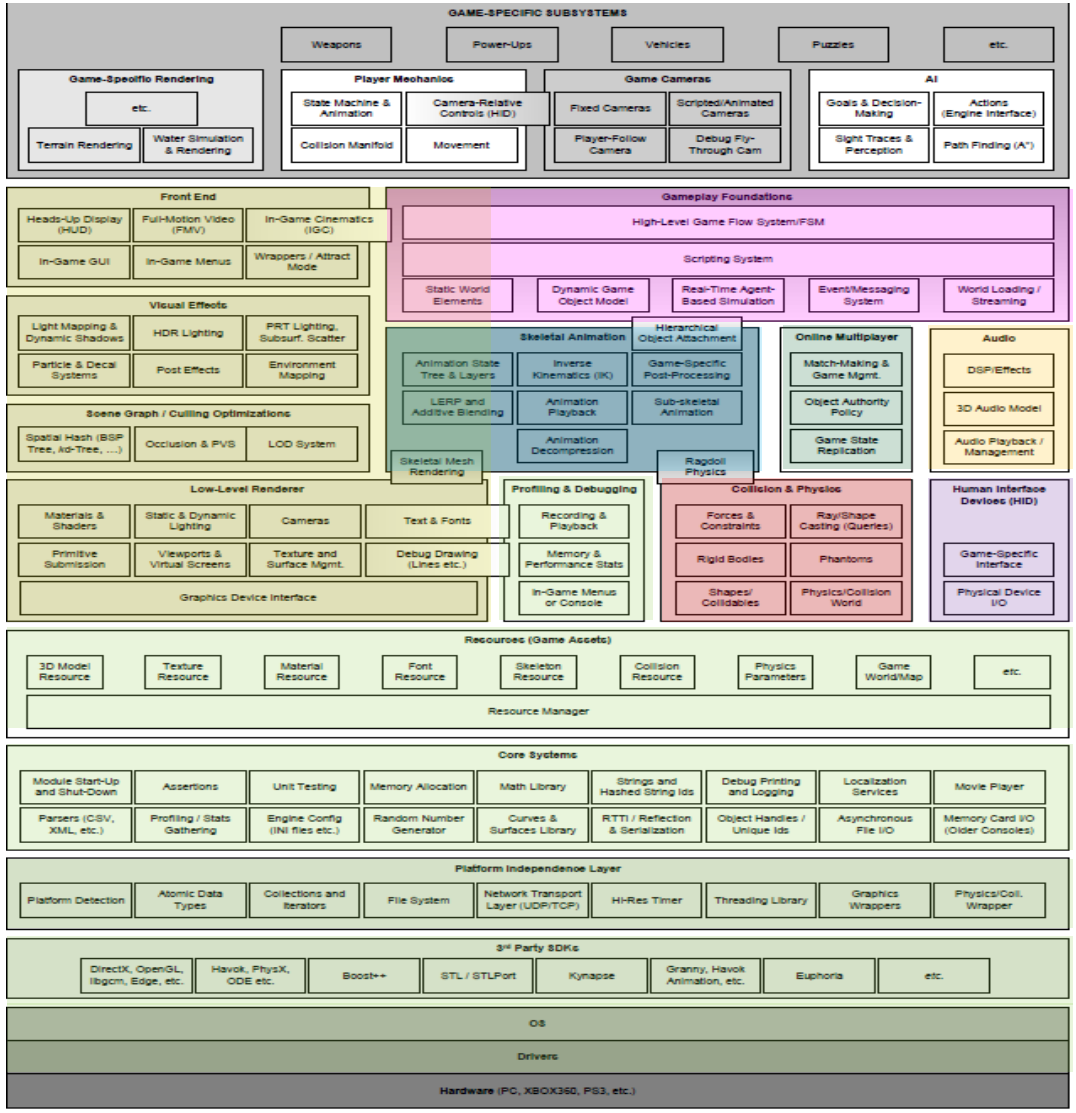
Αρχιτεκτονική

- Αποτελείται από τη σουίτα εργαλείων και συστήματων εκτέλεσης
- Τεράστια
  - Εκτείνεται από το επίπεδο υλικού έως το επίπεδο εφαρμογής υψηλού επιπέδου
- Σχεδιασμένη σε επίπεδα
  - Αποφεύγει τις κυκλικές εξαρτήσεις για μεγιστοποίηση της επαναχρησιμοποίησης και της δυνατότητας δοκιμής

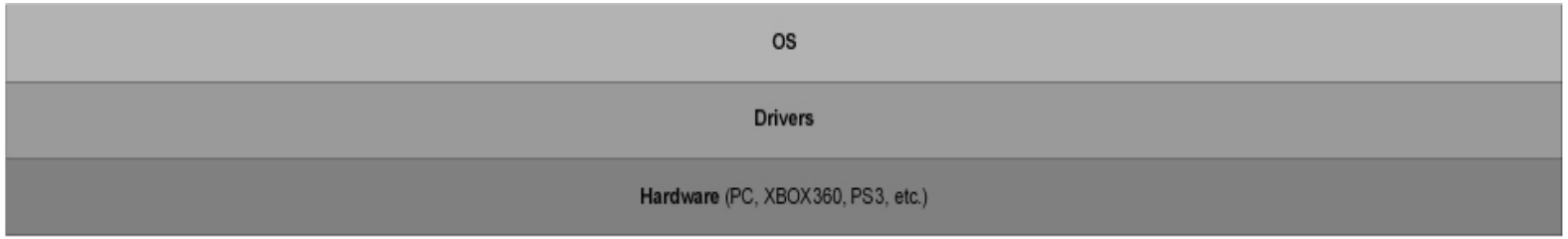


Συστήματα Μηχανής

- Συστήματα χαμηλού επιπέδου
- SDKs τρίτων
- Επίπεδο ανεξαρτησίας πλατφόρμας
- Βασικά Συστήματα (core systems)
- Διαχειριστής πόρων (resources manager)
- Profiling/Εντοπισμός σφαλμάτων
- Μηχανή απόδοσης γραφικών (rendering engine)
- Συγκρούσεις και Φυσική
- Κίνηση (Animation)
- Συσκευές ανθρώπινης διεπαφής
- Ήχος
- Σύστηματα τρόπου παιχνιδιού (gameplay systems)
- Δικτύωση
- Υποσυστήματα που αφορούν συγκεκριμένα παιχνίδια



Συστήματα Χαμηλού Επιπέδου



Υλικό

- Αυτό είναι το σύστημα στο οποίο το παιχνίδι τρέχει

Οδηγοί Συσκευών (Device Drivers)

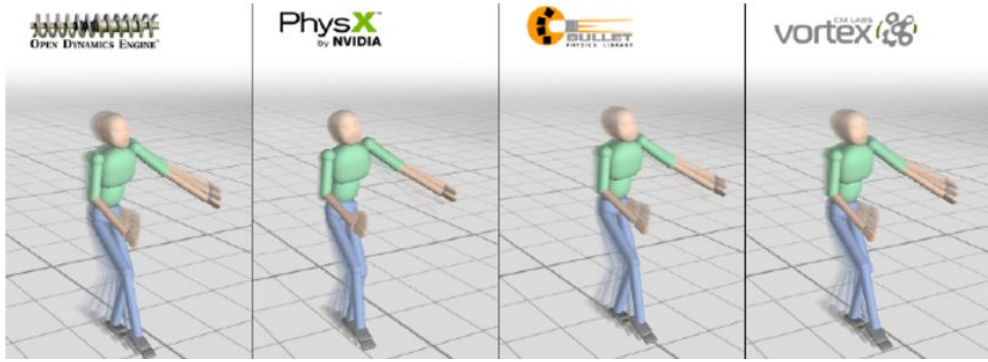
- Διαχωρίζει το OS και τα ανώτερα συστήματα από χαμηλού επιπέδου λεπτομέρειες επικοινωνίας με τις συσκευές

Λειτουργικό Σύστημα

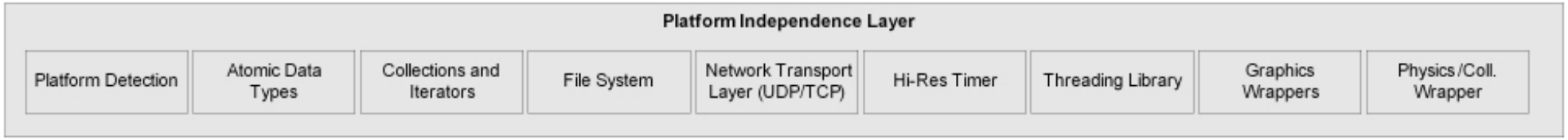
- Διαχειρίζεται την εκτέλεση πολλαπλών προγραμμάτων σε μία μηχανή
- Στις κονσόλες είναι εξαιρετικά λιτό

SDKs (τρίτων)

- Δομές Δεδομένων και Αλγόριθμοι
  - STL – C++ Standard Template Library
  - Boost
- Γραφικά
  - OpenGL και DirectX
- Συγκρούσεις και Φυσική
  - ODE, PhysX, Bullet, Vortex
- Κίνηση Χαρακτήρα
  - Granny 3D
- Τεχνητή Νοημοσύνη
  - Unity ML-Agents Toolkit, Unreal's AI Toolkit

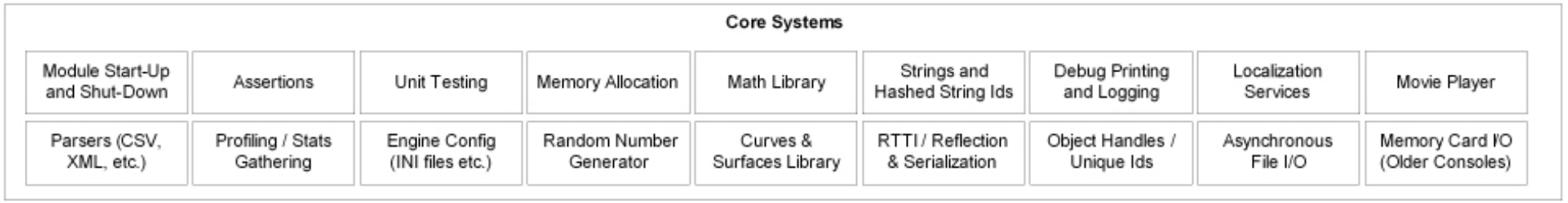


Επίπεδο Ανεξαρτησίας Πλατφόρμας



- Επιτρέπει την ανάπτυξη της μηχανής χωρίς την ανησυχία της υποκείμενης πλατφόρμας
- Παρέχει περιτυλίγματα (wrappers) για συγκεκριμένες λειτουργίες
- Περιλαμβάνει λειτουργίες όπως θεμελιώδης τύπους, υποστήριξη δικτύου, συστήματα αρχείων, κ.λπ.

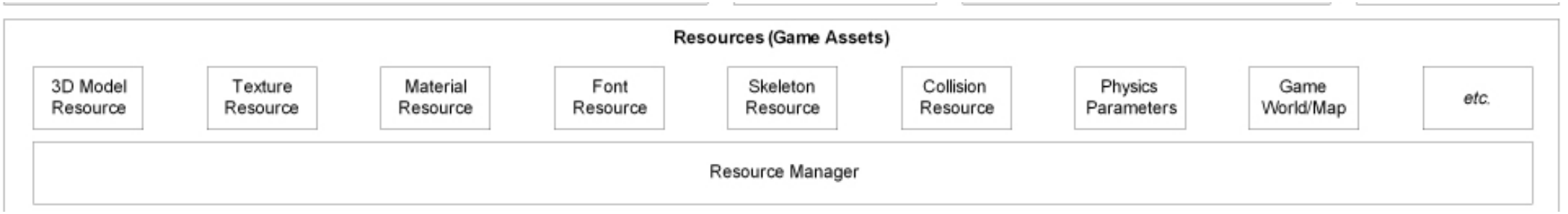
Συστήματα Πυρήνα



Μεταξύ άλλων:

- Διαχείριση μνήμης
- Βιβλιοθήκη μαθηματικών πράξεων – διανύσματα, μητρώα, αριθμητική ολοκλήρωση
- Προσαρμοσμένες δομές δεδομένων

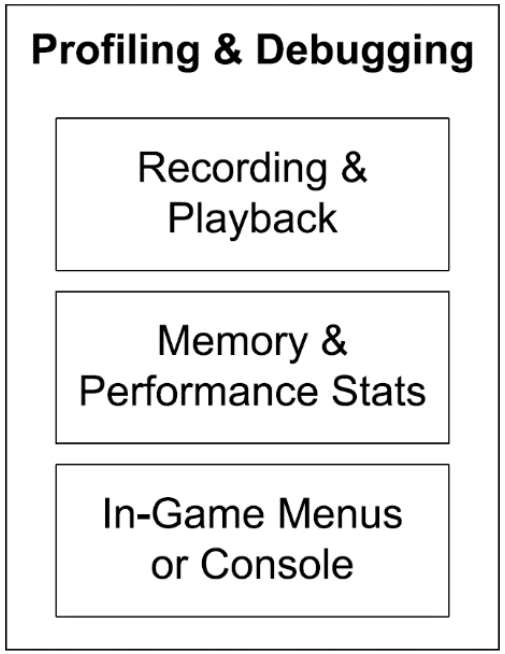
Διαχειριστής Πόρων



- Παρέχει μια ενοποιημένη διεπαφή για πρόσβαση στα στοιχεία του παιχνιδιού (assets)
- Το επίπεδο πολυπλοκότητας είναι κατά περίπτωση
  - Συχνά οι προγραμματιστές παιχνιδιών πρέπει να κάνουν απευθείας φόρτωση πόρων
  - Κάποιες μηχανές αναλαμβάνουν πλήρως την αποσυμπίεση και την πολύπλοκη διαχείριση των στοιχείων

