**AZƏRBAYCAN RESPUBLİKASI ELM VƏ TƏHSİL NAZİRLİYİ**
**AZƏRBAYCAN TEXNİKİ UNİVERSİTETİ**
**YÜKSƏK TƏHSİL İNSTİTUTU**

*Əlyazması hüququnda*

**Mehman Aydın oğlu Şəfiyev**
**Taleh Əhliman oğlu Məmmədli**
**Rəşad Teymur oğlu Məmmədov**

# "OYUNLARIN YARADILMASI MƏRHƏLƏLƏRİNİN TƏDQİQİ VƏ KİBERTƏHLÜKƏSİZLİK MƏSƏLƏLƏRİNİN ARAŞDIRILMASI"

mövzusunda

**MAGİSTR DİSSERTASİYASI**

060632 – "İnformasiya texnologiyaları və sistemləri mühəndisliyi"

249315 – "Kibertəhlükəsizlik (SABAH)"

**Elmi rəhbər:** t.ü.f.d., dos. Sevinc Həmzağa qızı Əliyeva

**BAKI – 2024**

**AZƏRBAYCAN TEXNİKİ UNİVERSİTETİ**

**YÜKSƏK TƏHSİL İNSTİTUTU**

# *MAGISTRANTIN ANDI*

"Oyunların yaradılması mərhələlərinin tədqiqi və kibertəhlükəsizlik məsələlərinin araşdırılması" mövzusunda təqdim etdiyimiz magistrlik dissertasiyasını elmi əxlaq normalarına və istinad qaydalarına tam riayət etməklə və istifadə etdiyimiz bütün mənbələri ədəbiyyat siyahısında əks etdirməklə yazdığımıza and içirik və magistrlik dissertasiyasının AzTU Kitabxana İnformasiya Mərkəzində saxlanılması, həmin mərkəz tərəfindən AzTU Rəqəmsal Repozitoriyasına daxil edilərək repozitoriyanın veb saytında yerləşdirilməsinə icazə veririk.

Mehman Şəfiyev

Taleh Məmmədli

Rəşad Məmmədov

# XÜLASƏ

İşin adı: Oyunların yaradılması mərhələlərinin tədqiqi və kibertəhlükəsizlik məsələlərinin araşdırılması.

Magistr dissertasiyası oyunların yaradılması mərhələlərinin tədqiqinə və kibertəhlükəsizlik məsələlərinin araşdırılmasına həsr edilmişdir.

Bu dissertasiya işinin əsas məqsədi oyunların yaradılmasında istifadə olunan texnologiyaların işləmə prinsipi və onlara yönəlmiş kibertəhdidlərin qarşısının alınmasına yönəlmiş məsələlərin araşdırılmasıdır.

İş 3 fəsil, 6 alt fəsildən ibardətdir.

I fəsil Kompüter oyunlarının inkişafı adlanır. Burada oyunların inkişafı zamanı istifadə olunan texnologiyalar və bu texnologiyaların gələcəkdə hansı istiqamətə doğru gedəcəyi haqda məlumat verilir. Bu fəsildə, oyun mühərrikləri üzərindən müxtəlif simulyasiyalar həyata keçirilmişdir.

Magistr dissertasiyasının II fəsli oyunların qurulmasında test mərhələlərindən bəhs edir. Bu fəsildə, oyunların qurulması zamanı istifadə olunan optimizasiya üsulları analiz edilmiş və testlər həyata keçirilmişdir.

Magistr dissertasiyasının III fəsil Oyunların yaradılması zamanı təhlükəsizlik məsələləri üzərindən analiz və tədqiqat aparılmışdır. Bu fəsildə  Müxtəlif şifrələmə metodları ilə oyun mühərriki daxilində olan məlumatların şifrələnməsi həyata keçirilmişdir.

Dissertasiya işinin sonunda nəticə və təkliflər, istifadə edilmiş ədəbiyyatların siyahısı verilmişdir.

