

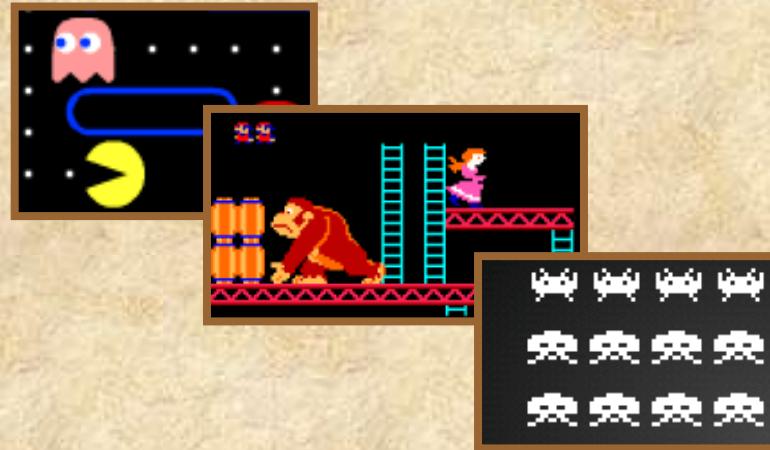


HY454 : ΑΝΑΠΤΥΞΗ ΕΞΥΠΝΩΝ ΔΙΕΠΑΦΩΝ ΚΑΙ ΠΑΙΧΝΙΔΙΩΝ

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ,
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ,
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ**



**ΔΙΔΑΣΚΩΝ
Αντώνιος Σαββίδης**



ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΩΝ Διάλεξη 2η



Περιεχόμενα

■ ***Κατηγορίες παιχνιδιών***

- Τι αντιμετωπίζουμε στην ανάπτυξη
- Η βιομηχανία παιχνιδιών
- Βασικές αρχιτεκτονικές έννοιες
- To game loop
- Χάρτης μαθήματος

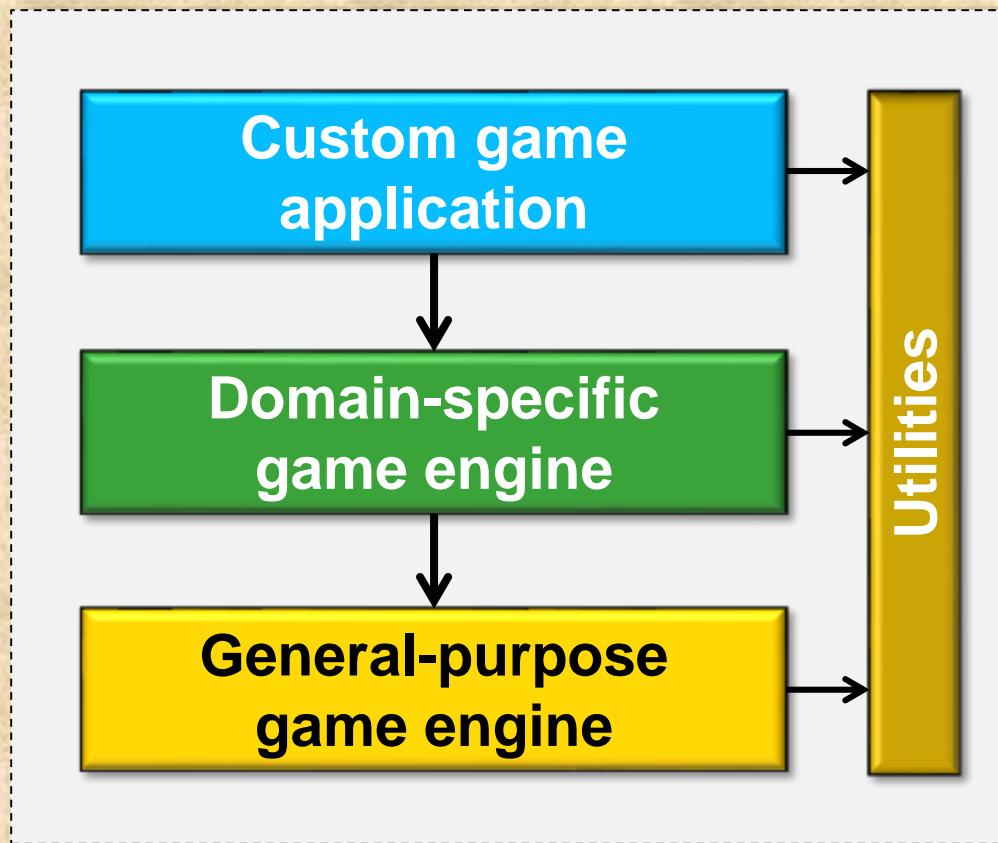


Κατηγορίες παιχνιδιών (1/4)

- Υπάρχουν **αρκετές κατηγορίες παιχνιδιών** όπου κάθε μία παρουσιάζει και διαφορετικές κατασκευαστικές προκλήσεις.
- Σήμερα η **κάθε κατηγορία αντιστοιχεί σε ξεχωριστή μέθοδο ανάπτυξης**. Συνήθως δεν συναντάμε κατασκευαστές οι οποίοι δραστηριοποιούνται σε πλέον της μίας κατηγορίας παιχνιδιών λόγω κόστους.
- Δεν μας απασχολεί να δώσουμε μία εξαντλητική ανάλυση όσο να σκιαγραφήσουμε τις κυριότερες κατηγορίες. Η βάση της ταξινόμησης είναι **το «θέμα του παιχνιδιού»** και **όχι τόσο ο «τρόπος παρουσίασης»** (π.χ. 2D, 3D, 2D1/2) **ή το «είδος του κόσμου του παιχνιδιού»** (ύπαιθρος, κτίριο, διάστημα, αέρας, νερό).