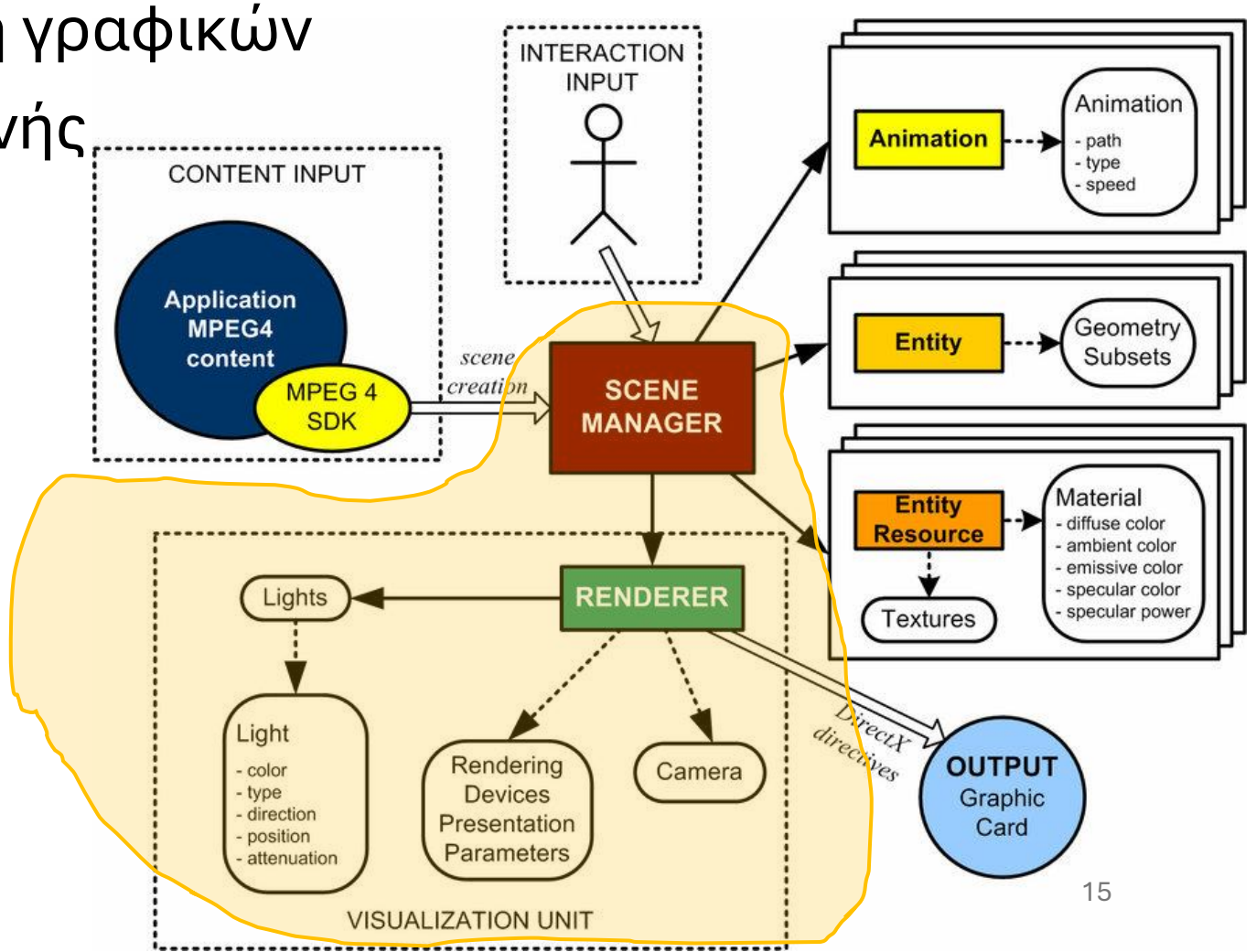
Profiling/Εντοπισμός Σφαλμάτων

- Χρονομέτρηση κώδικα
- Αναφορά στατιστικών στην οθόνη
- Αποθήκευση στατιστικών απόδοσης
- Καθορισμός χρήσης μνήμης
- Εγγραφή και αναπαραγωγή γεγονότων



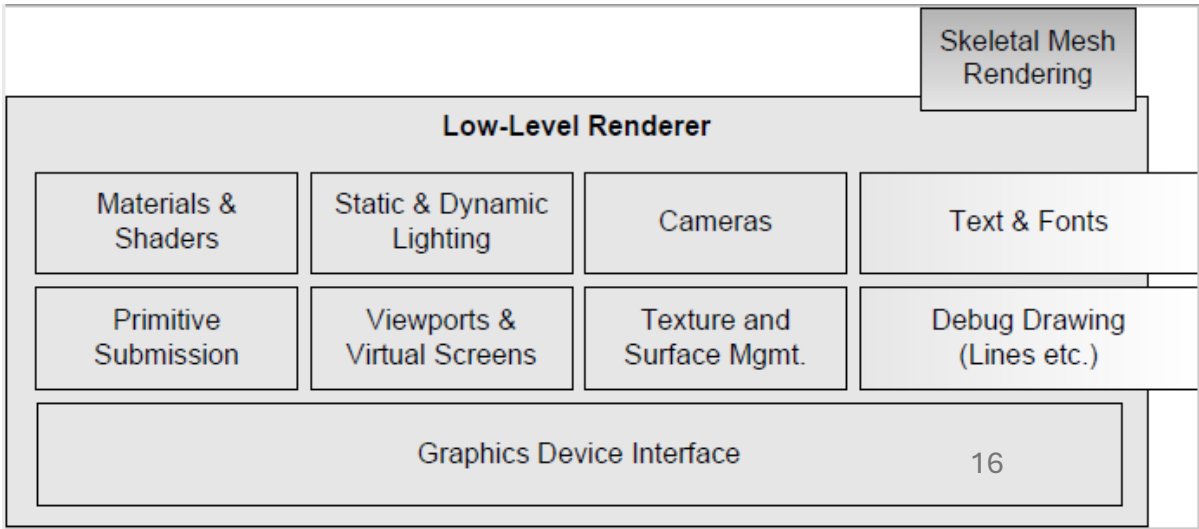
Μηχανή Απόδοσης Γραφικών

- Χαμηλού επιπέδου απόδοση γραφικών
- Διαχείριση γραφήματος σκηνής
- Οπτικά εφέ
- Front end



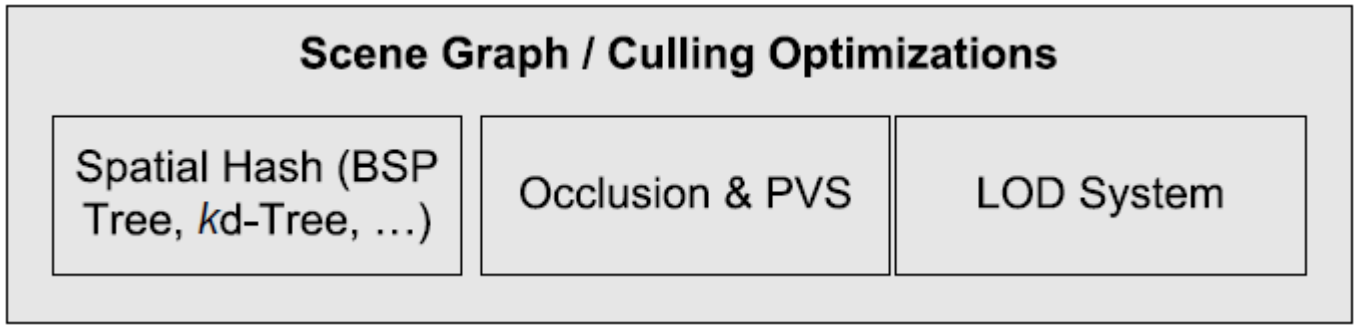
Χαμηλού Επιπέδου Απόδοση Γραφικών (Κυρίως η GPU)

- Επικεντρώνεται στην απόδοση θεμελιωδών στοιχείων όσο το δυνατόν καλύτερα
  - Δεν λαμβάνει υπόψη την ορατότητα
- Διεπαφή συσκευής γραφικών
  - Πρόσβαση και απαρίθμηση των συσκευών γραφικών
  - Αρχικοποίηση των συσκευών
  - Ρύθμιση προσωρινής αποθήκευσης
- Άλλα
  - Αναπαράσταση των θεμελιωδών γεωμετρικών αντικειμένων
  - Διεπαφή κάμερας
  - Σύστημα υλικών (materials)
  - Στατικό & δυναμικό σύστημα φωτισμού
  - Κείμενο και γραμματοσειρές



Γράφημα Σκηνής

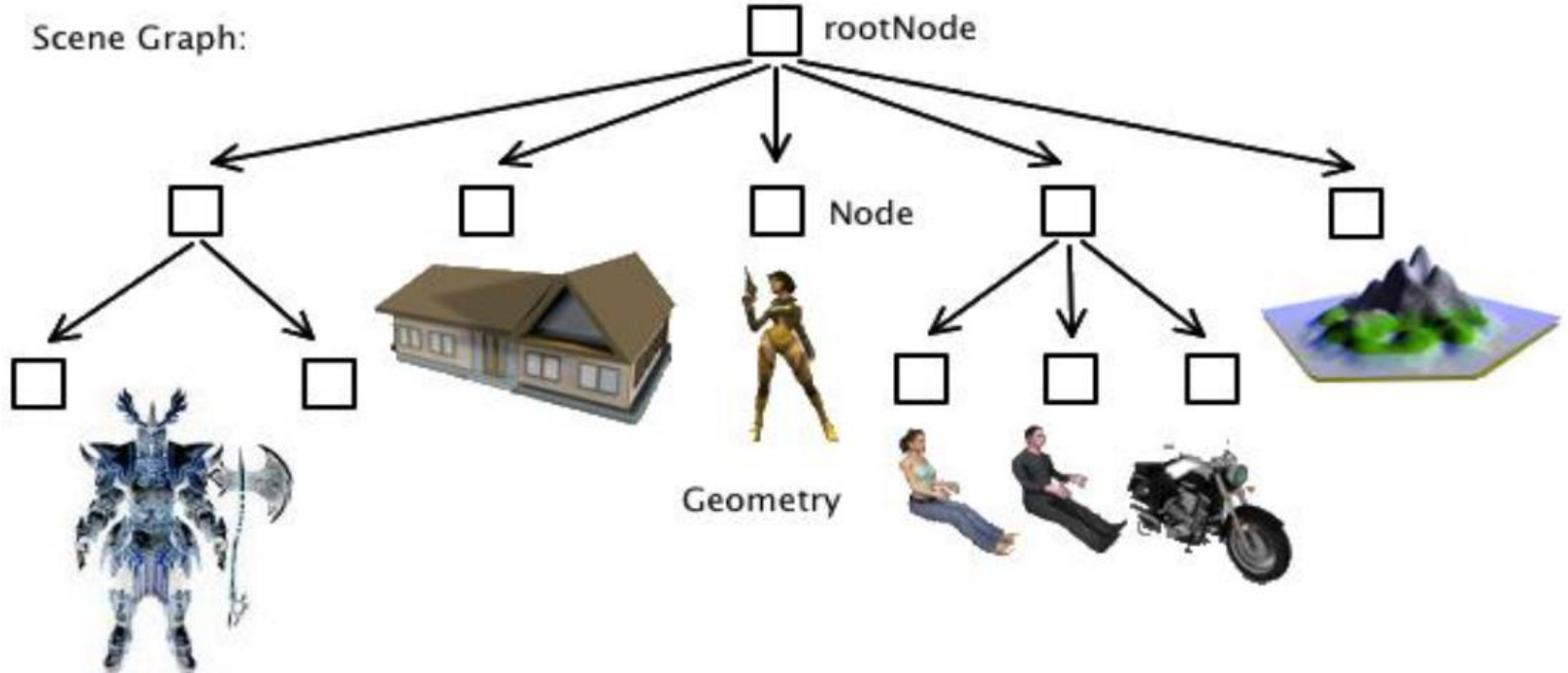
- Περιορίζει τον αριθμό των θεμελιωδών πράξεων που υποβάλλονται
- Χρησιμοποιεί frustum culling – αφαιρέστε πράγματα έξω από την κάμερα
- Χωρική υποδιαίρεση
  - BSP, quadtree, octree, kd-tree



Γράφημα Σκηνής

Το γράφημα σκηνής αντιπροσωπεύει τον τρισδιάστατο κόσμο

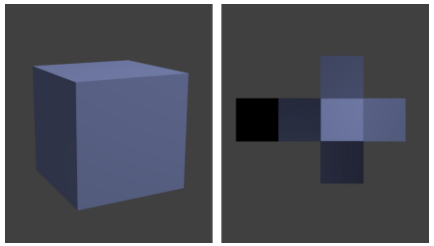
- Τα φύλλα του δένδρου αντιπροσωπεύουν δεδομένα (Γεωμετρία)
- Οι εσωτερικοί κόμβοι ομαδοποιούν και διαχειρίζονται δεδομένα

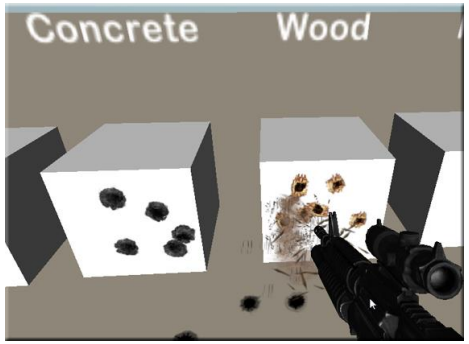


Οπτικά Εφέ

Visual Effects		
Light Mapping & Dynamic Shadows	HDR Lighting	PRT Lighting, Subsurf. Scatter
Particle & Decal Systems	Post Effects	Environment Mapping

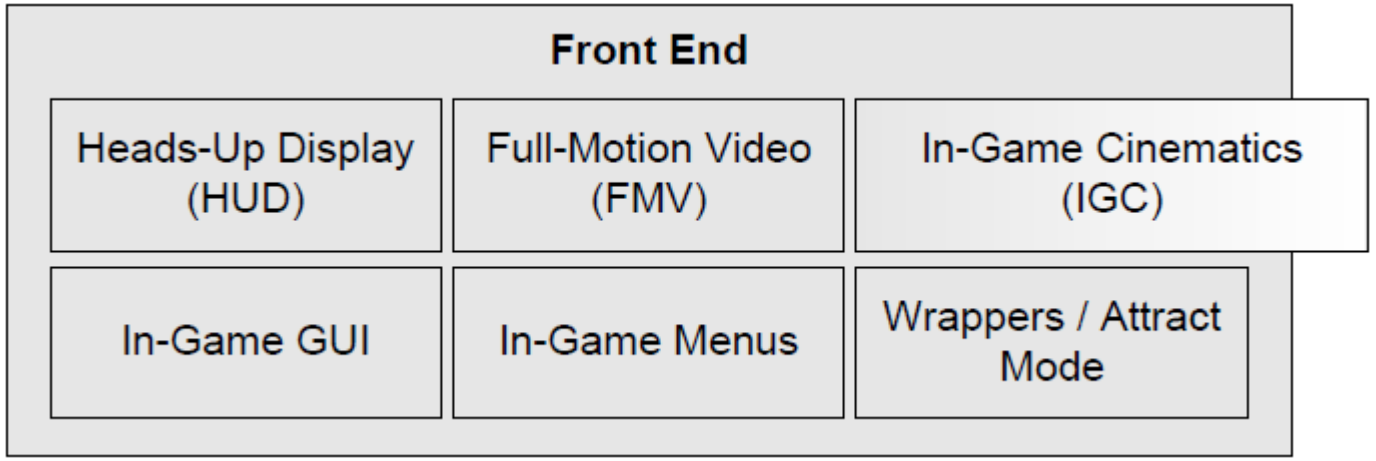
- Συστήματα Σωματιδίων (Particle systems) – π.χ., φωτιά
- Decal συστήματα
- Light mapping
- Δυναμική Σκίαση (Dynamic shadows)
- Μετα-εφέ Πλήρους Οθόνης (Full screen post effects)