## SUMMARY

Title of the work: Research on the stages of game development and the investigation of cybersecurity issues.

The master's thesis is dedicated to the study of the stages of game development and the investigation of cybersecurity issues.

The main objective of this dissertation is to explore the working principles of the technologies used in game development and to investigate issues aimed at preventing cyber threats directed at them.

The work consists of 3 chapters and 6 sub-chapters.

Chapter I: Development of Computer Games. This chapter provides information about the technologies used during the development of games and the future direction of these technologies. Various simulations were conducted using game engines in this chapter.

Chapter II: Testing Phases in Game Development. The second chapter of the master's thesis discusses the testing phases in game development. In this chapter, optimization methods used during game development are analyzed and tests were carried out.

Chapter III: Security Issues in Game Development. The third chapter of the master's thesis conducts analysis and research on security issues during game development. In this chapter, data within the game engine is encrypted using various encryption methods.

At the end of the dissertation, conclusions and suggestions, as well as a list of used literature, are provided.

# Mündəricat

# İXTİSARLARIN SİYAHISI

VFX Visual Effects – *Visual Effektlər*

BSP Binary Space Partitoning – *İkili Məkanın Bölünməsi*

AI Artificial İntelligence – *Süni İntellekt*

BT Behavior Tree – *Davranış Ağacı*

CS Crowd System– *Toplu Sistem*

AIC Artificial İntelligence Controller – *Süni İntellekt Kontrolleri*

PP Post Process– *Post Proses*

PPV Post Process Volume– *Post Proses Həcmi*

MFA Multi-Factor Authentication– *Çox faktorlu təsdiqləmədə*

SM Substrance Material– *Maddə Materialı(çoxqat material)*

# GİRİŞ

**Mövzunun aktuallığı.** Vizuallaşdırma və oyun proqramlaşdırırlması simulyasiya proqramlarının hazırlanmasında istifadə olunan əhəmiyyətli bir vasitə və tədqiqat sahəsidir. Nəticəsində alınan dəyərlərin real həyatda olacaq dəyərlərə yaxın olması və ya kiçik xəta payı olması üçün həm tətbiq daxilində olan texnologiyalar, həm də həmin tətbiqlərin arxa planında işləyən proqramlaşdırma dilləri bunlara təsir edən mühüm amillərdir.

Oyun proqramlaşdırması görsəl simulyasiya proqramların inkişafına böyük töhfə vermişdir. Real həyatdakı fiziki qanunların oyun mühərrikləri daxilində təsvir edilməsi proqramlaşdırma sahəsi ilə yanaşı kompüterin aparat təminatı ilə əlaqəli digər texnologiyaların inkişafına yol verir. Vizual olmayan simulyasıiya proqramlarından fərqli olaraq, həm vizual olaraq səhnələrin təsviri, həm də realistik dəyərlərin alınmasında istifadə olunan bu proqramlar müasir aparat və proqram təminatlarından istifadə edir. Vizual olaraq səhnələrin bu mühərriklər daxilində hazırlanması paralelində bir çox yeni sahələrin təkmilləşdirilməsinə səbəb olur ki, bunlardan başlıcası da oyun proqramlaşdırması sahəsidir.

Oyun proqramlaşdırmasında istifadə olunan texnologiyalar yalnız bu sahədə deyil, virtual produksiyalar, kino sektoru, simulyasiya kimi sahələrin müraciət etdiyi vasitə halına gəlmişdir, çünki bu üsullar ənənəvi proqramların hazırladığı məhsullardan realistik görünüşü və mexanikası ilə fərqlənir.

