Figura 1: Unity vs Unreal Engine, imagine: HTLO.

Dezvoltarea de jocuri video - motoare grafice

Nume:	Chelcea Claudiu-Marian
Universitate:	Universitatea Politehnica din București
Facultate:	Facultatea de Automatică și Calculatoare
Specializarea	Calculatoare și Tehnologia Informației
Grupa:	312CA
Data:	03.19.2021

Cuprins

1 De ce un motor grafic? 2
- 1.1 **Avantaje:** 2
- 1.2 **Dezavantaje:** 2

2 Ce motoare grafice sunt populare? 3

3 Diferenţe dintre Unity si Unreal Engine 4
- 3.1 Unity 4
- 3.2 Unreal Engine 4

4 Grafica pe calculator = matematică 5
- 4.1 Shader pentru lumină 5
- 4.2 Shader pentru sticlă si metal 5

5 Înapoi la motoare grafice 7
- 5.1 Concluzie: 7

6 Bibliografie 8

1 De ce un motor grafic?

Un motor grafic este un sistem conceput pentru crearea și dezvoltarea de *jocuri video*.

Există mai multe motoare de joc, care sunt proiectate să funcționeze pe console de jocuri video și calculatoare personale.

Funcționalitatea de bază oferită de obicei de un motor grafic include un motor de randare (engleză "renderer") pentru grafică **2D** sau **3D**, un motor de fizică sau de detectare a coliziunilor (și răspunsul la coliziune), sunet, scripting, animație, inteligență artificială, rețea, streaming, memorie de management, suport de localizare, etc.

Procesul de dezvoltare a jocului este de multe ori economisit, în mare parte, prin reutilizarea / adaptarea unui motor asemănător pentru a crea jocuri diferite.

1.1 Avantaje:

- Majoritatea, dacă nu tot codul poate fi generat de aplicație, deci va puteți ocupă doar de aspect, logică.
- În această direcție, gestionarea memoriei, încărcarea sprite-urilor, iluminarea (în motoare complexe) etc. au fost proiectate și testate temeinic.
- Dacă motorul este capabil de "multiplatform", portabiltiatea către alte platforme va fi ușurată semnificativ.

1.2 Dezavantaje:

- Dacă există o eroare în motor, dacă nu este open source, nu o puteți remedia.
- Motorul nu a fost conceput special pentru jocul dvs., deci poate fi mai puțin eficient decât codul pe care îl scrieți special pentru acesta.
- Motoarele de joc nu sunt, în general, gratuite. [2]

[2] Puteți citi mai multe la https://gamedev.stackexchange.com/questions/859/what-are-the-advantages-and-disadvantages-to-using-a-game-engine.

2 Ce motoare grafice sunt populare?

Pentru crearea unui joc la scala mică, este recomandat să se folosească un limbaj de programare, pentru a obține performanța maximă. Pentu lansarea unui joc de dimensiune medie, care poate fi vândut către alți oameni, este recomandat să se folosească un motor grafic, deoarece ușurează implementarea functionalitatilor de baza și, în cele mai multe cazuri, acestea sunt mai performante decât codul scris manual, întrucât au fost testate și optimizate constant până au fost introduse în aplicație.

Studio-urile de jocuri, în general, combină aceste două moduri de a crea un joc, implementând logica jocului prin Visual Scripting sau alte facilități oferite de motorul grafic folosit, adăugând propriile funcționalități prin cod, în general C++, iar, în cazuri rare, Csharp, Python, Java.

Lista **celor mai populare 8** motoare grafice, conform GAMEDESIGNING:[1]

1. Unreal Engine.
2. Unity
3. GameMaker
4. Godot
5. AppGameKit
6. CryEngine
7. Amazon Lumberyard
8. RPG MAKER

O analiză de la Gamasutra a constatat că **Unreal Engine** și **Unity** sunt printre cele mai populare motoare de joc. Această analiză s-a bazat pe jocuri lansate pe Steam și Itch.io.