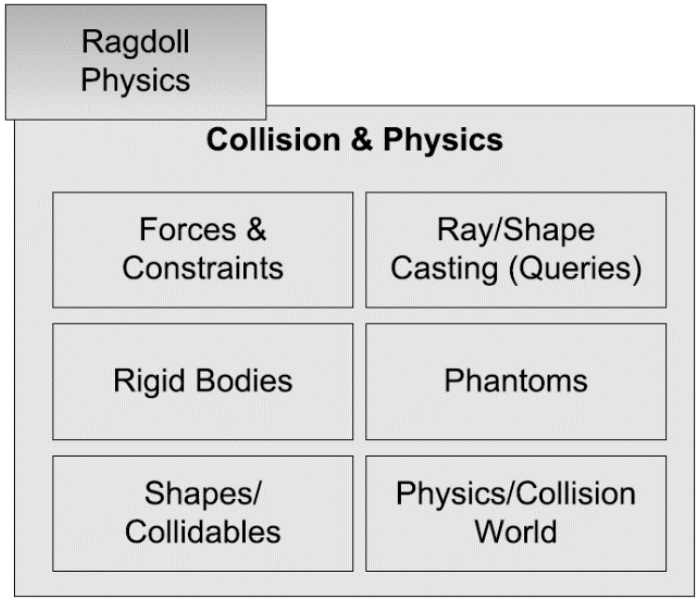
Front end

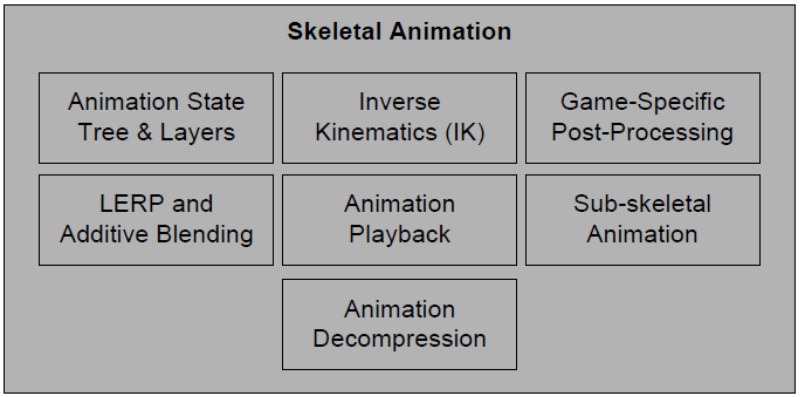
- HUD (Head-Up Display)
- Μενού
- GUI για διαχείριση χαρακτήρα
- Video για cut scenes



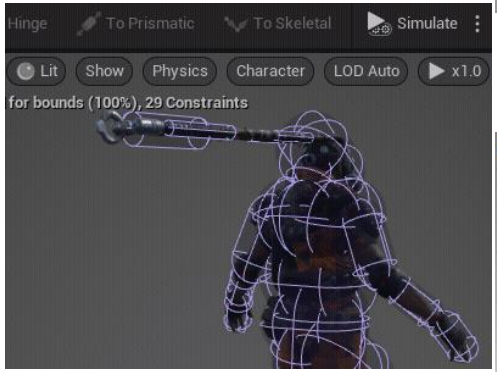
Συγκρούσεις και Φυσική

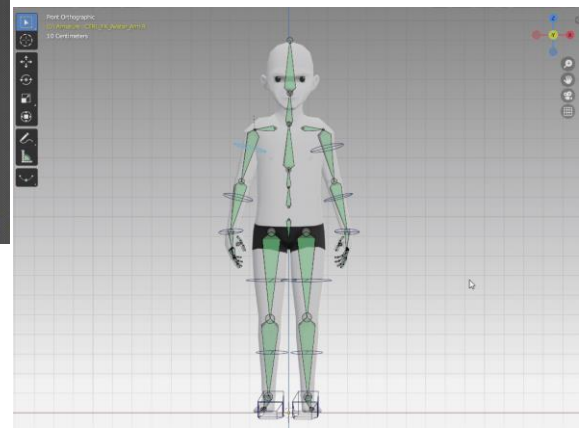
- Συνήθως δυναμική άκαμπτων σωμάτων
- Μηχανή Φυσικής – δύσκολο εγχείρημα από μόνο του
- Χρήση διαθέσιμων βιβλιοθηκών
  - Havok
  - PhysX
  - ODE
  - Bullet



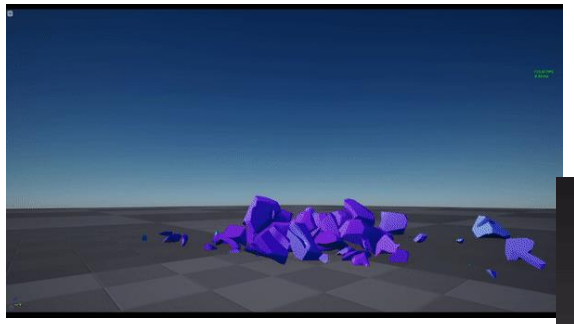


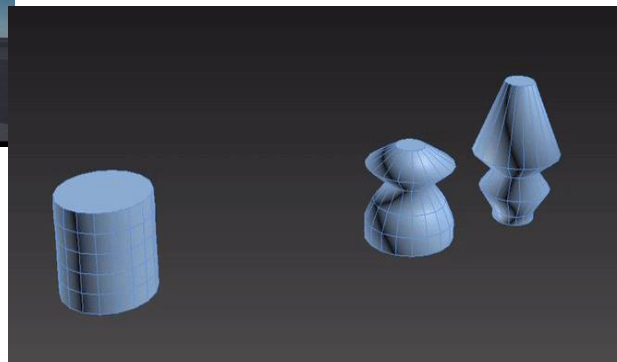
Κινούμενο Σχέδιο (Animation → Εμψύχωση Αντικειμένων)





- Χρησιμοποιούνται συνήθως πέντε τύποι κίνησης σχεδίου:
  - Sprite/texture κίνηση
  - Κίνηση ιεραρχίας άκαμπτου σώματος (Rigid body hierarchy animation)
  - Σκελετική κίνηση (skeletal animation)
  - Κίνηση Κόμβων (Vertex animation)
  - Μορφοποίηση (Morphing)



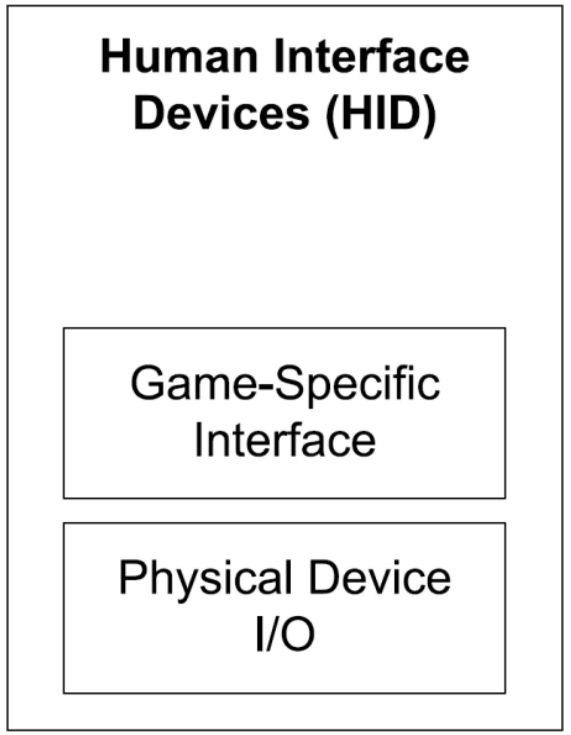


- Η πιο δημοφιλής είναι η σκελετική κίνηση

HID

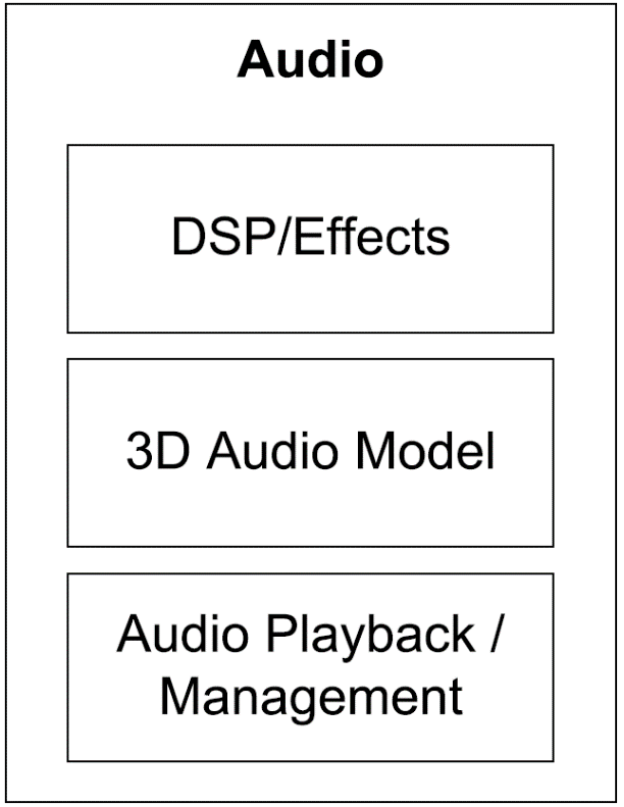
Μετατρέπει τα ακατέργαστα δεδομένα σε χρήσιμη πληροφορία

- Πληκτρολόγιο και ποντίκι
- Joypads
- Εξειδικευμένα χειριστήρια
- Kinect
- VR headsets
- …



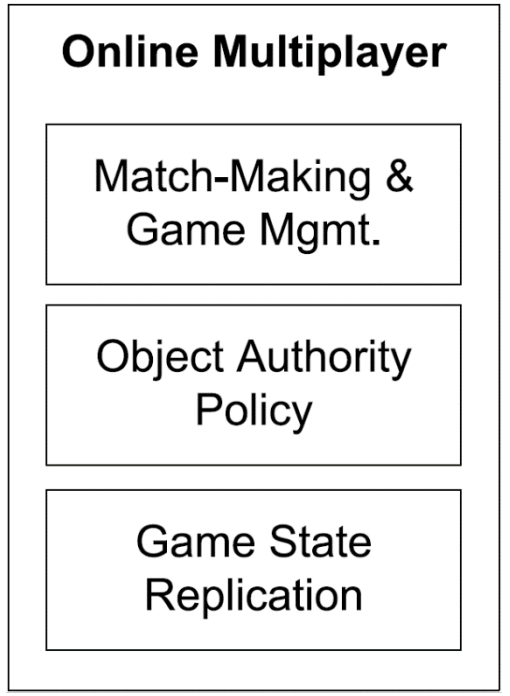
Ήχος

- Πολλές φορές τον παραμελούμε και τον αφήνουμε για το τέλος
- Διαφέρει σε πολυπλοκότητα ανάλογα με την ανάγκη
- Πολλά παιχνίδια χρησιμοποιούν υπάρχοντα εργαλεία
  - XACT
  - Scream

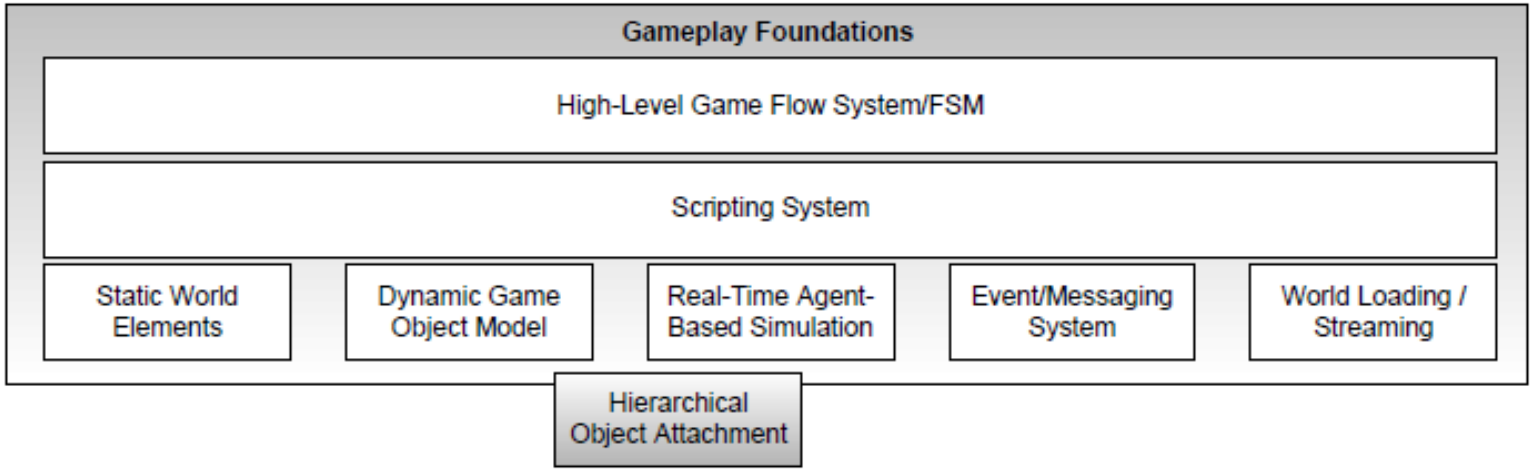


Δικτύωση/Πολλαπλοί Παίκτες

- Ενιαία οθόνη – πολλοί παίκτες στην ίδια οθόνη
- Διαχωρισμένη οθόνη – πολλαπλές προοπτικές στην ίδια οθόνη
- Δικτυωμένο – πολλοί υπολογιστές δικτυωμένοι μαζί
- Τεράστια διαδικτυακά παιχνίδια για πολλούς παίκτες – εκτελούνται σε κεντρικό διακομιστή.
- Κάποια θέματα (μεταξύ πολλών):
  - Lag: πρόβλεψη από την πλευρά του client ανάλογα με το παιχνίδι
  - Lag: UDP πρωτόκολλο λόγω ταχύτητας αν και λιγότερο αξιόπιστο
  - Cheating
  - …



Θεμελιώδες Σύστημα Υποστήριξης Gameplay



Όλα όσα κάνουν ένα παιχνίδι να είναι παιχνίδι

- Φόρτωση κόσμων
- Μοντέλα αντικειμένων παιχνιδιού
- Στατικά στοιχεία του κόσμου
- Προσομοιώσεις πρακτόρων σε πραγματικό χρόνο

Σύστημα Γεγονότων & Scripting Σύστημα

Σύστημα γεγονότων (event system)