Oyun proqramlaşdırmasının inkişafı daxilində istifadə olunan bir sıra sahələr, modelyerlik, vizual effektlər, animator kimi sahələrin inkişafına da yol açmşdır. Hər kiber mühitdə olduğu kimi bu mühitin təhlükəsizliyini təmin etmək üçün istifadə olunan texnologiyalar, istər oyun mühərrikləri daxilindən, istərsə də serverlər tərəfindən müxtəlif şifrələmə metodları ilə qorunur. Günümüzdə istifadə olunan Metahuman kimi texnologiyaların birbaşa olaraq oyun mühərrikləri(Unreal Engine) daxilinə asanlıqla adaptasiya edilmısi, anında mibil telefonlar vasitısilə bu modellərə animasiyaların

verilməsi, modellərin yaradılmasında olan bir çox qavramın bərabərində dəyişilməsinə yönələcək hərəkətlərə yol açmışdır. Vizuallaşdırmada istifadə olunan modellərin fərqli tətbiqlərdə hazırlanmasında istifadə olunan texnologiyanın zamanla oyun mühərrikləri daxilində və ya paralelində işləyən proqramlar halına çevrilməsi optimizasiya məsələlərinin uzun və problemli mərhələ olmaqdan qurtulmasına səbəb olur.

Oyun proqramlaşdırmasının kompüterin aparat təminatı üzərində olan təsirləri də qaçınılmaz olmuşdur ki, bunları prosessor və videokartlar üzərində açıqca görə bilərik. Videokartlarda istifadə olunan texnologiyalar oyun proqramlaşdırmasında istifadə olunan texnologiyaların sərhədlərini müəyyən etmişdir.

COVID-19 pandemiyasından sonra oyun sənayesinin böyüməyinin nəticəsində bu sahəyə olan kiberhücumlarında sayı uyğun olaraq artmışdır. Bunun nəticəsi kimi bu hücumlardan qorunma texnikaları da paralelində inkişaf etmişdir.

**Tədqiqatın məqsəd və vəzifələri.** Tədqiqatın məqsədi oyun proqramlaşdırmasında istifadə edilən oyun mühərrikləri daxilində olan texnologiyalardan istifadə edərək, realistik səhnələrin yaradılması, simulyasiya proqramlarının hazırlanması zamanı realistik mexanikanın yaradılmasını analiz edərək optimal həllər tapmaqdır. Bu məqsədə nail olmaq üçün aşağıdakı məsələlər qarşıya qoyulmuşdur:

- Oyun proqramlaşdırmasının tarixini analiz edərək, texnologiyanın gələcəkdəki vəziyyəti haqqında olan təxminlərə əsasən, istifadə edilən texnologiyaların aktuallığını da nəzərə alaraq öndəgedən oyun mühərriklərindən istifadə;
- Unreal Engine oyun mühərriyi vasitəsilə vizual səhnə, oyunlar, simulyasiya proqramlarının hazırlanması zamanı istifadə olunan texnologiyaların, və istifadə olunma qaydalarının analizi;
- Vizual effektlərin hazırlanmasında istifadə olunan texnologiyaların analizi;
- Oyun mühərriklərində süni intellekt texnologiyasının işləmə prinsiplərinin analizi;
- Oyun mühərriklərində mümkün optimizasiya həllərinin və optimizasiya üçün istifadə olunan yeni texnologiyaların analizi;

– Oyun sənayesində server və klient tərəflərinə olan hücumların araşdırılması;

– Oyun proqramlaşdırmasında

**Tədqiqatın predmeti və obyekti.** Tədqiqatın obyekti kibertəhlükəsizliyini təmin etmək məqsədilə oyun proqramlaşdırmasında şifrələmə və deşifrələmə metodlarından istifadədir. Tədqiqatın predmeti oyun proqramlaşdırmasına əsaslanan simulyasiya proqramlarının hazırlanması üsullarıdır.

**Tədqiqat metodları.** Tədqiqatda oyun mühərrikləri və proqramlaşdırma dilləri vasitəsilə oyunların, simulyasiya proqramlarının hazırlanması zamanı oyun mühərriklərində istifadə edilən texnologiyalardan analiz məqsədilə istifadə edilmişdir.