Κατηγορίες παιχνιδιών (2/4)



TYPICAL MACROSCOPIC SPLIT OF THE PRIMARY SUBSYSTEMS IN CURRENT GAMES



Κατηγορίες παιχνιδιών (3/4)

- Adventures, role-playing games
 - Σενάριο, πλοκή, γρίφοι, στόχος, μονοπάτια, βοηθητικά κρυμμένα αντικείμενα, ανατροπές δράσης
- Strategy games
 - Τακτική, AI, διαχείριση πολλών πόρων, μεγιστοποίηση κέρδους, τακτικοί αντίπαλοι, πρόβλεψη
- Beat them up / fighting / action games
 - Αντίπαλοι, έντονη δράση, κίνηση στο χώρο, ταχύτητα, βία, εξόντωση αντιπάλων, ρεαλισμός, ειδικά εφέ
- First / third person shooters
 - Χαρακτήρας, αποστολή «εξόντωσης», εξουδετέρωση εμποδίων, ειδικά όπλα, διάφορα κλειδιά, ρεαλισμός κίνησης και δράσης, περιστασιακοί αντίπαλοι



Κατηγορίες παιχνιδιών (4/4)

- Maze games
 - Λαβύρινθοι, προσανατολισμός, παγίδες, βοηθήματα, μέθοδος επίλυσης
- Sports games
 - Κανόνες, ρεαλισμός κίνησης, αθλήματα, ομάδες, επίδοση, ανταγωνισμός, κοινό, εξομοίωση ανθρώπινων κινήσεων
- Racing games
 - Οχήματα, πίστα, εμπόδια, αντίπαλοι, χρόνος, ταχύτητα, συγκρούσεις, κανόνες φυσικής, εξομοίωση μηχανικών κινήσεων
- World simulations
 - Πλάσματα, AI, κόσμοι, ένστικτα / συμπεριφορές, εξέλιξη, τελικός στόχος, μετάβαση καταστάσεων, κοινωνικοί κανόνες



Περιεχόμενα

- Κατηγορίες παιχνιδιών
- ***Tι αντιμετωπίζουμε στην ανάπτυξη***
- Η βιομηχανία παιχνιδιών
- Βασικές αρχιτεκτονικές έννοιες
- To game loop
- Χάρτης μαθήματος



Τι αντιμετωπίζουμε στην ανάπτυξη (1/5)

■ Μεγάλο μέσο κόστος παραγωγής

- Δεκάδες εκ. Ευρώ για την παραγωγή ενός AAA εμπορικού τίτλου, με 30-36 μήνες ανάπτυξης και πολύ απαιτητικό χρονοδιάγραμμα
- Πολλές περιπτώσεις παιχνιδιών που τελικά κόστισαν πολύ περισσότερο του προβλεπόμενου τόσο σε κόστος όσο και σε χρόνο
 - ◆ π.χ. το παλαιότερο πολυδιαφημισμένο πριν βγει στην αγορά παιχνίδι Messiah της Shiny Entertainment
 - ◆ To Assassin's Creed της Ubisoft χρειάστηκε τέσσερα χρόνια με 150 άτομα συνολικά στο production / development team
 - ~18 εκ ευρώ κόστος, αλλά πούλησε 8 εκ copies (περίπου 240 εκ ευρώ)
- καθώς και παιχνίδια στα οποία συμμετείχαν εξαίρετοι προγραμματιστές αλλά αυτά ποτέ δεν τελείωσαν και η εταιρεία τελικά έκλεισε
 - ◆ π.χ. το Golgotha της Crack dot com, στην οποία συμμετείχαν προγραμματιστές της ομάδας του Doom από την id Software.



Τι αντιμετωπίζουμε στην ανάπτυξη (2/5)

Πολλές διαφορετικές πλατφόρμες παιχνιδιών (1/2)

- Η δυνατότητα να κατασκευάσει κάποιος ένα παιχνίδι από την αρχή για όλες τις διαφορετικές πλατφόρμες, αλλά με κοινό το μεγαλύτερο τμήμα κώδικα είναι σήμερα ουτοπία
 - συνήθως επιλέγει έναν βασικό τύπο πλατφόρμας (πχ. PCs, consoles, smart phones)
- Η ανάπτυξη γίνεται παράλληλα για κάθε πλατφόρμα και σχεδόν πάντα υπάρχει μεγάλο πρόβλημα συγχρονισμού
- Το πιο συνηθισμένο: heavy version (PCs and consoles), light version (phones, tablets, TVs)



Τι αντιμετωπίζουμε στην ανάπτυξη (3/5)

Πολλές διαφορετικές πλατφόρμες παιχνιδιών (2/2)

- Ταχύτητα της GPU (Open GL OpenGL ES)
- Shared memory, thread programming, multitasking
- Χώρος αποθήκευσης σε δίσκο και τρόπος πρόσβασης
- Διαθέσιμη κεντρική μνήμη
- Κανόνες αποθήκευσης γραφικών δεδομένων σε συγκεκριμένα τμήματα μνήμης (bitmaps, audio, textures, κλπ)
- Ειδικές συσκευές εισόδου / εξόδου (ξεχάστε το απλό mouse και keyboard) – console peripherals, **touch** phones
- Περιορισμένη ταχύτητα (desktops vs phones)
- Διαφορετικά περιβάλλοντα ανάπτυξης και libraries



Τι αντιμετωπίζουμε στην ανάπτυξη (4/5)

■ Ιδιαίτερα μεγάλες απαιτήσει σε *testing*

- Συνήθως δεν είναι εφικτός ή ρεαλιστικός ο ενδελεχής έλεγχος των λειτουργιών και τμημάτων σε απομόνωση – unit testing
- τα data και γενικά το state ενός τμήματος είναι ασύμφορα πολυπληθή ώστε να δοκιμάσουμε να τα «φέρουμε» στις εναλλακτικές καταστάσεις απλά με testing statements (π.χ. ο animation manager που χειρίζεται την κινηματική πλέον των 200 αντικειμένων / χαρακτήρων του παιχνιδιού)
- οι διάφορες εναλλακτικές καταστάσεις που φυσιολογικά αναμένονται κατά την εκτέλεση συνιστούν διαφορετικά σενάρια ελέγχου, άρα δεν μπορούμε να γενικεύσουμε μόνο με ένα αντιπροσωπευτικό test unit
- προσλαμβάνονται απαιτητικοί και «μανιώδεις» gamers με σκοπό τόσο να αξιολογήσουν το παιχνίδι όσο και να καταγράψουν τα διάφορα λάθη – κάποιοι από αυτούς γίνονται πολύ γνωστοί και πληρώνονται αρκετά καλά