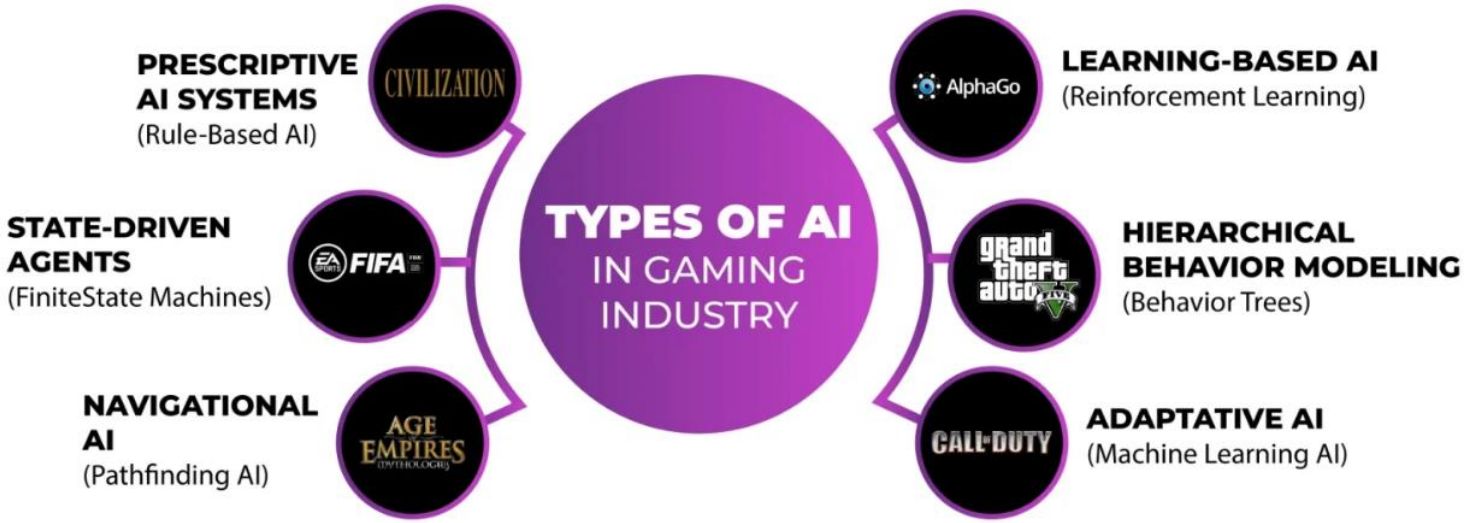
- Τα αντικείμενα πρέπει να επικοινωνούν μεταξύ τους
- Πιο εύκολο να το χειριστείς μέσω ενός κοινού συστήματος
- Τα αντικείμενα στέλνουν μηνύματα που δρομολογούνται στον χειριστή συμβάντων

Scripting σύστημα

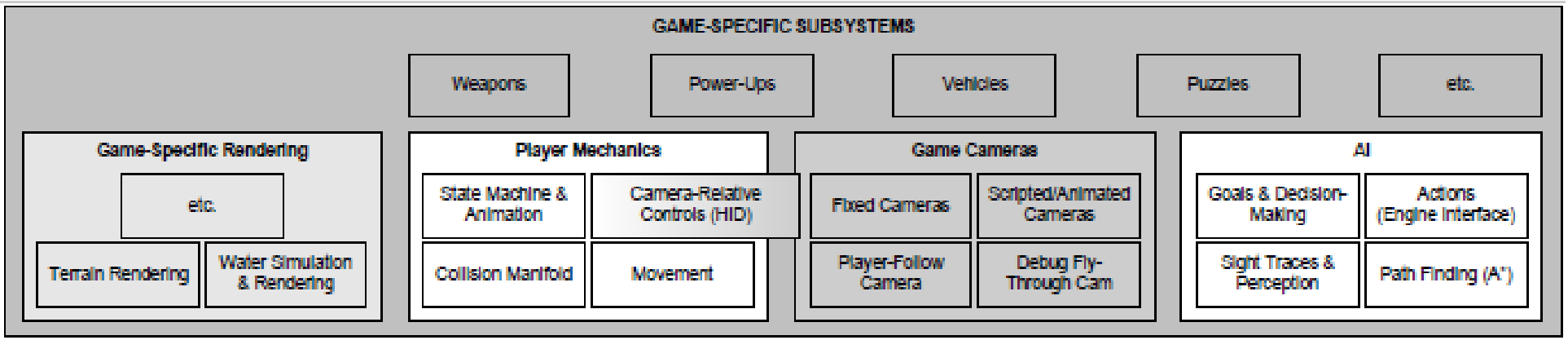
- Επιτρέπει τη δημιουργία νέας λογικής παιχνιδιού χωρίς μεταγλώττιση
- Επιταχύνει σημαντικά την ανάπτυξη λογισμικού

Βασικά Συστήματα AI

- Πράκτορες (αντίληψη περιβάλλοντος, μνήμη, αντίδραση σε ερεθίσματα,...)
- Σχεδιασμός διαδρομής (Path planning)
  - Παραγωγή δομής Navmesh (βοηθά στο σχεδιασμό διαδρομής)
  - Αποφυγή εμποδίων



Ειδικά υποσυστήματα που αφορούν το κάθε παιχνίδι ξεχωριστά



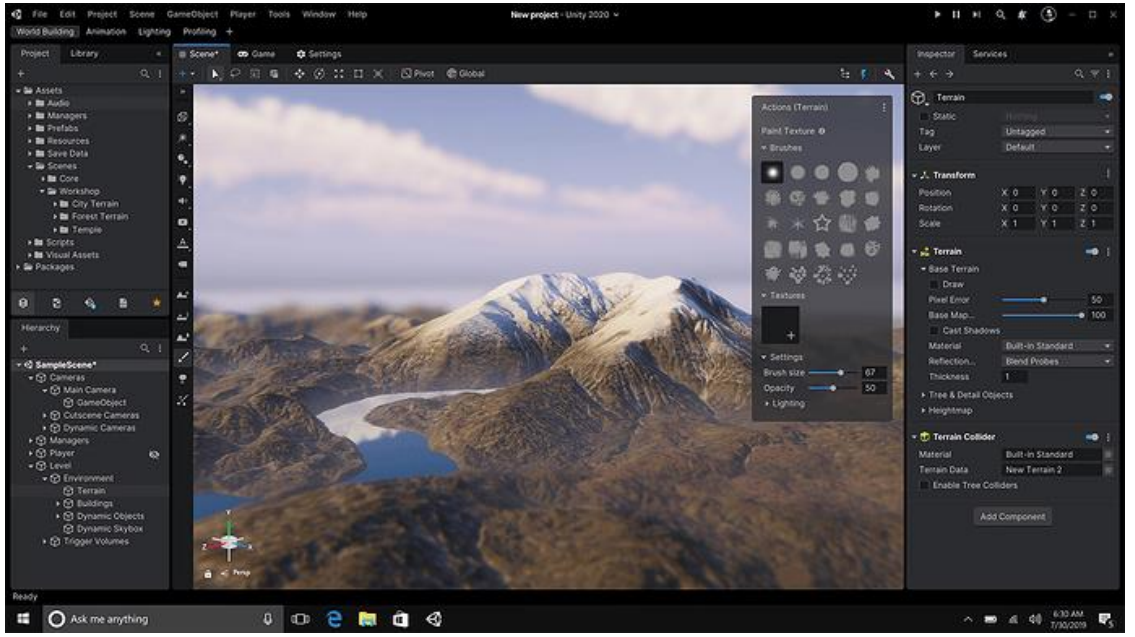
- Όλα αυτά που χρειάζονται για ένα παιχνίδι
- Αυτό το επίπεδο θα μπορούσε να θεωρηθεί εκτός της ίδιας της μηχανής παιχνιδιού

Δημιουργία Ψηφιακού Περιεχομένου
(Άλλο μάθημα για κάθε τύπο…)

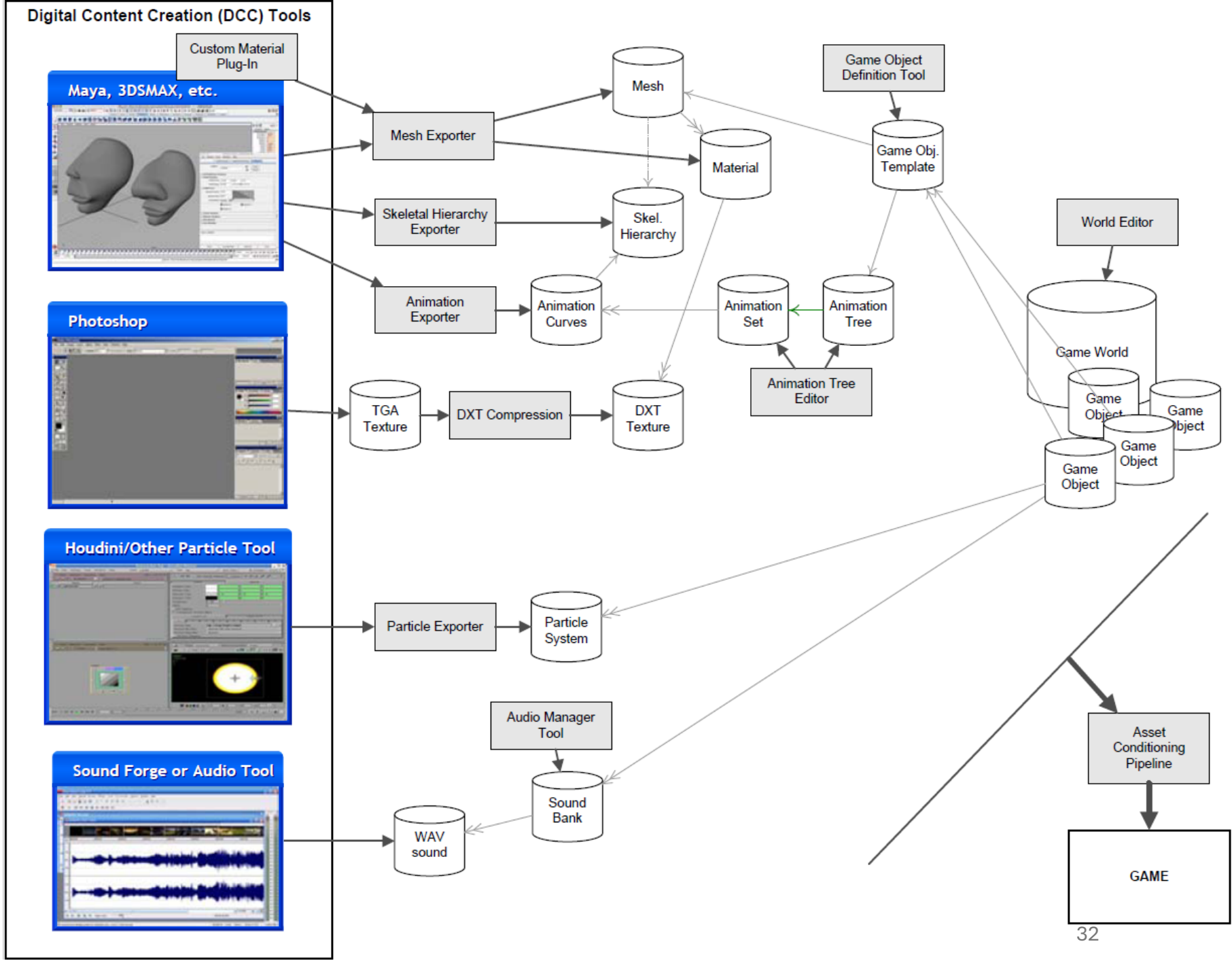
- Οι μηχανές παιχνιδιών ασχολούνται με δεδομένα με πολλές μορφές
- Τα δεδομένα πρέπει να δημιουργηθούν με κάποιο τρόπο
  - 3D Πλέγματα
  - Υφές
  - Ήχος
  - Κίνηση
- Συχνά δημιουργούνται χρησιμοποιώντας εξωτερικά εργαλεία
  - Maya/3ds Max
  - Blender
  - Photoshop
  - SoundForge

Διαχείριση Κόσμου Παιχνιδιού

- Συνήθως ενσωματώνεται στη μηχανή
- Απαραίτητο για να επιτρέπεται στους σχεδιαστές παιχνιδιών να συνεργάζονται με τη μηχανή



Εργαλεία και η Γραμμή Παραγωγής Στοιχείων Παιχνιδιού



Generative Game Engine (GGE)

Από το (όχι και τόσο μακρινό) Μέλλον

GGE: Η Υπόσχεση

- χαμηλότερο κόστος ανάπτυξης παιχνιδιών μέσω αυτοματοποιημένης δημιουργίας περιεχομένου
- πιο εύκολη η δημιουργία παιχνιδιών από μικρές ομάδες
- πραγματικά ανοιχτού κόσμου εμπειρίες με απεριόριστο, δυναμικά δημιουργημένο περιεχόμενο

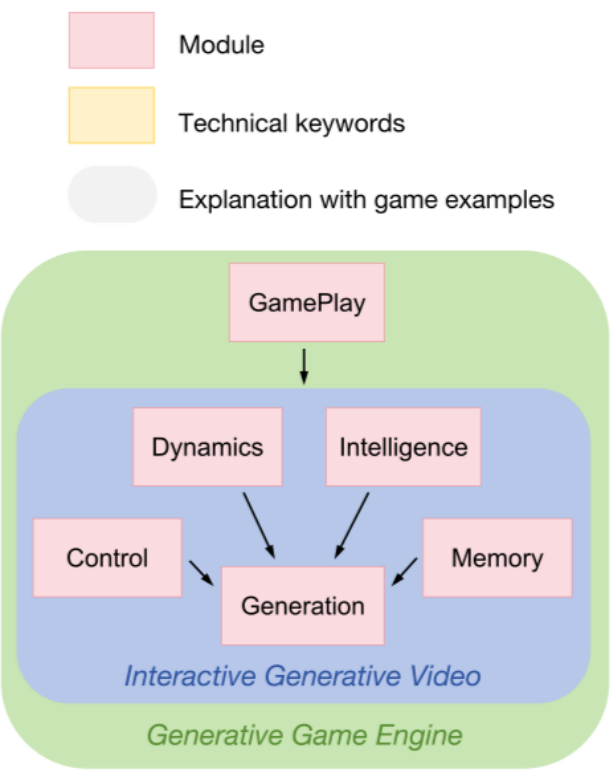
GGE: Τα Πλεονεκτήματα

- Γενικεύσιμες παραγωγικές δυνατότητες (GameFactory)
- Μοντελοποίηση κόσμου με επίγνωση της φυσικής (Nvidia Cosmos, Kling)
- Παραγωγή ελεγχόμενη από τον χρήστη για διαδραστικές εμπειρίες (GameNGen)
- Αξιοποίηση τεράστιας ποσότητας δεδομένων βίντεο για εκπαίδευση