**Elmi yeniliyin elementləri.** Tədqiqatın elmi yenilikləri aşağıda sadalanmışdır:

– LOD texnologiyasını əvəzləyən Nanite texnologiyasını əsasında, səhnələrin daha optimal işlənməsinin səbəbləri araşdırılmıış və analiz edilmişdir;

– Unreal Engine mühərriyində süni intellekt texnologiyasının simulyasiyalara tətbiqində olan Crowd System, AI Controller, AI Perception, Visibility kimi texnologiyalar üzərində analiz apararaq, alınan nəticələr əsasında sənayenin müxtəlif yerlərində tətbiq edilməsinə şərait yaranmışdır.

Eksperimentlər Lumen, Nanite, C++, Post Process, LOD Systems, vizual effektlər, AI Crowd System, Behavior Tree texnologiyaları üzərində aparılmışdır. Cascade, Niagara, AI Perception, AI Visibility, səhnə işıqlandırmasında real fizikada istifadə olunan həllər vasitələr kimi istifadə edilmişdir.

**Praktiki həll.** Əldə edilən nəticələrin praktiki əhəmiyyəti ondan ibarətdir ki, Unreal Engine mühərriyində olan Blueprint texnologiyası ilə birlikdə oyun proqramlaşdırmasında C++ üzərində yazılan koddan qurtularaq daha sadə anlaşılan sistem üzərində vizual effekt, 3D modellər, səhnənin işıqlandırması, render əməliyyatının aparılması, Lumen və Post Process kimi texnologiyalar test edilmiş, bu texnologiyaların üstünlükləri və çatışmamazlıqları öyrənilərək ən optimal həllərin seçilməsinə yol yaratmışdır. Bu üsullarla birlikdə texnologiyanın kibertəhlükəsizliyini təmin etmək üçün

müxtəlif kibertəhlükəsizlik alətlərinin tətbiqi bu sistemlərin təhlükəsizliyində istifadə oluna bilər.

**Nəticələrin aprobasiyası.** Dissertasiyanın əsas elmi-nəzəri və praktik nəticələri AMEA İnformasiya Texnologiyaları İnstitutunun və Azərbaycan Texniki Universitetinin seminarlarda aprobasiyadan keçmişdir.

# I FƏSİL. KOMPÜTER OYUNLARININ İNKİŞAFI

## 1.1. Kompüter oyunlarının inkişaf tarixi

Oyun insanlıq tarixində, onun yaşamı boyunca gündəlik həyatının sıxıcı, yorucu hissəsinə rəng qatacağı bir fəaliyyəti olmuşdur. Baxmayaraq ki, sənayeləşmə, şəhərləşmə və texnologiyanın inkişafı ilə həyatımızın bir hissəsinə əyləncə olaraq daxil olmuş bu oyunlar, insanlıq tarixi ilə müqayisədə çox kiçik bir hissəni özündə ehtiva edir. Çünki, qədim dövrlərdə insanlar təbiətdə yaşam üçün səy göstərdiyi bir zamanda oyuna yer ayırması absurd bir məsələ idi. Günümüzdə isə bu fəaliyyət artıq bir çoxumuzun gündəlik həyatında ayrılmaz və əvəzolunmaz bir paya sahibdir.

