Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.42:794
DOI: https://doi.org/10.24867/27BE07Jovanovic

РАЗВОЈ И АРХИТЕКТУРА *GAME ENGINE* ПОГОНА ИГРЕ

GAME ENGINE DEVELOPMENT AND ARCHITECTURE

Дејан Јовановић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Главни предмет рада јесте истраживање и проучавање области развоја погона игре за креирање 2D и 3D рачунарских игара. На основу стечених информација и знања, имплементиран је и у раду представљен Game Engine погон игре, десктоп апликација за Windows оперативни систем посебно дизајнирана за креирање и руковање јединственим компонентама пројеката 3D игара, користећи C# и C++ програмске језике. Имплементација апликације и њене архитектуре детаљно су објашњене, уз свеобухватан преглед теоријске основе и тренутног стања у области развоја погона игара.*

Кључне речи: *Game Engine погон игре, 3D модел, погон за графику, MVVM, DLL*

Abstract – *The main subject of the Master's Thesis is to research and study the field of game engine development for creating 2D and 3D computer games. Based on the acquired information and knowledge, the Master's Thesis features Game Engine, desktop application for a Windows operating system that is specifically designed to create and handle the unique 3D games project components, using C# and C++ programming languages. The implementation of the application and its architecture are explained in detail, with a comprehensive overview of the theoretical basis and current situation in the field of game engine development.*

Keywords: *Game Engine, 3D models, graphics engine, MVVM, DLL*

1. УВОД

Током развоја игара, први проблем са којим се развојни тимови суочавају, поред самог дизајна игре, јесте избор алата за развој. Осим избора алата, развојни тимови се сусрећу са бројним изазовима. Један од кључних фактора је оптимизација перформанси игре. Како би правилно функционисала и пружала задовољавајући доживљај корисницима, важно је осигурати да игра ради без застоја, посебно на различитим платформама и уређајима. Поред тога, дизајнирање ефикасне логике и механике захтева пажљиво планирање и имплементацију. Ефикасно управљање меморијом може побољшати перформансе игре и смањити потрошњу ресурса.

Узимајући све наведене изазове у обзир, тимови за развој игара често користе комбинацију алата, технологија и приступа како би постигли оптималне резултате. Балансирање између функционалности, флексибилности, брзине развоја и перформанси игре кључно је за стварање доживљаја који ће привући и задржати пажњу играча. Узимајући у обзир наведене факторе, за овај пројекат креиран је *Game Engine* погон игре. Представља комбинацију различитих елемената погона игара написаних у *C++* и *C#* програмским језицима, како би се искористиле предности које ови програмски језици пружају, уз могућност слободног развијања игара у жељеном правцу.

2. ПРОЈЕКТОВАЊЕ И ИМПЛЕМЕНТАЦИЈА ТЕХНИЧКОГ РЕШЕЊА

2.1. Рачунарске игре, историјат и развој

Рачунарске игре историјски датирају из 1950-их година, када су научници почели да дизајнирају једноставне игре за рачунаре који су тада били у развоју. Данас, садржај рачунарских игара је невероватно разноврстан, и њихов утицај на друштво је значајан. Показало се да игре имају позитиван утицај на когнитивне функције, укључујући побољшану меморију, пажњу и способност решавања проблема. Такође, примењиве су у образовним сврхама, при чему многе школе користе игре како би подучавале предмете попут историје, науке и математике. Међутим, постоји и брига због потенцијалних негативних ефеката прекомерног играња, попут зависности и социјалне изолације. Са техничке стране представљају велики изазов за развојне тимове. Захтевају високе перформансе и оптимизацију. Осим тога, развојни тимови морају имати разумевање за дизајн, уметничке аспекте, музику и звук како би створили занимљив и привлачан доживљај за играче.

2.2. Развој игара

Развој игара је процес стварања видео игара, од почетног концепта до крајњег производа. Представља сложен и изазован процес, захтевајући дубоко разумевање технологије и принципа дизајна. Захтева широк спектар техничких и уметничких вештина, укључујући програмирање, дизајнирање, визуелну уметност, аудио инжењеринг, управљање пројектима и још много тога. Процес развоја игара обично почиње фазом концептуализације и дизајна, где развојни тимови одређују основну механику, причу, карактере и друге битне елементе. Фаза програмирања је срце развоја, где развојни тимови користе погоне игара за

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, ред. проф.