Τι αντιμετωπίζουμε στην ανάπτυξη (5/5)

- Οι αρχικές φιλοδοξίες για το προϊόν δεν ανταποκρίνονται ποτέ στην τελική πραγματικότητα
 - Πάντα υπάρχουν λειτουργίες και χαρακτηριστικά που ποτέ δεν υφίστανται στο τελικό σύστημα
 - Πάντα υπάρχει ένα μεγάλο ποσοστό των τελικών χαρακτηριστικών που δεν είχαν προβλεφθεί κατά τον ορισμό του συστήματος και εισάγονται δυναμικά
 - Συνήθως το χρονοδιάγραμμα ξεφεύγει με μεγάλη απόκλιση (έως και 30%)
 - Συνήθως το τελικό κόστος είναι μεγαλύτερο από το εκτιμώμενο με μεγάλη απόκλιση (έως και 30%)
 - Συνήθως οριοθετείται η επόμενη έκδοση με την φιλοδοξία να είναι αυτή καλύτερη και πιο επιτυχής
 - Πάντα υπάρχει η πεποίθηση ότι κατασκευάζεται ο πιο εμπορικά επιτυχημένος τίτλος αλλά αυτό αποδεικνύεται αναληθές
 - Συνήθως στην επόμενη έκδοση (sequel) οι μισοί προγραμματιστές δουλεύουν σε άλλη εταιρεία



Περιεχόμενα

- Κατηγορίες παιχνιδιών
- Τι αντιμετωπίζουμε στην ανάπτυξη
- ***Η βιομηχανία παιχνιδιών***
- Βασικές αρχιτεκτονικές έννοιες
- To game loop
- Χάρτης μαθήματος



Η βιομηχανία παιχνιδιών (1/3)

- Μία από τις μεγαλύτερες βιομηχανίες στο χώρο πληροφορικής
 - Το 1994 έφτανε τα 5,8 δις ενώ το 2008 έφτασε τα 38 δις **ξεπερνώντας κατά πολύ τη βιομηχανία κινηματογράφου**
 - ◆ Τα 10 δις αφορούν αποκλειστικά PC games
 - Θεωρείται **το ταχύτερα εξελισσόμενο τμήμα του τομέα μέσων και ψυχαγωγίας** (όπου ανήκει η τηλεόραση και το internet)
 - ◆ Το 2019 έφτασε τα 110 δις
 - Η μόνη κατηγορία λογισμικού με τέτοιο τεράστιο αριθμό διαφορετικών προϊόντων
- Πολύ μεγάλος αριθμός εταιρειών, κάθε χρόνο εμφανίζονται πολλές νέες εταιρείες, κάθε χρόνο κλείνουν πολλές εταιρείες πριν καν παλιώσουν αρκετά
- Ελάχιστες συγκριτικά εταιρείες έχουν το μεγαλύτερο μερίδιο κέρδους
- Η διαχείριση της βιομηχανίας γίνεται από τους μεγάλους εκδοτικούς οίκους (publishers) οι οποίοι συνεργάζονται με συμβόλαια με τις διάφορες εταιρείες παιχνιδιών
 - Electronic Arts, Capcom, Activision, Interplay, Eidos, Buena Vista, Ubisoft, Hasbro, Microsoft, Infogrammes, Midway, Namco, Sega, Epic



Η βιομηχανία παιχνιδιών (2/3)

- Μία από τις μεγαλύτερες κοινότητες προγραμματιστών, με τη μικρότερη ηλικία εισαγωγής στο χώρο
 - συνηθισμένο να εμπλέκονται παιδιά Γυμνασίου και Λυκείου που έπειτα ασχολούνται επαγγελματικά (σχεδόν το 15% των συμμετοχών στο IGF, <http://www.igf.com/>, Independent Games Festival, έρχεται από κατασκευαστές κάτω των 18 ετών)
- Τη μεγαλύτερη ταχύτητα «εξέλιξης» από όλες τις κατηγορίες λογισμικού. Όχι από ερευνητικής άποψης, αλλά ως προς το τελικό αποτέλεσμα.
 - Συνήθως κάθε νέο προϊόν είναι ήδη ξεπερασμένο σε σχέση με τις προδιαγραφές των τίτλων που εκείνη τη στιγμή βρίσκονται σε ανάπτυξη
- Μεγάλη έμφαση στην ταχύτητα εκτέλεσης, στον πλούτο περιεχομένων (ιδιαίτερα στην λεπτομέρεια και τον ρεαλισμό των αναπαραστάσεων)
 - Τα παιχνίδια τείνουν να φτάνουν σε επίπεδο παραγωγής, σεναρίου, σκηνοθεσίας και ρεαλισμού τον κινηματογράφο
 - Ο κινηματογράφος τείνει να υιοθετήσει τη δράση και σχεδίαση των παιχνιδιών



Η βιομηχανία παιχνιδιών (3/3)

- Αρκετές κερδοφόρες κινηματογραφικές παραγωγές μεταφέρονται και ως παιχνίδια
 - Star wars, James Bond, Spiderman, Jurassic Park, Indiana Jones, Matrix, X-men, κλπ, κλπ
- Ωστόσο, ορισμένα παιχνίδια γίνονται τόσο δημοφιλή που τελικά μετατρέπονται και σε κινηματογραφικές παραγωγές
 - Mortal Kombat, Tomb Raider, Final Fantasy, Street Fighter, Resident Evil, Prince of Persia, Doom
- Παραδόξως, συγκριτικά μικρή ωριμότητα τεχνολογίας λογισμικού και ανάπτυξης σε σχέση με το τεράστιο εκτόπισμα που έχει αυτή η βιομηχανία
 - Είναι ελάχιστοι οι ξεχωριστοί προγραμματιστές και συνήθως αυτοί ηγούνται των ομάδων ανάπτυξης και των βασικών μηχανών
- ➔ **Μία άκρως ανταγωνιστική βιομηχανία με τεράστια αγορά αλλά με κακές συνθήκες εργασίας και πολλές υπερωρίες**