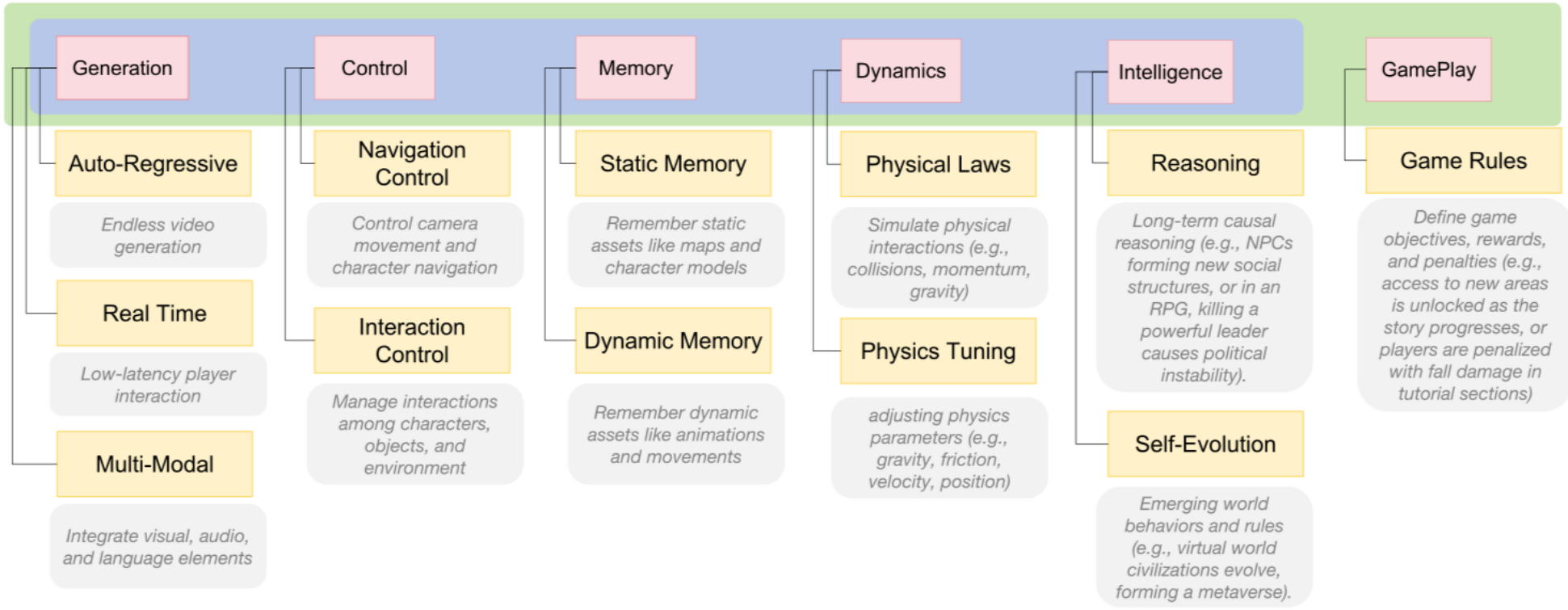
Ένα Παράδειγμα από το Oasis

Μία Άλλη Αρχιτεκτονική

(Προτεινόμενη και όχι Δοκιμασμένη)



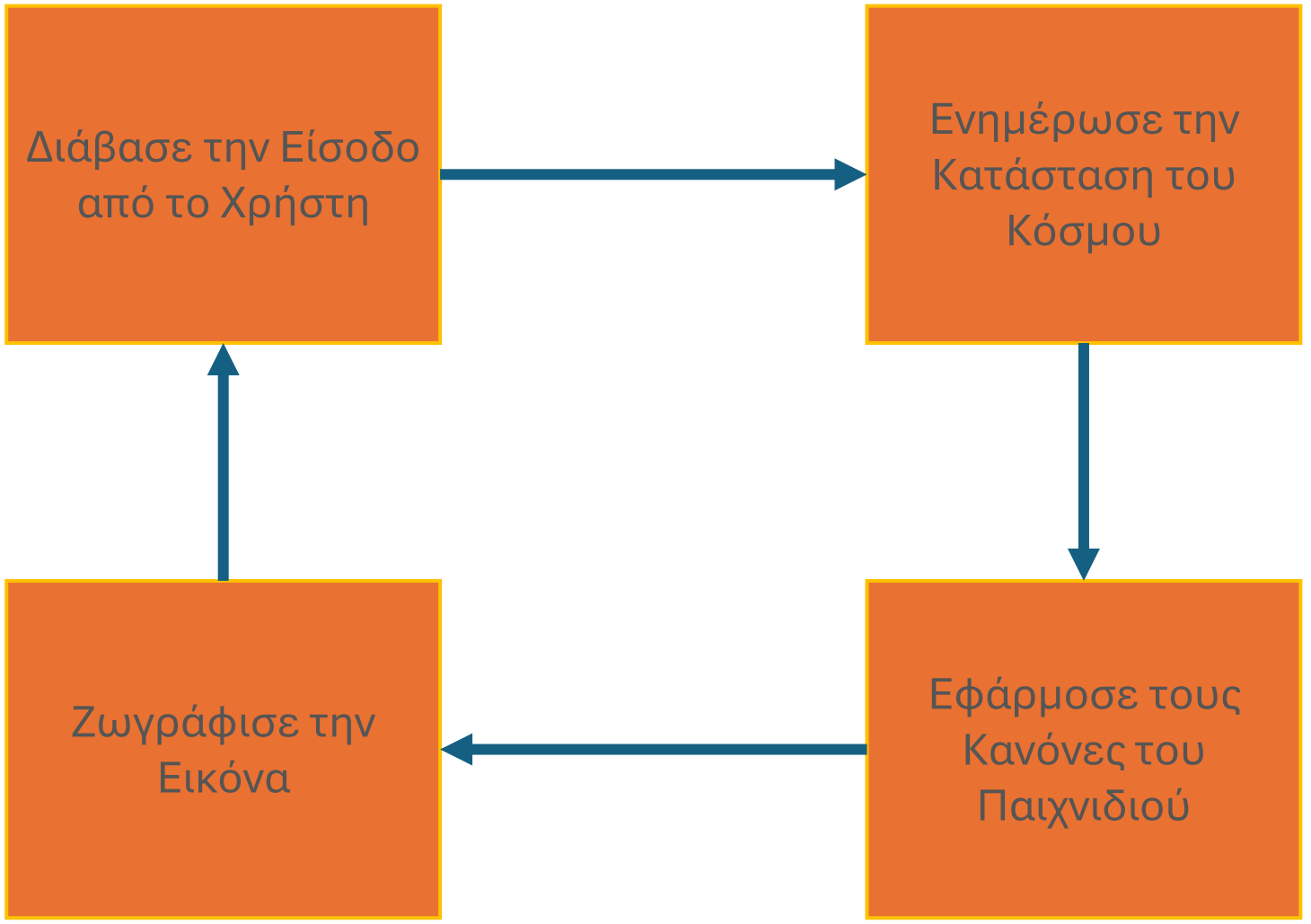
(a) Framework of Generative Game Engine



(b) Technical Keywords of Each Module in Generative Game Engine

Back to Reality ☹:

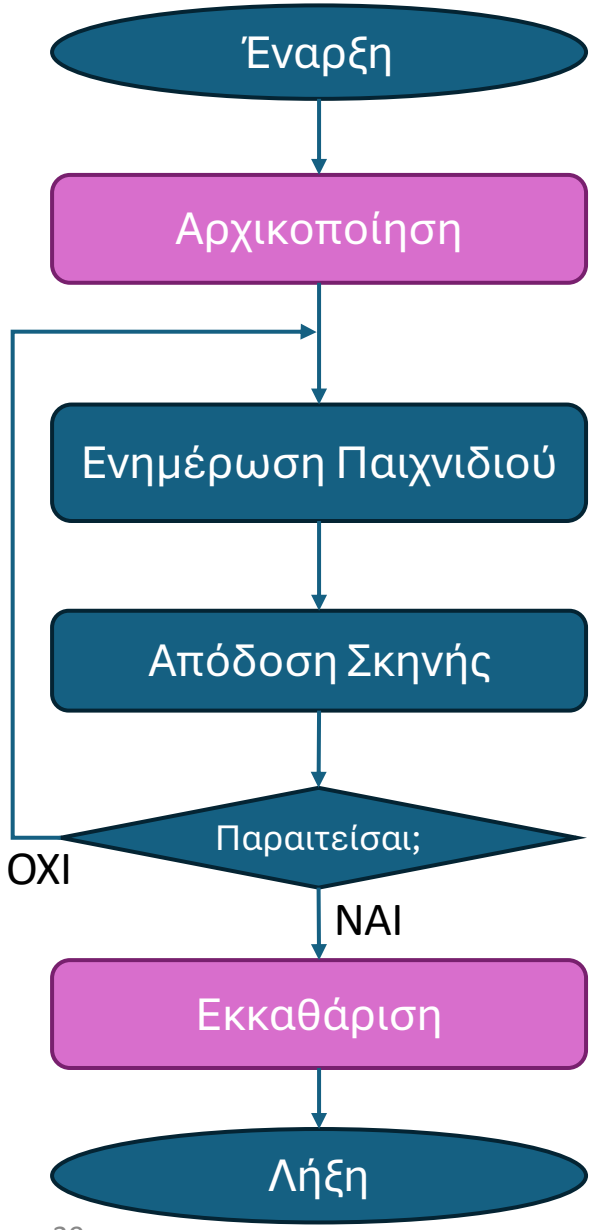
Βρόχος Παιχνιδιού (*Game Loop*)



Βρόχος

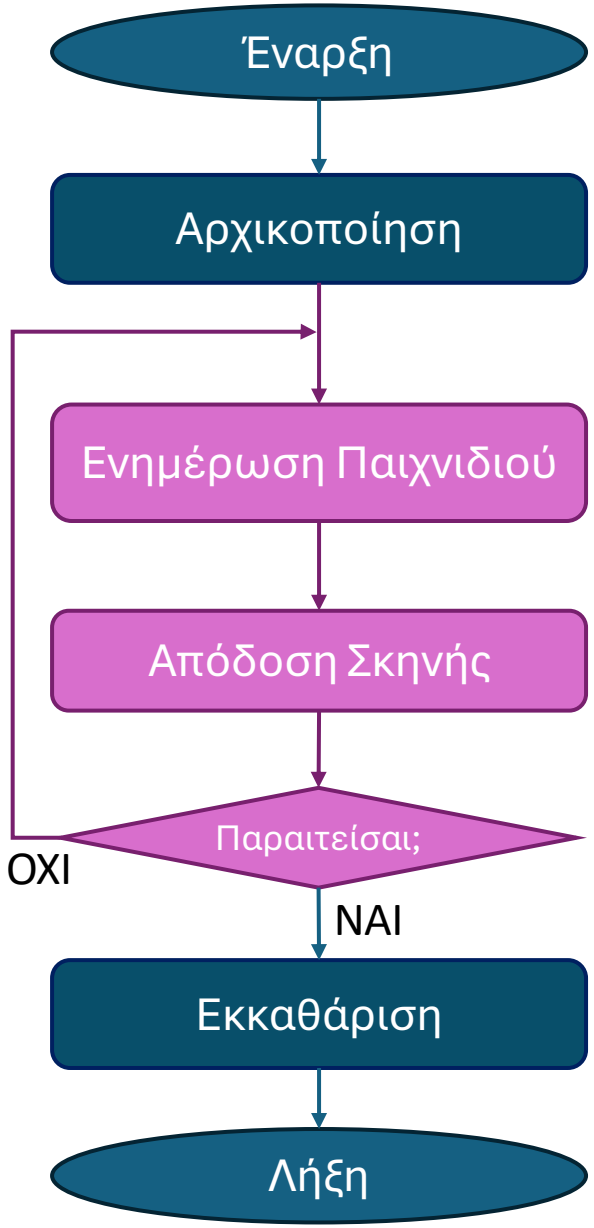
Αρχικοποίηση/Εκκαθάριση

- Το βήμα αρχικοποίησης προετοιμάζει όλα όσα είναι απαραίτητα (π.χ., υφές) για να ξεκινήσει ένα μέρος του παιχνιδιού
- Το βήμα εκκαθάρισης αναιρεί όλα όσα έκανε το βήμα προετοιμασίας, αλλά με αντίστροφη σειρά



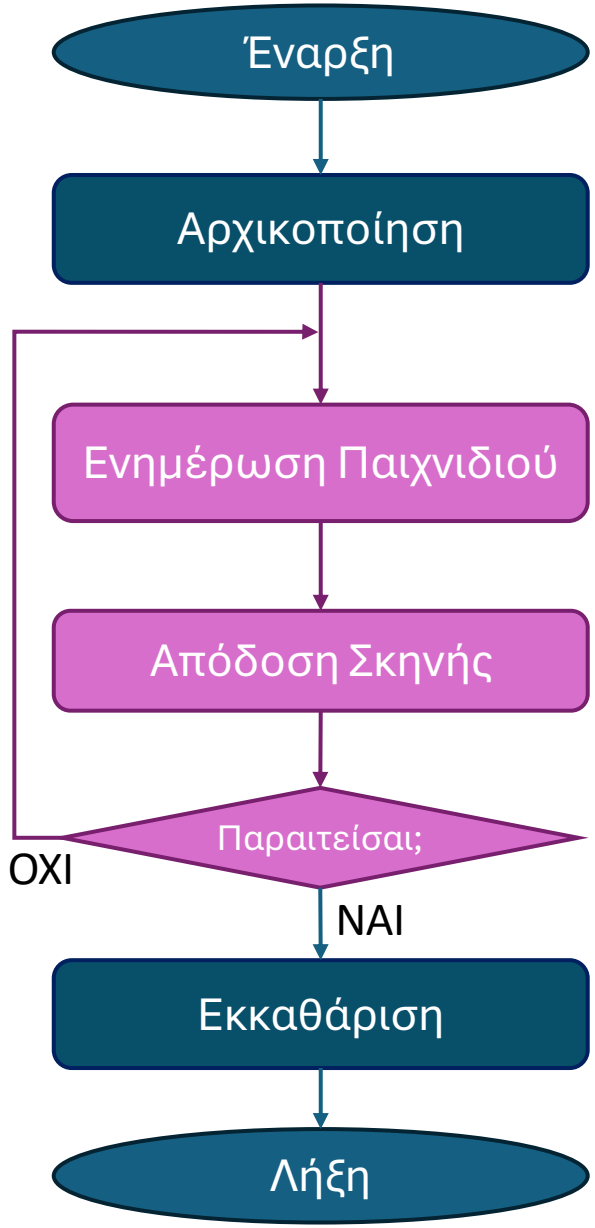
Κύριος Βρόχος

- Τα παιχνίδια οδηγούνται από έναν βρόχο παιχνιδιού που εκτελεί μια σειρά εργασιών σε κάθε frame
- Ορισμένα παιχνίδια έχουν ξεχωριστούς βρόχους για το front end (*Απόδοση Σκηνής*) και για το ίδιο το παιχνίδι
- Άλλα παιχνίδια έχουν ενοποιημένο κύριο βρόχο