имплементацију основних механика игре. Стварање звука је још један кључни аспект развоја, јер може значајно утицати на укупну атмосферу игре и доживљај играчима.

2.3 Погон игре

Погон или мотор игре [1] је кључна софтверска компонента интерактивних програма, која омогућава развојним тимовима креирање богатих и узбудљивих видео игре, смањујући време и потребне ресурсе чиме се значајно убрзава развојни процес. Пружа сет унапред дефинисаних алата, библиотека и функција за лакше дизајнирање, имплементацију и тестирање. Омогућава развојним тимовима да креирају комплексне и интерактивне светове игара са дефинисаним правилима игре, додајући им карактере, објекте, анимацију и физику која подсећа на реалан свет. Поред основних елемената попут графичког приказа за *2D* и *3D* графику, симулације физичких закона, звука, скрипти, анимација и умрежавања, погони игара пружају и могућност за коришћење вештачке интелигенције, детекцију судара и одговор на сударе. Пре погона игара, игре су обично биле стваране као јединствени ентитети, дизајниране од нуле. Услед појаве погона игара, развојни тимови уместо да логику игре имплементирају од нуле, лиценцирају основне делове софтвера и фокусирају се на дизајн сопствених ресурса игре. Поновно употребљиви погони игара омогућавају бржи и лакши развој, што је драгоцена предност у конкурентној индустрији видео игара. На слици 1, приказана је поновна употреба погона игара, који је представио Џејсон Грегори (Jason Gregory) у својој књизи „*Game Engine Architecture, Second Edition*“ [1].

Слика 1 - *Граф поновне употребљивости погона игре који је представио Џејсон Грегори*

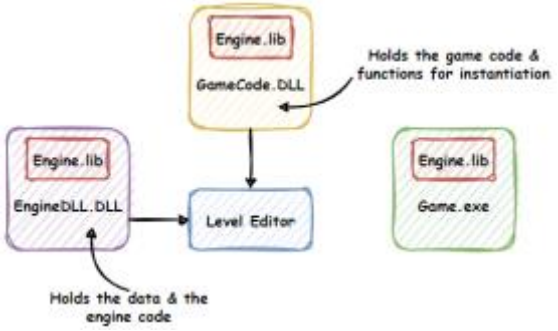
Све компоненте у погону игре се граде и интегришу како би подржале циљеве развоја игара. Неке од најважнијих компонената су главни програм игре (енгл. *main game program*), уређаји за унос информација (енгл. *input devices*), погон за графику (енгл. *rendering engine*), погон за аудио (енгл. *audio engine*), вештачка интелигенција (енгл. *AI, artificial intelligence*), погон за физику (енгл. *physics engine*), и умрежавање (енгл. *networking*).

Један од највећих изазова са којима се суочавају развојни тимови погона игара су хардверска ограничења. Што је игра већа (било да се ради о већем броју објеката или сложенијој логици) погон игара мора обавити више рачунања, што захтева високо оптерећење *CPU*-а. Перформансе игре могу бити утицане брзином и количином доступне меморије на хард-диску (енгл. *HDD, hard disk drive*). У случају споријег *HDD*-а, може доћи до дужих времена учитавања објеката и саме игре. Још један фактор хардверског ограничења је количина и брзина *RAM* меморије (енгл. *random-access memory*). Неки развојни тимови одлучују да већину ресурса учитају у *RAM* меморију како би убрзали процес учитавања. Међутим, за веће игре или игре са високом графичком резолуцијом, ово може бити изазов, јер захтева велике количине *RAM* меморије. Иако је циљ погона игара да омогући креирање више различитих игара, то није у потпуности остварив. Свака нова игра која се развија на истом погону игре је само варијација претходне, а уколико развојни тимови желе да направе другачију врсту игре, мораће да прилагоде или додају нове функционалности у погон игре. Издавачи, суочени са снажном конкуренцијом на тржишту игара, често задржавају своја открића и алате за себе. Многи развојни тимови стварају интерна решења и алате за креирање погона игре, што отежава другима развојним тимовима да сазнају како су они развили своја решења. Ово ствара недостатак стандардизације у методама и праксама. Као резултат, нови развојни тимови морају сами да истражују како да креирају погон игре, јер не постоје универзалне смернице.