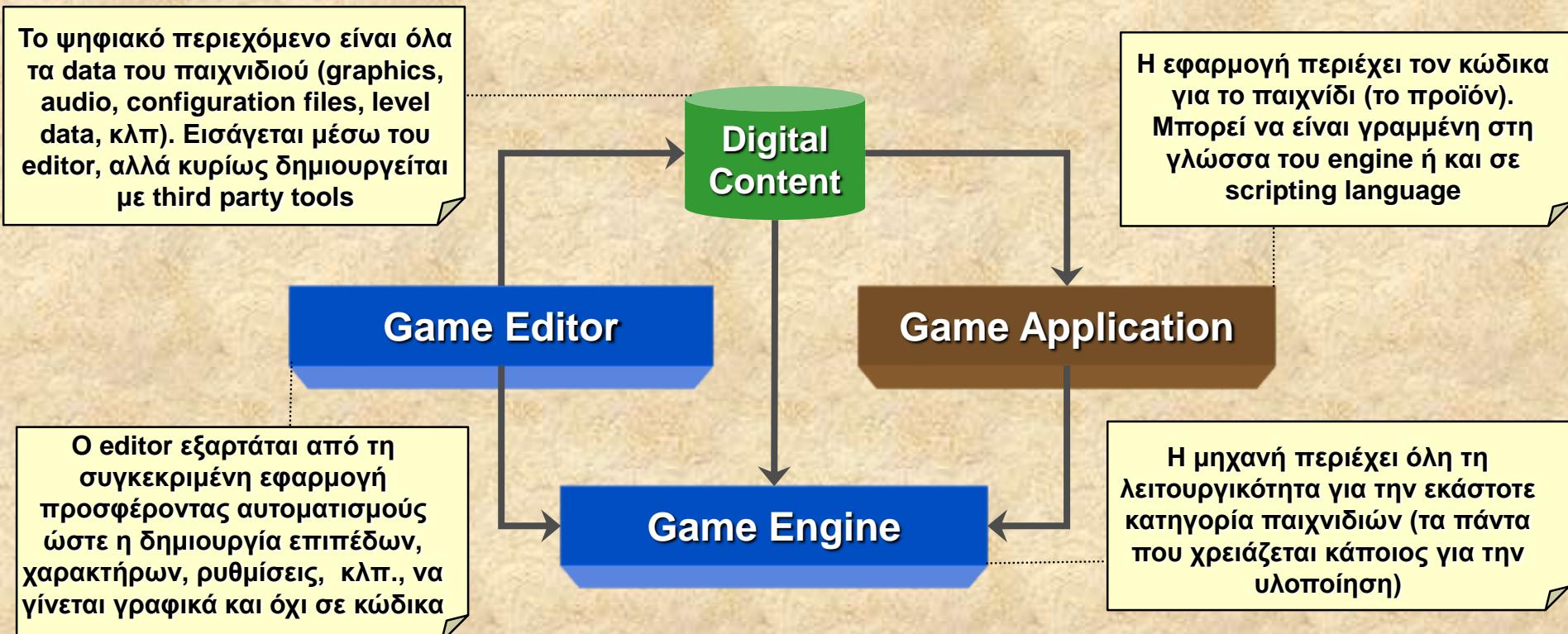
Περιεχόμενα

- Κατηγορίες παιχνιδιών
- Τι αντιμετωπίζουμε στην ανάπτυξη
- Η βιομηχανία παιχνιδιών
- ***Βασικές αρχιτεκτονικές έννοιες***
- To game loop
- Χάρτης μαθήματος



Βασικές αρχιτεκτονικές έννοιες (1/5)

- Η πρώτη διαισθητική ανάλυση ενός video game, χωρίς ακόμη να σκιαγραφείται καμία αρχιτεκτονική, οδηγεί σε τέσσερα στοιχεία:





Βασικές αρχιτεκτονικές έννοιες (2/5)

- Θα μας απασχολήσει κυρίως η κατασκευή του game engine και για μία συγκεκριμένη κατηγορία παιχνιδιών
 - Σκηνογραφία και κανόνες φυσικής
 - Τρόπος δράσης και μέθοδοι παρουσίασης
 - Κόστος περιεχομένου και τρόπος δημιουργίας
- Πολλές από τις τεχνικές των 3D games προήλθαν από τον κινηματογράφο, εξελίχθηκαν από τη βιομηχανία παιχνιδιών και «επέστρεψαν» στον κινηματογράφο για την δημιουργία ψηφιακής δράσης – digital action, γνωστό στους κινηματογραφιστές άπλα ως “*the CGs*” δηλ. computers graphics



Βασικές αρχιτεκτονικές έννοιες (3/5)

- Στην βασική μηχανή εμφανίζεται η ανάγκη οι παρακάτω έννοιες να είναι αρχιτεκτονικά τμήματα με κάποιο τρόπο
 - **Rendering**
 - ◆ Η βασική μηχανή ζωγραφικής που πρέπει να είναι απλή και ταχύτατη, προσφέροντας τα βασικά συστατικά στοιχεία που απαιτούνται για τις «οπτικές» ανάγκες του παιχνιδιού, χρησιμοποιώντας τα graphics data
 - **Terrain**
 - ◆ Ο χώρος στον οποίο διαδραματίζεται το παιχνίδι, η δομή του από γραφικά στοιχεία και η γεωμετρία του, ο τρόπος παρουσίασης με χρήση του rendering engine, ο τρόπος αποθήκευσης, οι κανόνες φυσικής που επιβάλει, η πλούγηση και προοπτική, διαχωρισμός design-time / runtime δεδομένων
 - **(Non Player) Characters - NPCs**
 - ◆ Τα όντα και αντικείμενα του κόσμου του παιχνιδιού, οι κανόνες κίνησης (κινηματική), η συμπεριφορά τους, τα χαρακτηριστικά τους, οι κατηγορίες τους, η (γεωμετρική) σχέση τους με το terrain, AI
 - **Players**
 - ◆ Ο παίκτης (παίκτες), οι δυνατότητές του, οι υποστηριζόμενες ενέργειες, η εμφάνιση και κίνηση, ο τρόπος υποστήριξης πολλών παικτών, η σχέση του με τους characters