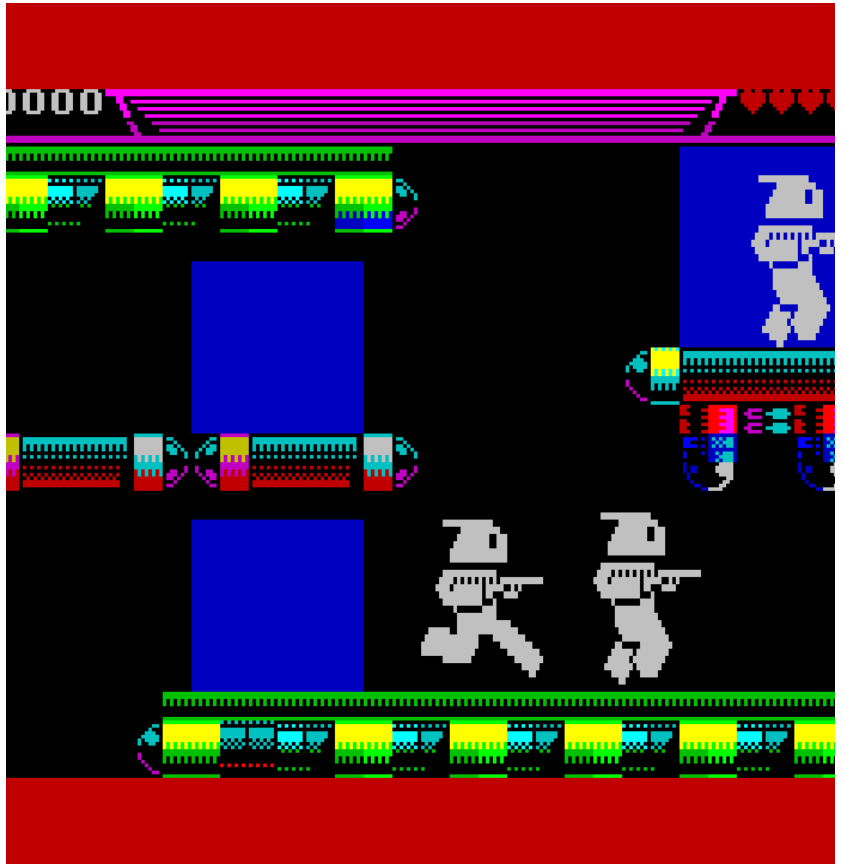
Κύριος Βρόχος: Εργασίες

- Χειρισμός χρόνου
- Διαχείριση εισόδου παίκτη
- Δικτύωση
- Προσομοίωση
- Ανίχνευση σύγκρουσης και απόκριση
- Ενημερώσεις αντικειμένων
- Απόδοση Γραφικών και Ήχου
- Διάφορες άλλες εργασίες

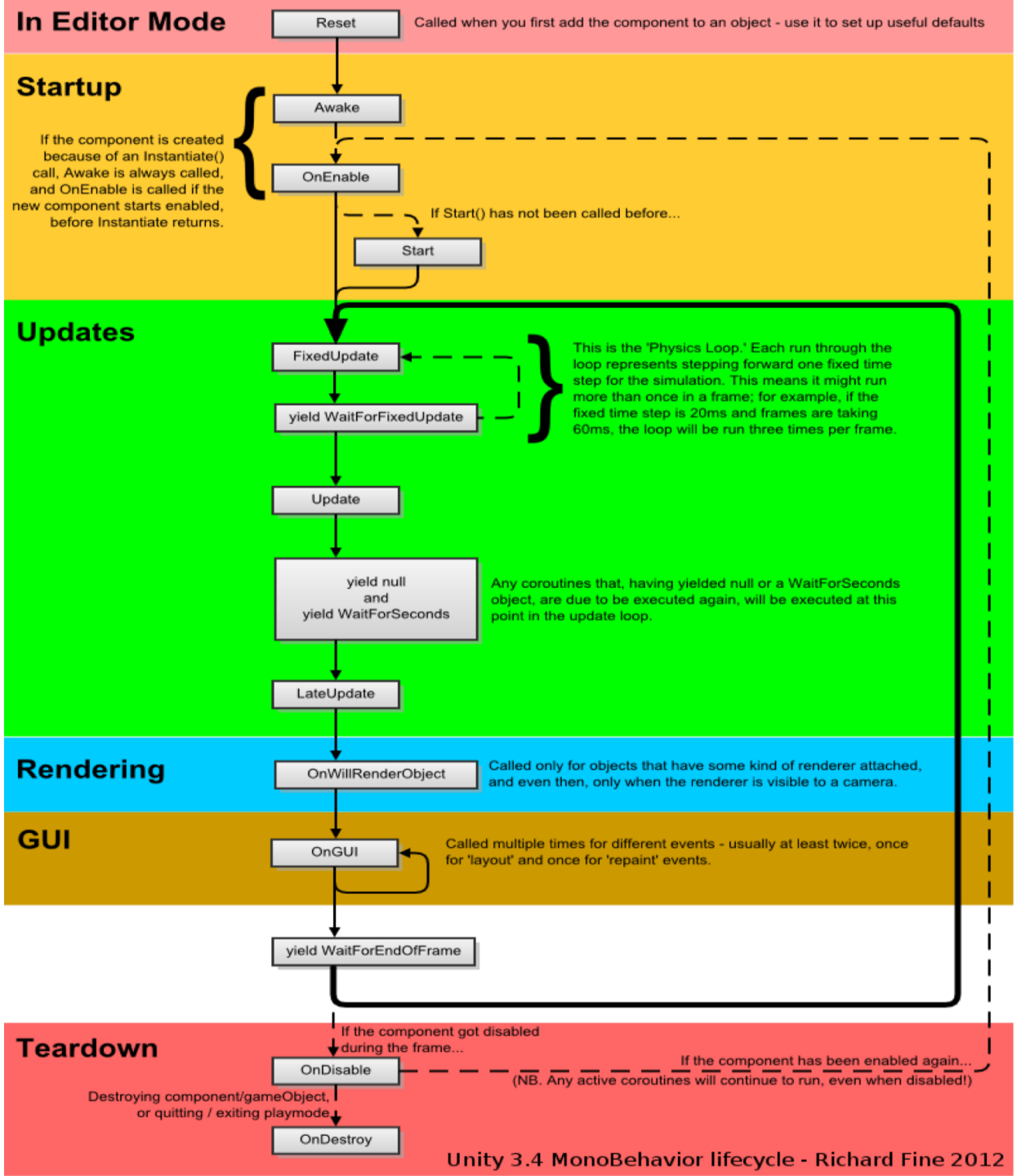


Βρόχος από τις Παλιές Μέρες…

- Παλιά (πολύ παλιά) οι κάρτες γραφικών ήταν πολύ αργές (αν είχες ☺)
- Οι προγραμματιστές βελτιστοποιούσαν την απόδοση χρησιμοποιώντας
  - Εξειδικευμένο υλικό – επέτρεπε την επικάλυψη σταθερού αριθμού από sprites
  - XOR λειτουργία
  - Αντιγράφαν ένα τμήμα του φόντου και αφού σχεδίαζαν το sprite το επανάφεραν το συγκεκριμένο τμήμα
- Σήμερα, όταν η κάμερα κινείται, όλη η σκηνή αλλάζει
  - Όλα στη τρέχουσα σκηνή ακυρώνονται
  - Είναι πιο γρήγορο να ξανασχεδιάσετε τα πάντα παρά να προσπαθήσετε να καταλάβετε τι να ξανασχεδιάσετε



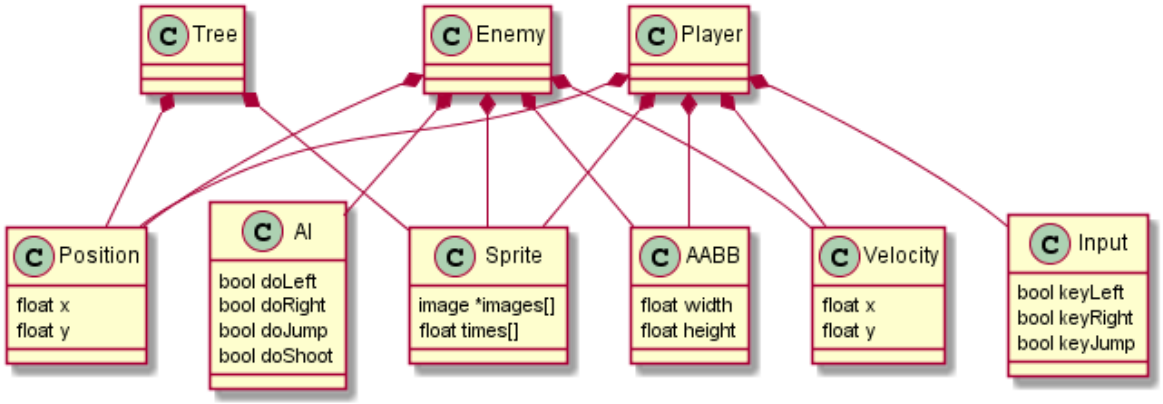
Γενική Μορφή Βρόχου στην Unity



Οντότητες Παιχνιδιού

Ο βρόχος παιχνιδιού επιδρά πάνω στις *οντότητες του παιχνιδιού*

- Γενικά είναι οτιδήποτε στον εικονικό κόσμο του παιχνιδιού με το οποίο ο παίκτης μπορεί να αλληλοεπιδράσει
- Πιο συγκεκριμένα, είναι ένα αυτοτελές κομμάτι διαδραστικού περιεχομένου
- Μόνο αλληλοεπιδρών περιεχόμενο πρέπει να γίνεται οντότητα στο παιχνίδι



Οντότητες – Επικοινωνία

- Η απλούστερη μέθοδος είναι μέσω κλήσεων συναρτήσεων
- Πολλά παιχνίδια χρησιμοποιούν ένα πλήρες σύστημα ανταλλαγής μηνυμάτων
  - Συνήθως ένα μοναδικό αντικείμενο (singleton)
- Πρέπει να είστε προσεκτικοί σχετικά με τη διαχείριση των μηνυμάτων
  - Τα μηνύματα πρέπει να είναι μικρά και με συγκεκριμένα μοτίβα
  - Χιλιάδες μηνύματα ανά frame, παραπάνω υπολογισμός λόγω πολλών κλήσεων σε new() και delete().
  - Αντί για new() και delete κάποιος μπορεί να χρησιμοποιήσει προκαθορισμένες δεξαμενές μνήμης (κάτι σαν ταχυδρομικά κουτιά)

Αναπαράσταση Οντοτήτων

Κληρονομικότητα (κλασσική προσέγγιση): Ένα Ορκ είναι ένας τύπος της κλάσης Εχθρός που είναι ένας τύπος της κλάσης Χαρακτήρας.

+ Εύκολο στη σχεδίαση

– Μπορεί να γίνει αρκετά άτεγκτο:

- Πώς ένας εχθρός μπορεί να είναι και πωλητής όπλων; (diamond problem – πολλαπλή κληρονομικότητα)
- Γιατί μία απλή οντότητα Βράχος να πρέπει να κουβαλά και άχρηστο κώδικα (π.χ., για ζωή);

Σύνθεση (μοντέρνα προσέγγιση): (Entity-Component System – ECS)

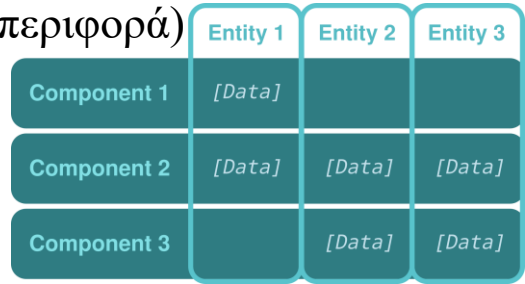
- Οντότητα: Ένα ID (π.χ., αντικείμενο 42)
- Συνιστώσα: Ένα τμήμα δεδομένων (π.χ., Θέση_{Σ}, Φυσική_{Σ}, Ζωή_{Σ}, κτλ.)
- Σύστημα: Η λογική που εφαρμόζεται στις συνιστώσες (το Σύστημα_Φυσικής αναζητά οντότητες που έχουν ως συνιστώσα τη Θέση_{Σ} και τη Φυσική_{Σ})

+ Ευέλικτο (απλά πρόσθεσε/αφαίρεσε μία συνιστώσα για να δώσεις σε ένα αντικείμενο νέα συμπεριφορά)

+ Ιδανικό για πολυπύρηνους CPU

+ Τοπικότητα μνήμης (λόγω των συμπιεσμένων πινάκων)