3. ПРИКАЗ ИМПЛЕМЕНТИРАНОГ РЕШЕЊА

3.1. Генерална архитектура *Game Engine* погона

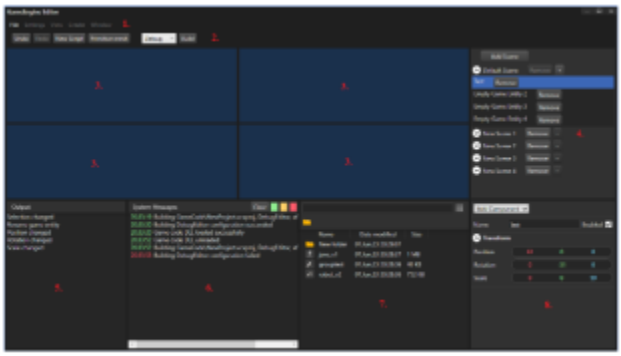
На слици 2, приказана је архитектура *Game Engine* погона игре. Главна компонента *Game Engine* погона игре је статичка библиотека погона (*Engine.lib*) која пружа све неопходне функционалности за покретање игре. Међутим, сама библиотека није довољна за креирање комплетног пројекта игре. Користи се и едитор нивоа (*Level Editor*) за изградњу корисничког интерфејса. Један од кључних елемената који олакшава комуникацију између погона игре и едитора нивоа су библиотеке динамичких веза [2]. Ове библиотеке омогућавају ефикасну размену функционалности и података између два дела *Game Engine* погона игре. Потребни су *DLL* погона (*EngineDLL.DLL*) и *DLL* кода игре (*GameCode.DLL*) како би се увезале функције из оба *DLL*-а у едитор нивоа ради њихове успешне комуникације.



Слика 2 - *Архитектура Game Engine погона игре*

3.2. Дизајн корисничког интерфејса едитора нивоа

На слици 3, приказан је едитор нивоа са свим својим компонентама интерфејса означеним бројевима од 1 до 8. Те компоненте су трака менија (1), трака са алатима (2), четири независна рендер екрана (3), прозор сцена и њихових ентитета (4), прозор за испис порука (5), прозор за системске поруке (6), претраживач садржаја (7) и, прозор својства селектованог ентитета сцене (8).



Слика 3 - *Приказ едитора нивоа Game Engine погона*

3.3. Пројекат – сцене – ентитети игре – компоненте

У процесу развоја игара пројекат, сцене, ентитети и компоненте игре имају важну улогу у организацији и структури игре као њени кључни концепти. На слици 4, приказана је архитектуре овог концепта унутар *Game Engine* погона игре. Пројекат је централни контејнер који садржи све податке и ресурсе игре, сцене дефинишу подручја за игру унутар саме игре, ентитети игре представљају интерактивне елементе, а компоненте пружају специфичне функционалности и понашања ентитетима.

Слика 4 - *Архитектура игре креиране у Game Engine*

3.4. Ентитети оријентисани на податке

Ентитети оријентисани на податке (енгл. *Data-Oriented Entitiec*) [3] је приступ организацији података у играма. У овој парадигми, фокус је на ефикасном складиштењу и обради података, са раздвајањем одговорности између података и понашања. Стварни подаци за сваки ентитет се складиште у одвојеним компонентама, које су структурисане колекције података. Нуди предности у погледу перформанси, оптимизацијом шаблона приступа подацима и манипулацијом фрагментације меморије. Овај систем се обично користи у сценаријама са великим бројем ентитета или када су перформансе кључни фактор. У *Game Engine* погону игре, овај приступ представља напредну верзију дизајна оријентисаног на податке, при чему су ентитети само листа индекса, без класа или сличних концепата (приказано на слици 5).

Слика 5 - *Ентитети оријентисаних на податке у Game Engine погону игре*

Овај приступ је додатно убрзао процес, смањујући оптерећење процесора и број пропуста кеш меморије.