[1] Aceste statistici sunt actuale la data de 19 Martie 2021.

3 Diferenţe dintre Unity si Unreal Engine

Răspunsul la care este mai bun este o chestiune dificilă. Unii vor susține că Unreal este mai bun doar pentru faptul că este o alegere de top pentru studiourile AAA. Cu toate acestea, alții vor cita faptul că Unity este mai bine rotunjită și, pentru dezvoltatorii independenți, este adesea o intrare mai bună în industrie. Obiectiv, însă, este unul mai bun decât celălalt?

3.1 Unity

Unity este frecvent utilizată deoarece este:

- O platformă de creare 3D deschisă și avansată în timp real.
- Folosit pentru a produce jocuri în mai multe genuri.
- Integrat cu instrumente cheie de dezvoltare a jocurilor, inclusiv IDE-uri, instrumente grafice și controlul versiunilor.

3.2 Unreal Engine

Unreal Engine este frecvent utilizat deoarece este:

- Un motor de joc multiplataforma, cu suport pentru peste 25 de platforme.
- O platformă de dezvoltare 3D în timp real.
- Integrat cu instrumente cheie de dezvoltare a jocurilor, inclusiv IDE-uri, instrumente grafice și controlul versiunilor.

De ce e util să folosim un motor grafic? În caz că nu folosim un motor grafic, o să folosim formule de tipul următor pentru un simplu shader asupra oceanului: [3]

$$\mathbf{P}(x,y,t) = \begin{pmatrix} x + \sum\left(Q_i A_i \times \mathbf{D}_i.x \times \cos\left(w_i \mathbf{D}_i \cdot (x,y) + \varphi_i t\right)\right), \\ y + \sum\left(Q_i A_i \times \mathbf{D}_i.y \times \cos\left(w_i \mathbf{D}_i \cdot (x,y) + \varphi_i t\right)\right), \\ \sum\left(A_i \sin\left(w_i \mathbf{D}_i \cdot (x,y) + \varphi_i t\right)\right) \end{pmatrix}.$$

[3] Autor necunoscut al formulei

4 Grafica pe calculator = matematică

4.1 Shader pentru lumină

Efectul de estompare al luminilor punctiforme se numește de obicei atenuare. Atenuarea unei lumini reale este guvernată de legea pătratului invers care spune că puterea luminii este invers proporțională cu pătratul distanței de la sursa de lumină. Aceasta este descrisă în termeni matematici prin următoarea formulă:

$$L_{distance} = \frac{L_1}{distance^2} \qquad (L_1 \quad este \quad intensitatea \quad luminii)$$

Această formulă nu oferă rezultate bune în grafica 3D. De exemplu, pe măsură ce distanța devine mai mică, puterea luminii se apropie de infinit. În plus, dezvoltatorul nu are control asupra rezultatelor, cu excepția setării puterii inițiale a luminii. Acest lucru este prea limitativ. Prin urmare, se adaugă câțiva factori la formulă pentru a o face mai flexibilă:

$$L_{distance} = \frac{L_1}{Atenuare_{constanta} + Atenuare_{liniara} * distanta + Atenuare_{exponentiala} * distanta^2}$$

4.2 Shader pentru sticlă si metal

În shader-ul de metal și sticlă, vom utiliza aproximarea Schlick pentru a calcula reflectanța F și transmitanța T. Reflectanța (sau coeficientul de reflecție) este fracțiunea de lumină care se reflectă, iar transmitanța (sau coeficientul de transmisie) este fracțiune de lumină care este refractată. Sub modelul nostru, toată lumina este fie reflectată, fie transmisă, deci $T + F = 1$.