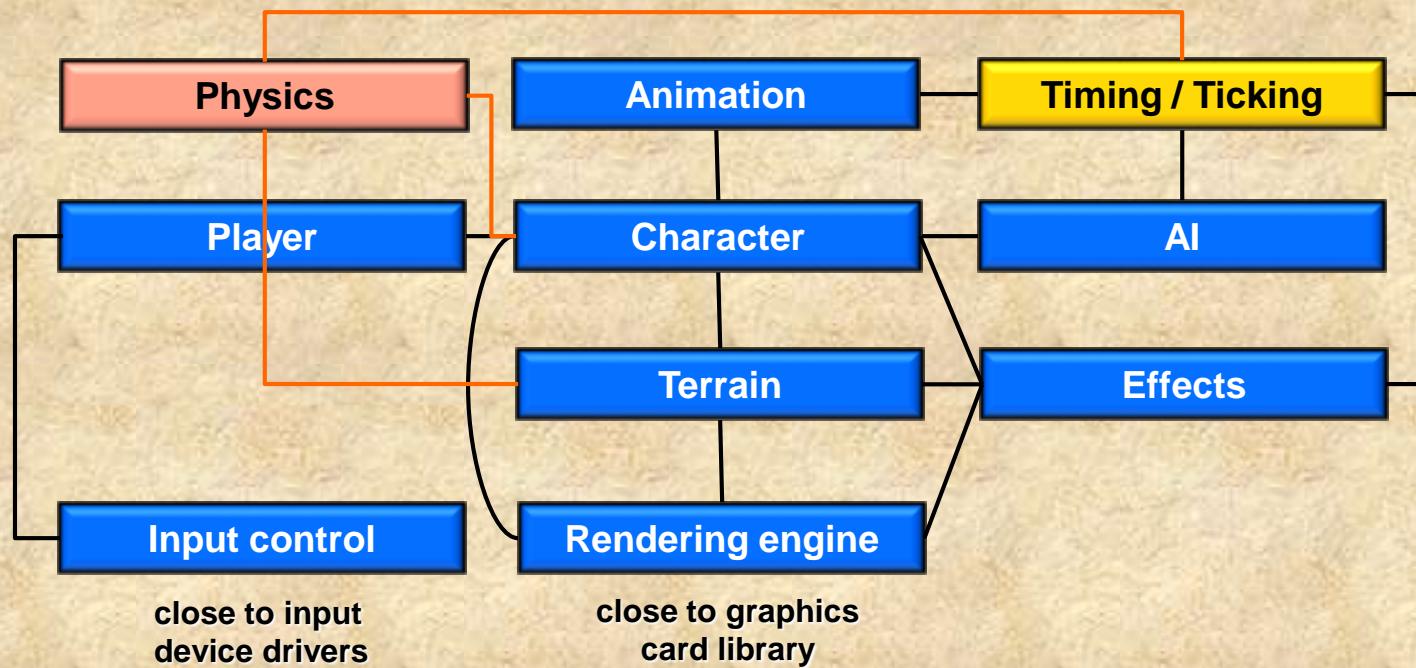


Βασικές αρχιτεκτονικές έννοιες (4/5)

- **Input**
 - ◆ Οι συσκευές και τα κανάλια εισόδου, η αντιστοιχία σε λογικές ενέργειες του παίκτη, ο τρόπος προσαρμογής της αντιστοίχησης αυτής
- **Animation**
 - ◆ Οι οπτικοί μετασχηματισμοί των χαρακτήρων, κινηματική, οι διάφοροι τύποι animation, ο τρόπος εκτέλεσης και διαχείρισης των animations, συντονισμός, μετάβαση από ένα animation σε κάποιο άλλο
- **Effects**
 - ◆ Ειδικές κατηγορίες οπτικών effects, τρόπος υλοποίησης, σχέση με το μοντέλο του terrain, σχέση με τους χαρακτήρες και τον παίκτη
- **AI**
 - ◆ Μηχανή λογικής των χαρακτήρων, μεταβάσεις sense → think → act, σύνδεση με τη μηχανή timing και ticking (λέγεται και AI frame rate)
- **Timing**
 - ◆ Χρονισμός και καθολικό ρολόι (χρόνος), τρόπος δρομολόγησης γεγονότων που βασίζονται στο χρόνο, κανόνες χρονο-δρομολόγησης
- **Physics**
 - ◆ Κυρίως rigid-body, νόμοι κίνησης (ταχύτητα, μάζα, συγκρούσεις)

Βασικές αρχιτεκτονικές έννοιες (5/5)

- Για τα παραπάνω δίνουμε μία πρώτη σκιαγράφηση των αναμενόμενων τμημάτων της βασικής μηχανής, καθώς και των μεταξύ τους σχέσεων





Περιεχόμενα

- Κατηγορίες παιχνιδιών
- Τι αντιμετωπίζουμε στην ανάπτυξη
- Η βιομηχανία παιχνιδιών
- Βασικές αρχιτεκτονικές έννοιες
- ***To game loop***
- Χάρτης μαθήματος



To game loop (1/10)

- Πρόκειται για το βασικό, ουσιαστικά θεμελιώδες, loop που εκτελούν σχεδόν όλες οι κατηγορίες παιχνιδιών δράσης και θα δούμε τον τρόπο υλοποίησης ανά κατηγορία πλατφόρμας
- PC games
 - Υπάρχει το system loop το οποίο τρέχει υποχρεωτικά και αφορά τη διαχείριση παραθύρων
 - ◆ repainting, minimize, maximize, resize, switch in, switch out, destroy
 - Θα πρέπει να γίνεται κλήση στο game loop, one iteration per time, μέσα από αυτό το loop
 - Εναλλακτικά μπορεί να το system loop να τρέχει ως ξεχωριστό thread