3.5. Скрипте ентитета

Скрипта ентитета у *Game Engine* погону игре функционише као „мозак“ погона игре, дефинише понашање елемената игре. С обзиром да се ентитети излажу коду игре путем класа ентитета, оне се наслеђују како би се конструисала основна класа за све скрипте ентитета. Дакле, класа ентитета служи као основа за скрипту ентитета, која је темељ за све друге класе скрипти.

Све класе скрипти заједно чине код игре. Ове класе скрипти делују као компоненте ентитета у погону игре. Имплементацијом ове логике добија се архитектура где ентитет има једну скрипт компоненту која има показиваче на инстанце кодова скрипти складиштених негде у меморији.

3.6. Проточна обрада геометрије и ресурса

Компонента проточне обраде геометрије (енгл. *geometry pipeline*) у *Game Engine* погону игре је одговорна за трансформацију и приказивање објеката на екрану. Састоји се од низа фаза које манипулишу и обрађују геометријске податке како би се генерисала коначна слика. Геометрија може бити импортована као *3D* модел креиран алатима за креирање ресурса. Импортовани модели се шаљу на процес обраде и паковања да би се креирао ресурс у формату који *Game Engine* погон игре може користити и чувати као датотеку ресурса у едитору нивоа. Проточна обрада ресурса (енгл. *asset pipeline*) обухвата скуп процедура и алата који се користе за управљање импортом, обрадом, оптимизацијом и интеграцијом различитих ресурса унутар *Game Engine* погона игре. Олакшава ефикасно управљање ресурсима током читавог процеса развоја игара омогућавајући развојним тимовима да одрже структурисан и ефикасан радни ток. Побољшава сарадњу између уметника и програмера као чланова развојних тимова, смањујући појаву грешака и повећава перформансе и визуелни квалитет ресурса унутар игре.

3.7. Проточна обрада *Direct3D*-а

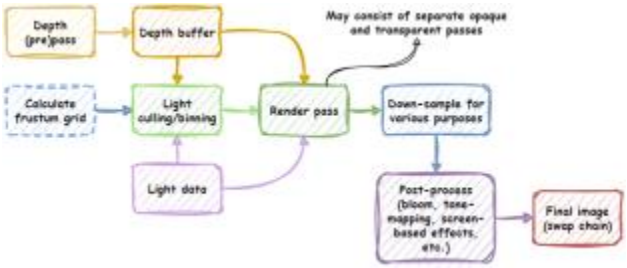
Direct3D [4] је графички интерфејс за програмирање апликација развијен од стране *Microsoft*-а. Користи се претежно за рендеровање тродимензионалне графике и креирање интерактивних мултимедијалних апликација, посебно у контексту развоја игара. Проточна обрада графике или ток графике (енгл. *graphics pipeline*) [5] назива се и проточна обрада *Direct3D*-а или ток *Direct3D*-а.

У *Game Engine* погону игре односи се на низ операција кроз које *GPU* пролази да би приказао *3D* графику на екрану рачунара. Овај процес је описан као проточна обрада рендеровања са неколико различитих фаза, које трансформишу геометријске податке и текстуре у коначну слику која се рендерује.

3.8. *Forward+* архитектура рендеровања

Реалистично осветљење сцене представља изазован, ако не и најизазовнији део рендеровања. *Game Engine* погон игре користи *Forward+* [6] архитектуру осветљења у реалном времену (слика 6). *Forward+* је техника рендеровања која комбинује *Forward* рендеровање са уклањањем поплочаног светла како

би се смањио број светала који се морају узети у обзир током сенчења. Побољшава *Forward* рендеровање тако што прво одређује која светла се преклапају са којом површином у простору екрана. Током фазе сенчења, потребно је узети у обзир само светла која се „потенцијално“ преклапају са тренутним фрагментом. Састоји се од пролаза уклањања светлости (енгл. *light culling pass*), непрозирног пролаза (енгл. *opaque pass*), и транспарентног пролаза (енгл. *transparent pass*).