– Είναι πιο περίπλοκο στην κατανόηση – σχεδίαση

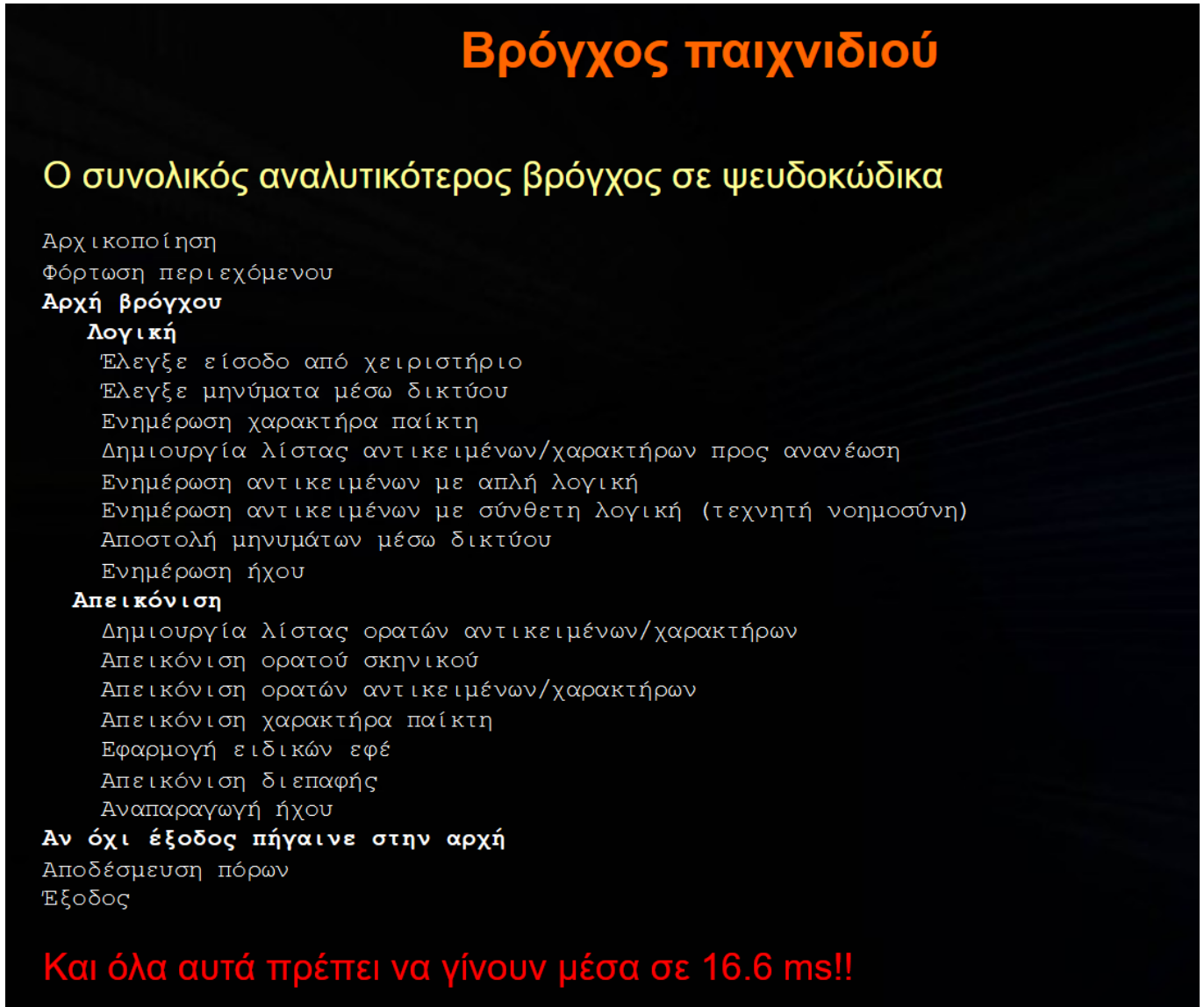


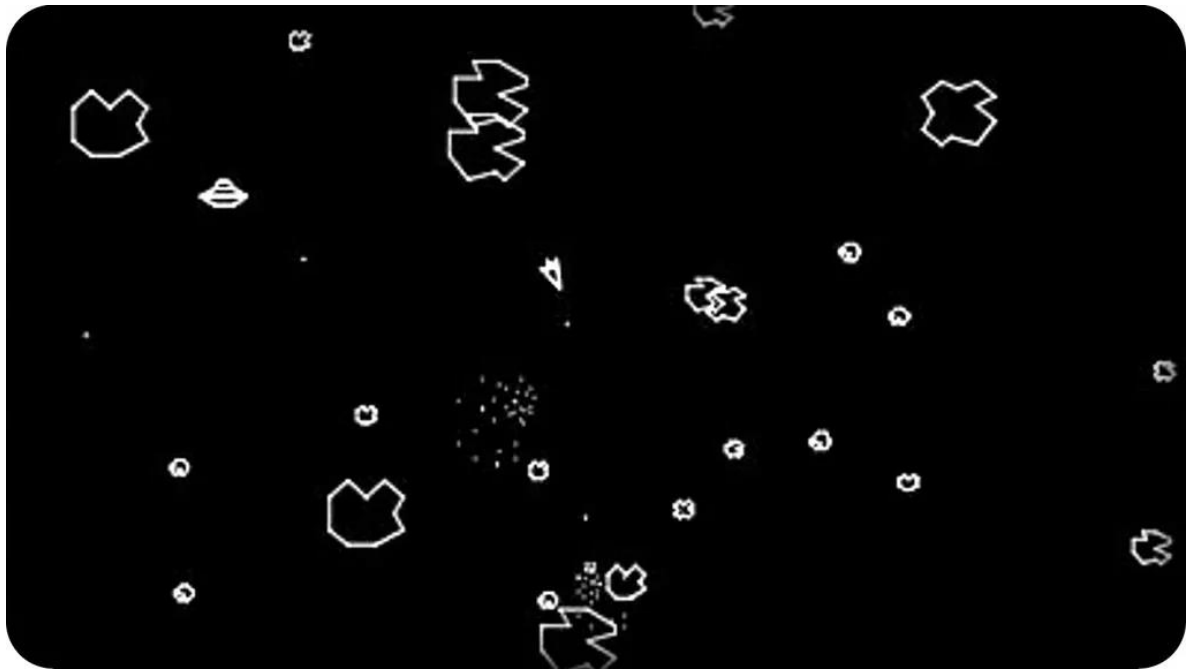
Αποσύνδεση της Απόδοσης Γραφικών

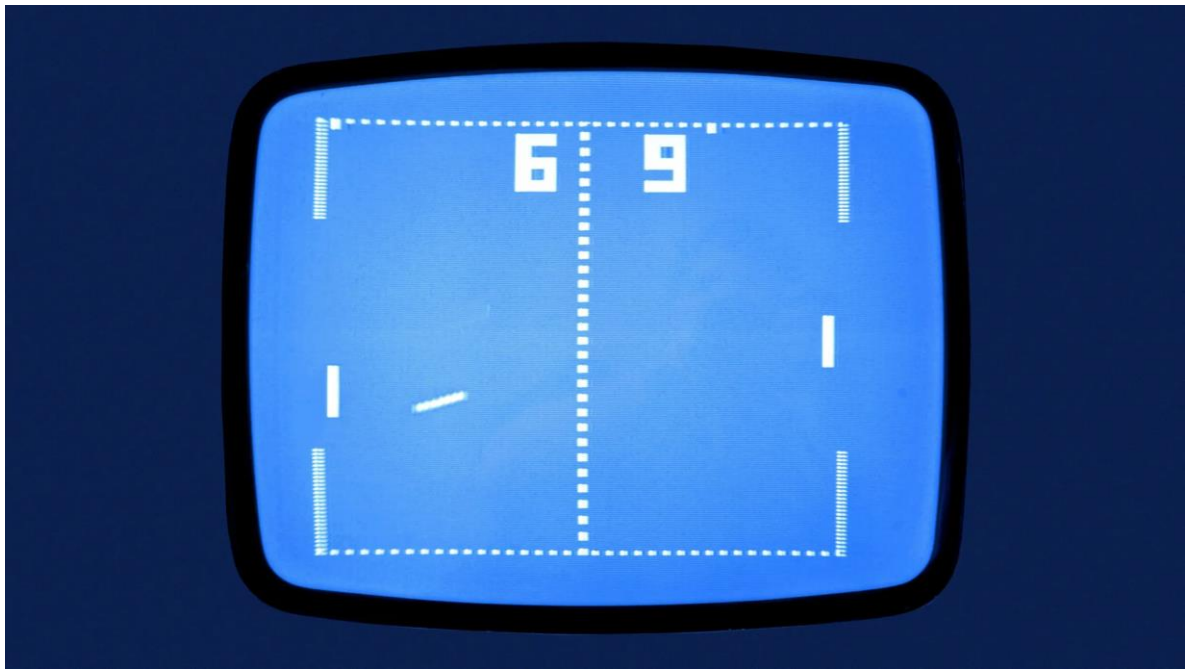
- Μπορεί να αποσυνδέσει το βήμα απόδοσης από τα βήματα προσομοίωσης και ενημέρωσης
  - Βρόχος παιχνιδιού 30 fps
  - Δυνατότητα γραφικών 100fps
- Έχει ως αποτέλεσμα υψηλότερο fps, πιο ομαλή κινούμενη εικόνα και μεγαλύτερη απόκριση
- Η εφαρμογή είναι δύσκολη και μπορεί να είναι επιρρεπής σε σφάλματα

```
while ( !IsDone() ) {

    UpdateTime();

    if(TimetoRunSimulation())
        RunSimulation();

    if(SimulationNotRun())
        InterpolateState();

    RenderWorld();

}
```

Ο Βρόχος πάλι για 60 fps...







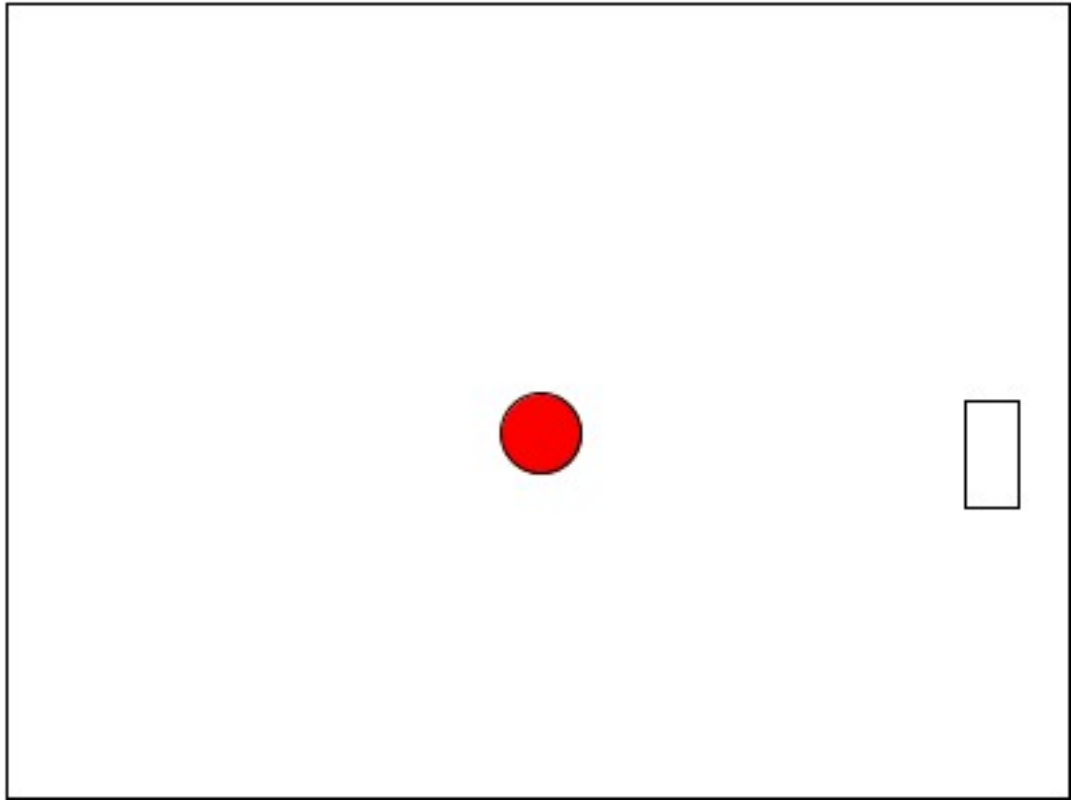
Παραδείγματα Βρόχων

Pong!

Πώς το προγραμματίζουμε;

Βασικός Βρόχος Παιχνιδιού

```
Initialize
do
    update ball (physics)
    update paddle (user input)
    if (collide) do something
    draw stuff
until done
Clean up
```



```
void main(){ //Πιο αναλυτικά
        initGame();
        while(true){
                readHumanInterfaceDevices();
                If (quitButtonPressed())
                        break;
                movePaddles();
                moveBall();
                collideAndBounceBall();
                handleOutOfBounds();
                renderPlayfield();
        }
}
```

Pong!

```
Initialize do
    update ball (φυσική)
    update paddle (είσοδος χρήστη)
    if (collide) do something
    draw stuff
until done
Clean up
```

Απλή φυσική

$$x \mathrel{+}= dx$$
$$y \mathrel{+}= dy$$

Μπορείτε να προσθέσετε και επιτάχυνση:

$$dx \mathrel{+}= ddx$$

Pong!

```
Initialize do
    update ball (φυσική)
    update paddle (είσοδος χρήστη)
    if (collide) do something
    draw stuff
until done
Clean up
```

Περιοδικά έλεγχε τον buffer για είσοδο:

```
if (keyPressed && keyCode == DOWN)
      py = constrain(py+2,0,height);
```

Εναλλακτικά χρησιμοποιώντας events του συστήματος

Pong!

```
Initialize do
    update ball (φυσική)
    update paddle (είσοδος χρήστη)
    if (collide) do something
    draw stuff
until done
Clean up
```

Αν χτυπήσει στο τοίχο ή στην πλατφόρμα τότε κάνε μία ενέργεια

```
if (pong.hitLeft()) {
  pong.reverseX();
  }
```

Pong!

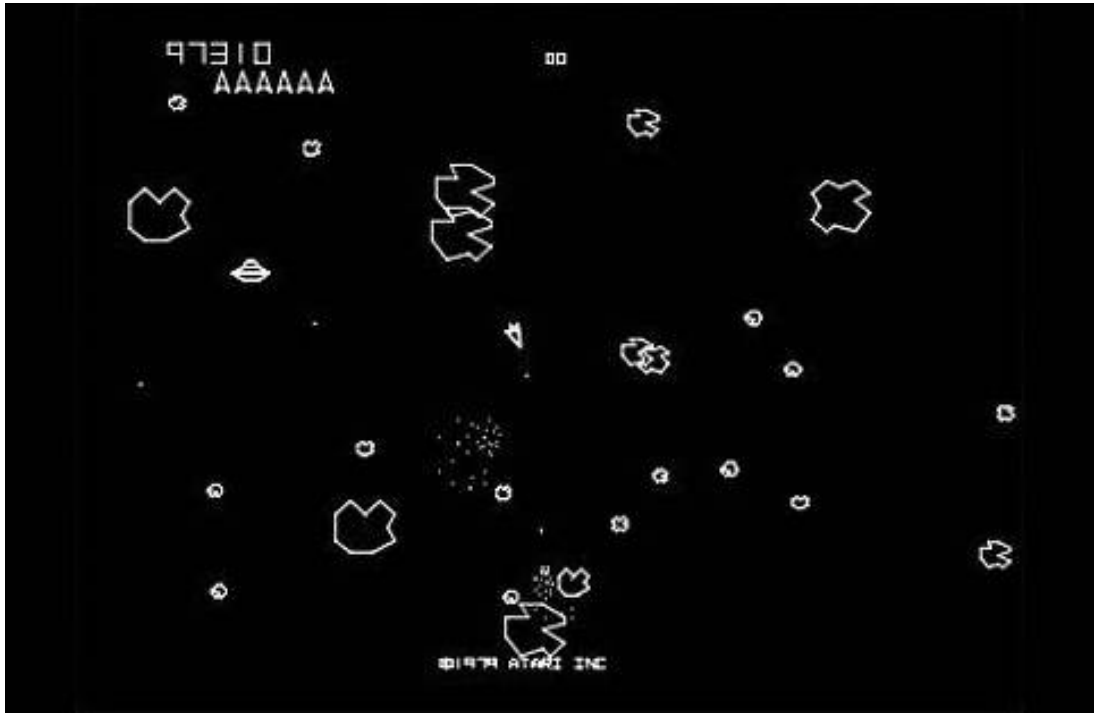
```
Initialize do
    update ball (φυσική)
    update paddle (είσοδος χρήστη)
    if (collide) do something
    draw stuff
until done
Clean up
```

Ζωγράφισε το επίπεδο, την μπάλα και την πλατφόρμα

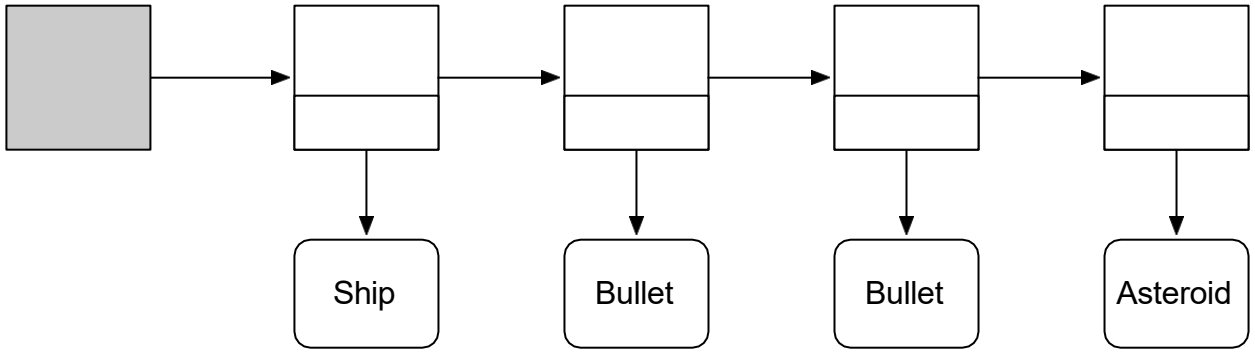
```
// draw ball
color c = color(255,0,0); // red(RGB)
fill(c);
ellipse(x,y,radius,radius);
```

Asteroids!