To game loop (2/10)

```
// data of the system loop thread and their initial values
bool hasPostedDestroyMessage = false;
HWND mainWindow = (HWND) 0;
HWND (*wndCreateCallback) (WNDPROC) = NULL;

...
// the function for the system loop thread
void _cdecl WinSystemThread (HANDLE setupEvent) {

    // creation of the main window
    if (!wndCreateCallback)
        if (isFullScreen)
            mainWindow = CreateFullScreenWindow();
        else
            mainWindow = CreateNormalWindow();
    else
        mainWindow = (*wndCreateCallback) (WinProc); ←

    // the system loop
    SetEvent(setupEvent);
    MSG msg;
    while (!hasPostedDestroyMessage)
        if (PeekMessage(&msg, mainWindow, 0, 0, PM_REMOVE)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
}
```

Υλοποίηση για Windows με χρήση δύο threads (εδώ είναι το message loop) Κάτι αντίστοιχο μπορεί να απαιτηθεί σε άλλες πλατφόρμες.

Σε περιπτώσεις που το βασικό παράθυρο στο οποίο γίνεται rendered το game παρέχεται από ανεξάρτητο subsystem (θα το χρειαστείτε εάν υλοποιήσετε game editor)



To game loop (3/10)

```
class Game { // app::Game namespace, the mother application
public:
    using Action = std::function<void(void)>;
    using Pred   = std::function<bool(void)>;
private:
    Action render, anim, input, ai, physics, destruct, collisions, user;
    Pred done;
    void Invoke (const Action& f) { if (f) f(); }
public:
    void SetRender (const Action & f) { render = f; }
    // rest of setters are similarly defined
    void Render (void)           { Invoke(render); }
    void ProgressAnimations (void) { Invoke(anim); }
    void Input (void)            { Invoke(input); }
    void AI (void)               { Invoke(ai); }
    void Physics (void)          { Invoke(physics); }
    void CollisionChecking (void) { Invoke(collisions); }
    void CommitDestructions (void) { Invoke(destruct); }
    void UserCode (void)          { Invoke(user); }
    bool IsFinished (void) const { return !done(); }
    void MainLoop (void);
    void MainLoopIteration (void);
};
```

We use a class that uses functions from unknown subsystems as a flexible extensible Façade pattern.

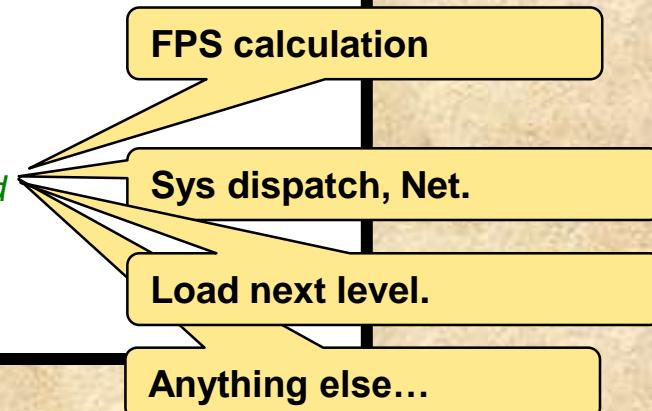
The traditional use of an interface with pure virtual methods and derivatives is not adopted here.



To game loop (4/10)

```
void app::Game::MainLoop (void) {
    while (!IsFinished())
        MainLoopIteration();
}

void app::Game::MainLoopIteration (void) {
    Render();
    Input();
    ProgressAnimations();
    AI();
    Physics();
    CollisionChecking();
    UserCode(); // hook for custom code at end
    CommitDestructions();
}
```





To game loop (5/10)

```
PROFILER_STARTTOTAL();

// STEP. I
if (!DisplayAll()) break;

// STEP. II
if (!InputHandling()) break;

HandleSlowMotion(); // Boxed

if (!paused && !InWait()) { // Boxed

    // STEP. III
    if (!ProgressAnimations()) break;

    // Step. IV
    if (!DoAI()) break;

    // Step. V
    if (!UserCallbacks()) break;

    // Step. VI
    if (!CollisionDetection()) break;

    // Step. VII
    if (!DestructionManagement()) break;
}

// FPS / LPS calculation. Does not depend on
// game loop time.
//
++loopCounter;
te = ugettime();
if ((te - ts) >= ONE_SECOND) {
```

Part of game loop, Under GO engine,
GEAR, Game Loop Framework (2001).

Εδώ το system loop
dispatching γίνεται στην αρχή
(δεν φαίνεται) πριν το profiling.

Τα user callbacks είναι κώδικας
που εκτελείται στο τέλος κάθε
game loop και μπορεί να
παρουσιάζει κάποια standard
πράγματα (π.χ., score,
περιγράμματα, ή κάποια
stencils της σκηνής).

Τα destructions είναι μία
τεχνική που θα δούμε βάσει της
οποίας τίποτε δεν κάνουμε
delete στο game logic, απλώς
το μαρκάρουμε ως dead,
κάνοντας μαζικά delete στο
τέλος toy loop.



Ένθετο

```
bool GameFramework::GameLoopIteration (void) {  
  
    Invoke(loopStartActions);  
    FPSCalculatorGet().LoopBegin();  
  
    // 1.input  
    if (!Input())  
        return false;  
  
    // 2.animation  
    AnimationClock::Tick();  
    AnimatorManagerGet().Progress();  
    ProgressSteppers(); ←  
    InvokeActions();  
  
    // 3.rendering  
    Render(); ←  
    ←  
    // 4.ai  
    for (auto& f : aiHandlers)  
        f();  
  
    GlobalLateDestructor::GetSingleton().commit();  
  
    return true;  
}
```

Custom loop, installed in Game,
ankh-1 engine, Game
Frameworks, 2016.