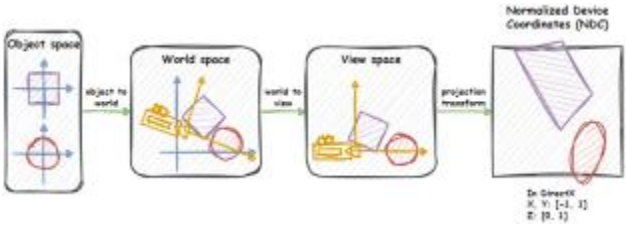


Слика 6 - *Forward+ архитектуре рендеровања у Game Engine погону игре*

3.9. Трансформација објеката

Game Engine погон игре врши скуп различитих математичких операција трансформације објеката пре него што се објекти рендерују и прикажу на екрану (слика 7). Објекти су дефинисани на основу локалног координатног система који се заснива на простору објекта (*Object space*). Свако теме у *3D* моделу има позицију која је релативна у односу на локалне координате објекта. Ова позиција у свету игре је трансформација света и често је комбинација трансформације, ротације и скалирања оригиналног објекта. После ове трансформације, за објекте се каже да се налазе у простору света (*Word space*).

Неопходно је да *Game Engine* погон игре изрази позицију и оријентацију сваког објекта кроз координатни систем камере познат као простор погледа (*View space*) камере. Следећа фаза трансформације објекта је трансформација пројекције (*Projection transformation*) у којој погон игре сваку тачку у простору видљиву камери уноси у NDC. У овој фази погон игре компресује све објекте које камера види у оквиру. Оно што се види на равни оквира камере биће скалирано према величини оквира и приказано на екрану током растеризације.



Слика 7 - *Приказ процеса трансформације објекта у Game Engine погону игре*

3.9. *FBX* формат датотеке

FBX [7] је формат датотеке, који се првенствено користи за размену *3D* модела, анимација и других података између различитих софтверских апликација. Сврха *FBX*-а је да обезбеди свестран и компатибилан формат датотеке, који пружа стандардизован начин за пренос сложених *3D* асета између различитих програма, без губитака важних података или визуелне вредности. Може садржати широк спектар података, укључујући геометрију, текстуре, анимацију скелета, материјале, мешање облика, светла, камере и још много тога. Омогућава очување хијерархије на различитим софтверским платформама, олакшавајући сарадњу развојних тимова. Користећи *FBX*, *Game Engine* погон игре може несметано делити ресурсе, поједностављујући радне токове и побољшавајући компатибилност у раду са другим погонима игара. У основи *FBX*-а лежи концепт графа сцене, служи као контејнер за све елементе. Граф сцене је изграђен на хијерархијској структури састављеној од *FBX* чворова (веза родитељ-дете) који представљају одређени елемент у сцени.

4. ЗАКЉУЧАК

У овом раду истражена је и описана комплексност процеса развоја погона игара и пажљиво деконструисана кроз два кључна поглавља. Објашњен је сложен дизајн и имплементација техничког решења са описом слојева развоја игара, како би се истакли основни темељи конструкције погона игара. Нуди увид у различите аспекте у оквиру развоја пружајући визуелну манифестацију концептуалног дизајна и имплементације кроз развој *Game Engine* погона игре.

5. ЛИТЕРАТУРА

[1] J. Gregory, Game Engine Architecture, Second Edition, New York: A K Peters/CRC Press, 2014.

[2] Microsoft, "What is a DLL," 04 2023. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library. [Accessed 07 2023].

[3] D. Engelbrecht, "Building a Data-Oriented Entity System," 05 2017. [Online]. Available: https://bitsquid.blogspot.com/search?q=Building+a+Data-Oriented+Entity+System. [Accessed 08 2023].

[4] Microsoft, "Direct3D," 09 2021. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/direct3d. [Accessed 08 2023].

[5] F. Giesen, "A trip through the Graphics Pipeline 2011," 07 2011. [Online]. Available: https://fgiesen.wordpress.com/category/graphics-pipeline/. [Accessed 08 2023].

[6] Jeremiah, "Forward vs Deferred vs Forward+ Rendering with DirectX 11," 09 2015. [Online]. Available: https://www.3dgep.com/forward-plus/. [Accessed 08 2023].

[7] Autodesk, "Adaptable file format for 3D animation software," 2023. [Online]. Available: https://www.autodesk.com/products/fbx/overview. [Accessed 08 2023].

Кратка биографија:

Дејан Јовановић рођен је 1994. године у Лозници. Након средње школе, 2013. године уписује Факултет техничких наука у Новом Саду. Мастер академске студије на смеру Софтверско инжењерство и информационе технологије уписује 2020. године.