Teta este unghiul dintre suprafața normală și vectorul de la cameră la vârful i. R este reflectanța la incidență normală (adică fracția de lumină reflectată atunci când $Teta_i$ este zero). Este o constantă care va fi transmisă în umbră Determinăm F prin aproximarea lui Schlick după cum urmează:

$$F \approx R_0 + (1 - R_0)(1 - cos\theta_i)^5 \tag{1}$$

Modelul de iluminare Phong:
Modelele de iluminare Phong și Blinn-Phong au fost primele modele de iluminare încercate mimând fenomene de reflexie specular. Cu toate acestea, aceste modele nu se bazează pe realitate pentru că modelul Phong împarte iluminatul în trei termeni diferiți: ambient, difuz și specular. Conform modelului Phong:

$$k_{spec} = \cos^n(E, R) \tag{2}$$

unde n este coeficientul specular, E indică în direcția vizualizatorului, iar R este reflectarea vectorului luminii primite despre vectorul normal. Valorile mai mari ale n produc momente speculare mai luminoase, mai localizate.

Un model mai bun de iluminare: Cook-Torrance.
Există multe alte tehnici de iluminare care modelează mai bine distribuția microfacetelor (avioane plate foarte mici) care alcătuiesc o suprafață. Una dintre acestea este cea bazată fizic Modelul Cook-Torrance, utilizat în stocul PBR shader într-o serie de aplicații populare de creare 3D (inclusiv Blender). Veți folosi acest model pentru a calcula termenul specular. În special,

$$k_{spec} = \frac{DFG}{V * N}$$

unde D = factorul de distribuție Beckmann:

$$D = \frac{exp(-\tan^2(\alpha)/m^2)}{4m^2 \cos^4(\alpha)},$$

$\alpha = arccos(n * h)$, F este coeficientul Fresnel (calculat folosinde aproximarea lui Schlick), iar G este geometric termen de atenuare cauzat de auto-umbrirea microfacetelor de la suprafață,

5 Înapoi la motoare grafice

Am observat că construirea proprie a shader-lor este o muncă foarte dificilă și necesită foarte multe cunoștințe de matematică și fizică.

Pentru a face lucrurile mai ușoare, un dezvoltator de jocuri începător va folosi întotdeauna un motor grafic. Din statistici [4][5][6] a reieșit că **Unity** a reieșit că **Unity** și **Unreal Engine** sunt cele mai utilizate motoare grafice, așa că, care este diferența dintre ele?

Topic	Unity	Unreal Engine
Preț	Gratuit și 5% din profit	Gratuit și 5% din profit
Interfață	Ușor-Medium	Greu
Resurse disponibile	Foarte multe	Limitate
Dezvoltate jocuri	Programare C#	Programare C++
Grafică	Medie	Foarte bună

5.1 Concluzie:

Alegerea unui motor grafic pentru realizarea unui joc este foarte importantă și este făcută, în special, în funcție de tipul de joc dorit.

Astfel, pentru realizarea jocurilor 2D - platformer, 2.5D sau jocuri pentru telefon, jocuri care nu necesită cea mai mare atenție la detaliile grafice, este preferat motorul Unity, pentru rapiditatea pe care o oferă în dezvoltarea jocurilor video.

Pentru realizarea jocurilor 3D, cu predilecție a jocurilor AAA, este utilizat Unreal Engine, datorită tuturor functionalitatilor pe care le oferă, sau, în caz contrar, se utilizează un motor grafic dezvoltat de către companie, denumite motoare 'în-house'.

[4]Statistici: https://www.perforce.com/blog/vcs/most-popular-game-engines
[5]Statistici: https://www.gamedesigning.org/career/video-game-engines/
[6]Statistici: https://gamedevacademy.org/best-game-engines/

6 Bibliografie

[1] Wikipedia: motor grafic link

[2] Gamedesigning: website link

[3] Gamasutra: website link

[4] HTLO: website link

[5] GameDev Academy: website link

[6] Sursa formulei shader-ului de lumină

[7] Sursa formulei shader-ului de sticlă si metal