- Περισσότερα αντικείμενα
  - Διαστημόπλοια
  - Σφαίρες
  - Αστεροειδείς
  - Εχθρικά διαστημόπλοια
  - GUI: Σκορ, ζωές



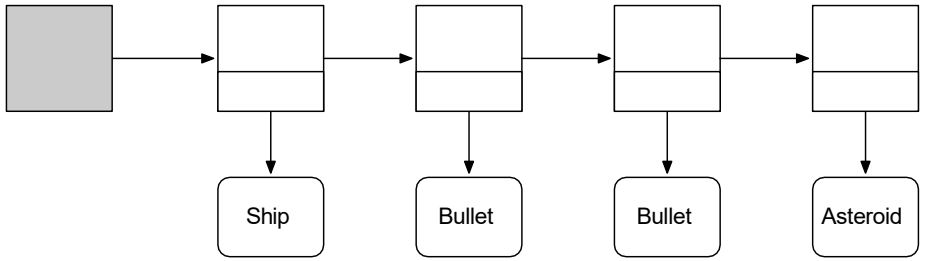
- Λίστα από οντότητες
- Βρόχος:
  - Ενημέρωση όλων
  - Αλληλεπιδράσεις (υπολογιστικά ακριβό)
  - Γραφική απόδοση όλων



Γράφημα Σκηνής έναντι Λίστας Αντικειμένων

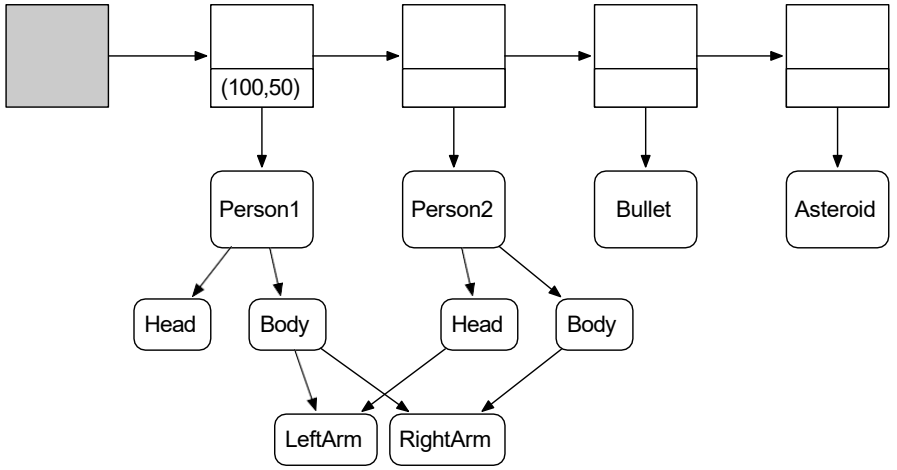
Λίστα Αντικειμένων στο Asteroids

- Όλα τα αντικείμενα είναι απλά, καμία αρθρωτή κίνηση



Γράφημα Σκηνής

- Κατευθυνόμενο γράφημα, σύνθετα αντικείμενα
- Τμήματα μπορεί να διαμοιράζονται



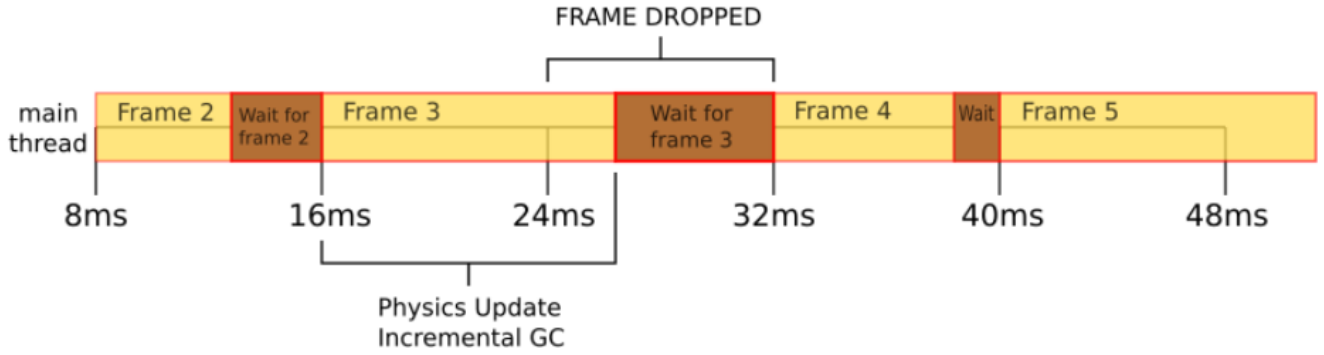
Χρόνος

Χρόνος

- Χρόνος Frame (όχι σταθερός)
  - Εκτέλεση εργασιών σε κάθε frame
  - Το πιο σημαντικό είναι η απόδοση της σκηνής

- Χρόνος προσομοίωσης φυσικής
  - Βήματα στην προσομοίωση φυσικής
  - Μπορεί να τρέχει γρηγορότερα από το χρόνο frame για να γίνει σωστά η φυσική (αποφύγετε μεγάλα βήματα)

- Πραγματικός χρόνος
  - Ρολόι συστήματος
  - Για συγχρονισμό μουσικής, βίντεο και άλλων που χρειάζονται πραγματικό χρόνο



Ρυθμός Απόδοσης έναντι Ρυθμού Ενημέρωσης

- Η απόδοση των γραφικών πρέπει να γίνεται όσο πιο γρήγορα (υψηλότερα fps)

vs

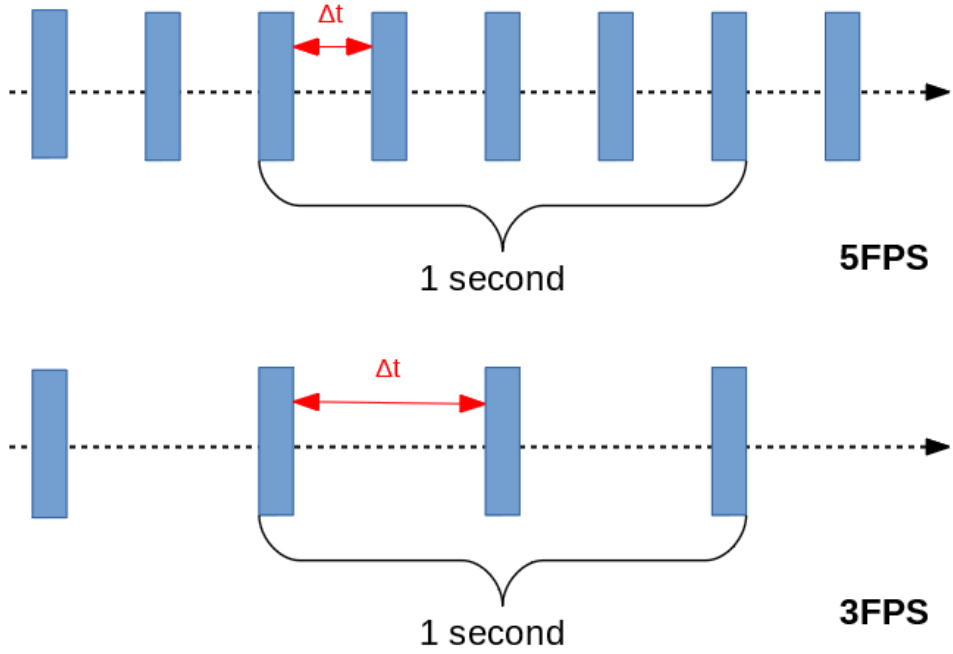
- Η ενημέρωση της κατάστασης του παιχνιδιού πρέπει να γίνεται με σταθερό ρυθμό

Σύγχρονη Λύση:

Αποσύνδεση των δύο ρυθμών

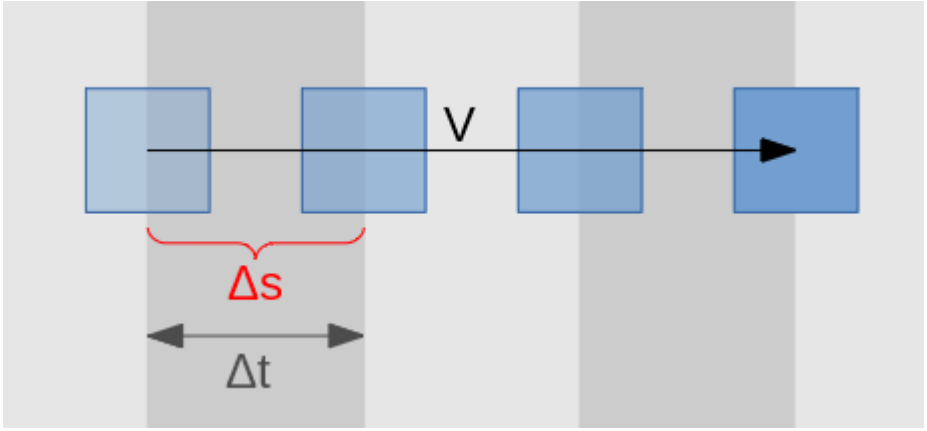
Παλιά…

- Τα πρώτα παιχνίδια δεν μετρούσαν τον πραγματικό χρόνο που πέρασε
- Τα αντικείμενα του παιχνιδιού μετακινούνταν κατά μία σταθερή ποσότητα ανά επανάληψη
  - Ο ρυθμός κίνησης των αντικειμένων εξαρτιόταν από την ταχύτητα της CPU



- Όταν αναβαθμίζατε τον υπολογιστή χάλαγαν όλα επειδή το παιχνίδι έτρεχε πολύ γρήγορα
- Το κουμπί «turbo» χρησιμοποιήθηκε για την επίλυση αυτού του προβλήματος σε ορισμένες περιπτώσεις.

$$\Delta s = v \cdot \Delta t$$

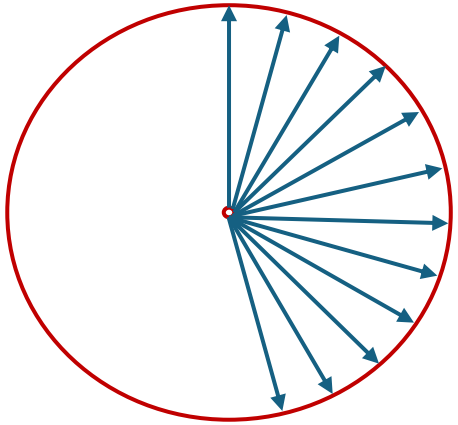


$$\Delta t = deltaTime$$

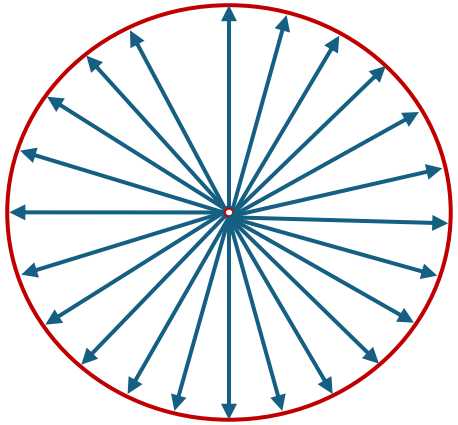
deltaTime

30 fps 1 2 3 4 5 6

60 fps 1 2 3 4 5 6 7 8 9 10 11



30 fps



60 fps

```
using UnityEngine;
// Rotate around the z axis at a constant speed
public class ConstantRotation : MonoBehaviour
{
    public float degreesPerSecond = 2.0f;
    void Update()
    {
        transform.Rotate(0, 0, degreesPerSecond * Time.deltaTime);
    }
}
```

Παραδείγματα
2003 NFS Underground

Η φυσική του παιχνιδιού εξαρτάται από τη ταχύτητα της CPU

Far Cry 2 (2008)

- Η φυσική ή το AI δεν κλιμακώνονται σωστά με τα FPS

Dark Souls 2: Scholar of the First Sin (2015)

Ανθεκτικότητα όπλων

Πείραμα:

- Cardinal Tower Bonfire
- Soldier Hollow outside the door (can kill in one hit)
- Puzzling Stone Sword - 60 durability and 1H-R1 can easily hit corpses
- Kill hollow (1 hit), and then slash the corpse 15 times, correcting my positioning each hit compensate for the forward step
- Record durability of Puzzling Stone Sword, rest at bonfire
- Repeat 3 more times for repeatability

60 FPS	30 FPS
31	45
27	44
25	44
23	43

Riposte

Πείραμα:

- Rapier
- Drakeblood Knights (high HP, easy to parry)
- Parry, riposte, record damage, repeat until dead

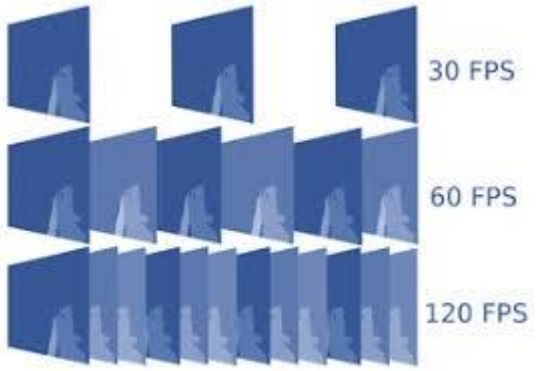
60 FPS	30 FPS
1077	718
1077	718
dead	718

Πώς υπολογίζεται το *deltaTime*;

- Μια μέθοδος που χρησιμοποιείται συχνά είναι η απευθείας ανάγνωση του χρόνου και ο υπολογισμός του deltaTime
- Έχει κάποια προβλήματα
  - Χρησιμοποιούμε τον χρόνο αυτού του frame ως εκτίμηση του χρόνου των επόμενων frame
  - Μπορεί να οδηγήσει σε διαδοχικές καθυστερήσεις και αστάθεια
- Θα μπορούσαμε να χρησιμοποιήσουμε έναν τρέχοντα μέσο όρο
  - Λειαίνει λίγο τα πράγματα
  - Οι μεγάλοι μέσοι όροι εξομαλύνουν το χρόνο, αλλά προσαρμόζονται πιο αργά

Ρύθμιση του Ρυθμού
(*Καρφώνοντας τα FPS*)

- Είναι καλύτερο να ρυθμίσετε τον ρυθμό περιμένοντας μεταξύ των frame
- Πρέπει τα fps να είναι αρκετά σταθερά
- Συνέπεια
- Πιο εύκολο να κάνετε μια λειτουργία εγγραφής και αναπαραγωγής



Unity vs Unreal vs Godot

- Update(): τρέχει μία φορά κάθε frame
- FixedUpdate(): τρέχει με σταθερό ρυθμό (συνήθως ότι έχει σχέση με φυσική το βάζουμε εδώ – rigidbody)

- Tick(): τρέχει μία φορά κάθε frame
- Φυσική: τρέχει με το tick αλλά υποδιαιρεί (substepping) αν το deltatime είναι μεγάλο
- Φυσική: εναλλακτικά async physics (σε άλλο thread για σταθερό ρυθμό)

- _process(): τρέχει μία φορά κάθε frame
- _physics_process: τρέχει με σταθερό ρυθμό (συνήθως ότι έχει σχέση με φυσική το βάζουμε εδώ – rigidbody)