Linkage to physics

Here we choose to render after
handling animations



To game loop (6/10)

```
switch (message) { // Handle window messages of interest to the engine.

    // Posts WM_QUIT, causing message-loop exit.
    //

    case WM_CLOSE: { PostQuitMessage(0); break; }
    case WM_CREATE: { aladinWnd = wnd; break; }

    case WM_DESTROY: { // Due to system-oriented win destruction.
        havePostedDestroyMsg = true;
        NotifyWindowDestroyed(); // Notify window destroyed.
        break;
    }

    case WM_ACTIVATE: {
        if (LOWORD (wparam) == WA_INACTIVE)
            NotifyDisplaySwitchOut();
        else
            if (! (BOOL) HIWORD (wparam))
                PostMessage(
                    aladinWnd,
                    msgIdForCallProc,
                    (DWORD) ALADIN_DirectX::HandleDisplaySwitchIn,
                    0
                );
        break;
    }

    case WM_PAINT: {
        ValidateRect(wnd, NULL);
        PostMessage(
            aladinWnd,
            msgIdForCallProc,
            (DWORD) ALADIN_DirectX::HandleDisplaySwitchIn,
            0
        );
    }
}
```

Part of system loop, Under GO engine,
ALADIN, Direct X implementation
(2001).

Στο system loop θα σας απασχολούν ουσιαστικά τα events που σχετίζονται με:

(α) window destruction by the user,

(β) window gain / loose focus (pause / resume game),

(γ) window repaint (αντιμετωπίζεται ως focus in).

Events για window resize σας απασχολούν μόνο εάν το view terrain είναι resizable αντίστοιχα.



To game loop (7/10)

Smartphones

- Τα περισσότερα δεν έχουν multitasking αλλά έχουν multithreading
- To system loop εκτελείται ως ένα internal thread ενώ η κλήση στο game logic γίνεται συνήθως σε δύο σημεία
 - **rendering calls:** κλήση του game dispatcher με συγκεκριμένο χρονισμό (πχ, 30 / 60 φορές το δευτερόλεπτο)
 - **input events:** event handlers για κλήση του game input logic



To game loop (8/10)

loop is internal

game logic
organized with
callbacks

```
// system internal thread
while (not suspended or destroyed) {
    if (is time to render)
        invoke the registered GameDispatcher
    if (there is an input event)
        invoke the registered InputEventHandler
}

----- // input logic -----
void InputEventHandler (Event event) {
    record the 'event' in a local event queue
}

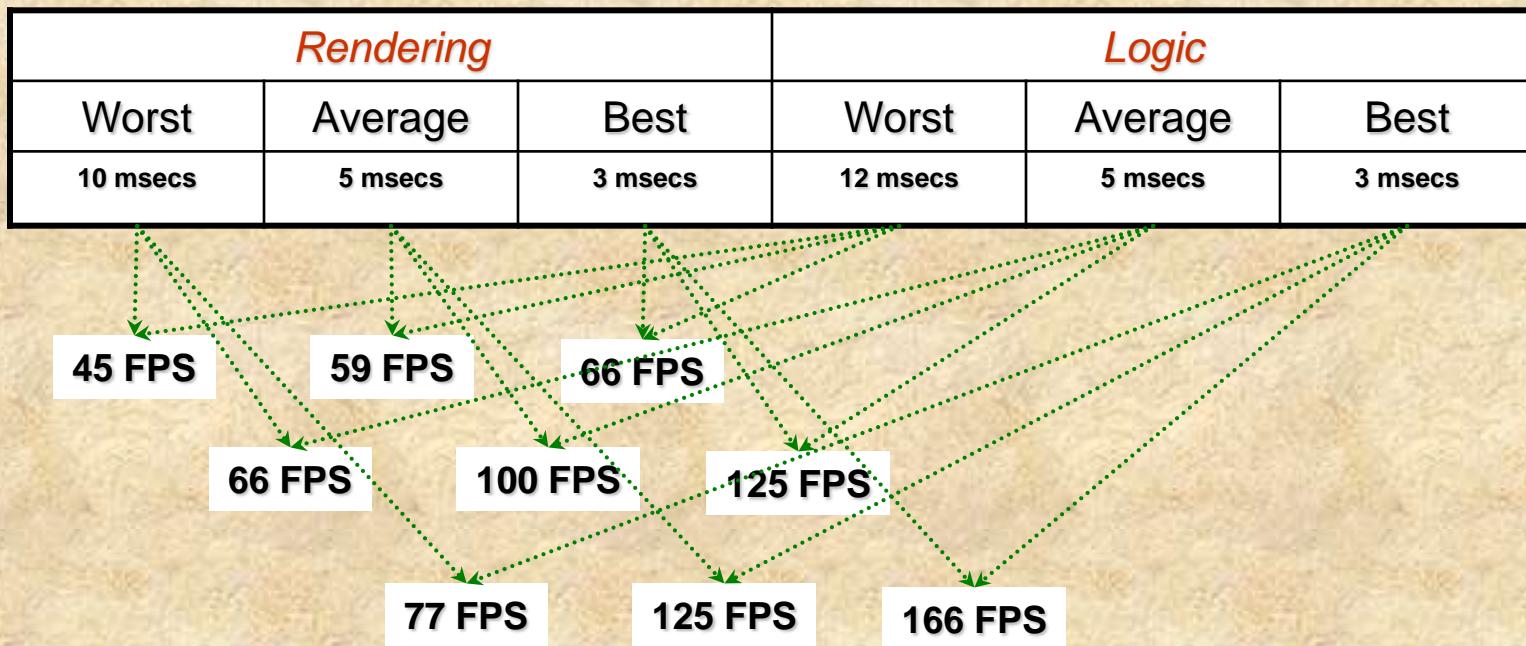
----- // game loop logic -----
void MainLoopOneIteration() {
    InputManagement(); // just reads from local input queue
    AnimationProgress();
    ArtificialIntelligence();
    CollisionChecking();
    CommitDestructions();
    FPSCalculation();
}

void GameDispatcher (...) {
    SetCurrentTime();
    MainLoopOneIteration();
}
```

game loop is implicit, split in two parts

To game loop (9/10)

- Υπολογισμοί για το πώς συμπεριφέρεται το frame rate του παιχνιδιού μας



Το παραπάνω μας δίνει χρήσιμη πληροφορία μόνο όταν όλες οι τιμές είναι πάνω / κάτω από το επιθυμητό / ανεπιθύμητο όριο

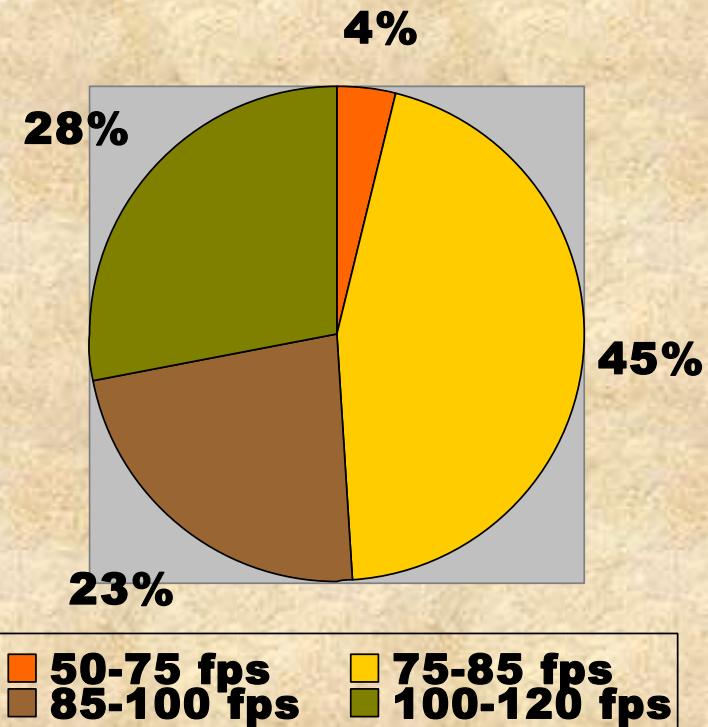
To game loop (10/10)

- Κατά την εκτέλεση θεωρούμε το MAX_FPS σε μία τιμή π.χ. 200. Έπειτα έχουμε:

```
unsigned fpsDistribution[MAX_FPS];
unsigned loopCounter=0;
...
while (NotFinished()) {
    ...
    ++fpsDistribution[fps];
    ++loopCounter;
}
```

• Στο τέλος ζωγραφίζουμε την κατανομή του frame rate στο συνολικό χρόνο του παιχνιδιού με τα εξής ποσοστά:

• Το ποσοστό του συνολικού χρόνου του παιχνιδιού με rendering j FPS είναι:
 $\text{fpsDistribution}[j]*100 / \text{loopCounter}$





Παράδειγμα: game profiler

Program execution profiling report.

Date and time of execution Tue Feb 28 08:59:21 2006

Main profiled unit information follows.

Total accumulated time 88.437000 secs,
total 3755 rounds have been profiled,

max unit time 156 msecs,
min unit time 15 msecs,

average unit time 23 msecs.

Information per profiled unit follows.

Total 8 profiled units.

Unit with symbolic name 'DY'.

Total accumulated time 66.325000 secs,

min unit time 15 msecs,

max unit time 93 msecs,

Rendering

average unit time 17 msecs,

unit round time 15 msecs, when main unit max reached,
while 74% of total profiling time spent for this unit.

Unit with symbolic name 'AM'.

Total accumulated time 6.298000 secs,

min unit time 0 msecs,

max unit time 63 msecs,

Animation manager

average unit time 1 msecs,

unit round time 63 msecs, when main unit max reached,
while 7% of total profiling time spent for this unit.

6.5 FPS min!
66.6 FPS max!

Unit with symbolic name 'IM'.

Total accumulated time 2.600000 secs,

min unit time 0 msecs,

max unit time 47 msecs,

Input

average unit time 0 msecs,

unit round time 31 msecs, when main unit max reached,
while 2% of total profiling time spent for this unit.

Unit with symbolic name 'CD'.

Total accumulated time 0.952000 secs,

min unit time 0 msecs,

max unit time 32 msecs,

Collision detection

average unit time 0 msecs,

unit round time 0 msecs, when main unit max reached,
while 1% of total profiling time spent for this unit.

Unit with symbolic name 'AI'.

Total accumulated time 11.980000 secs,

min unit time 0 msecs,

max unit time 78 msecs,

AI

average unit time 3 msecs,

unit round time 47 msecs, when main unit max reached,
while 13% of total profiling time spent for this unit.

Unit with symbolic name 'DM'.

Total accumulated time 0.158000 secs,

min unit time 0 msecs,

max unit time 16 msecs,

Destruction manager

average unit time 0 msecs,

unit round time 0 msecs, when main unit max reached,
while 0% of total profiling time spent for this unit.



Περιεχόμενα

- Κατηγορίες παιχνιδιών
- Τι αντιμετωπίζουμε στην ανάπτυξη
- Η βιομηχανία παιχνιδιών
- Βασικές αρχιτεκτονικές έννοιες
- To game loop
- **Χάρτης μαθήματος**



Χάρτης μαθήματος

- Από πλευράς κατασκευαστικής θα ασχοληθούμε κυρίως με:
 - Κατασκευή 2D 1/2 platform games με multiple layers (terrains, animations, sprites, platforms) και χρήση ενός physics engine (rigid body)
 - Τα μυστικά των bitmap-based graphics και τεχνικές με calculation pre-caching για επιτάχυνση και διάφορα dirty tricks
 - Χρήση απλών συστατικών στοιχείων για επίτευξη αισθητικά πλούσιας δράσης



Χάρτης μαθήματος

- Τα χαρακτηριστικά των παιχνιδιών αυτών είναι ότι βασίζονται σε **προκατασκευασμένα μοντέλα** χαρακτήρων και σκηνογραφίας
- Η μηχανή τα χρησιμοποιεί ως bitmaps ενώ όλοι οι μετασχηματισμοί βασίζονται σε **αποτύπωση εναλλακτικών φωτογραφιών** που δρομολογούνται από τη μηχανή με τον **κατάλληλο χρονισμό**
- Επομένως δεν έχουνε να κάνουμε καθόλου με γραφικά στοιχεία τύπου **γραμμών, πολυγόνων ή επιφανειών**, αλλά μόνο με τη **διαχείρηση bitmaps**



Artwork