20-ci əsrin ortalarına doğru dünya üzərində gedən proseslərin texnologiyaya, əsasən də kompüterlərin inkişafına göstərdiyi təsirin nəticəsi kimi, onunla əlaqəli bir çox yeni texnologiyaların toxumları da səpilmiş oldu. Bu texnologiyalardan biri də, irəlidə dünyada kiçikdən-böyüyə, əyləncə sektorundan hərbi və elmi araşdırmalarda istifadə ediləcək olan kompüter oyunları olacaqdı. Kompüter oyunlarını yalnız əyləncə məqsədi ilə istifadə etdiyimiz, uşaqların sevimlisi olan bir vasitə deyil. 20-ci əsrin 50-ci illərindən başlayaraq oyun olaraq adlandıra biləcəyimiz ilk oyunlar hazırlanmağa başlanmışdı. Təbii ki, həmin dövrlərdə 2-ci Dünya Müharibəsindən sonra bir çox qrafiki simulyasiyalar olunmağa başlansa da bu təkmilləşmələr gün üzünə çıxmırdı. 1952-ci ildə ilk dəfə A.S. Douglas "OXO"( Şək. 1.1) oynunu hazırladı. Bəli, bu oyun bizim uşaqlığımızda dəftərlərdə, öz parta yoldaşımızla, oynadığımız oyundur. Bu oyunun qrafiki olaraq Cambridge University-də EDSAC kompüterində hazırlanması bu ilə təsadüf etdi və bu oyun ilk video oyunlardan hesab olunur. Lakin, ilk interaktiv kompüter oyunu 1962-ci ildə MİT-də Steve Russel, Martin Greatz və Wayne Wiitanen tərəfindən təkmilləşdirilən "Specawar" adlı oyun olmuşdur. Kompüter oyunlarının inkişaf tarixinə nəzər yetirsək ilk başda simulyasiyalar üzərində hazırlanan "oyunlar" ilk kompüterlər və kontrollerlər tərəfindən idarə olunurdu. Bu oyunlar qrafiki cəhətdən çox da təkmil deyildi. Zamanın texnologiyasına bağlı olaraq daha optimal yollar fikirləşən mühəndislər əsasən oynanış

mexanikasına fikir verirdilər. Mexanikada olan sadəlik, oynanış hissiyatı və bunun gətirdiyi davamlıllıq, qrafiki əksikliyi gözardı etməyə imkan yaradırdı.

Şək. 1.1 İlk yaradılan OXO oynunun təsviri (A S Douglas, 1954)

Təbii ki, ilk dövrlərdə oyunların hazırlanmasında istifadə olunan heç bir oyun mühərriyindən söz gedə bilməz. Spacewar-ın hazırlanması üçün Assembly dilindən istifadə edilmişdir. Assembly kompüter proqramlanmasının bir çox hissəsində geniş istifadə olunan aşaağı səviyyə bir dil idi. PDP-1 kompüterində hazırlanan bu oyun Assembly üzərində hazırlanmasından qaynaqlanan optimizasiyaya sahib idi. Bura qədər olan hissəni kompüter oyunlarının gerçək yüksəlişi hesab etmək olmaz. Çünki bu mərhələdə hələ oyun firmaları və sektorun təməlləri hələ yeni atılır, fikirlər irəli sürülürdü. Ən əsası isə bunun üçün komandalar formalaşdırıldı. Proqramlaşdırma dillərinin artması ilə bərabər texnologiyanın bu hissəsini anlamağa və inkişaf etdirməyə çalışan bir çox fərdi şəxs də vardı. 1970-ci ildə əsas hesab olunan yüksəlişin təməlləri atıldı. 1972-ci ildə "Atari" tərəfindən hazırlanan "Pong"( Şək. 1.2.) oyunu nailiyyəti də bərabərində gətirdi. Baxmayaraq ki, həmin zamanlarda bir çox evdə kompüter yox idi, konsollar üzərindən oynanılan bu oyunlara böyük maraq var idi.


Şək. 1.2. Atari-nin hazırladığı Pong oyunu (Allan Alcorn, 1972)

1980-ci illərə doğru oyun sənayesinin yüksəlişinin tam şəkildə özünü göstərdiyi illər olmuşdur. Bu dövrü "Qızıl dövr" deyə adlandırırlar. Məşhur oyunların bir neçəsinin yaranması bu dövrə təsadüf edib. "Mario"( Şək. 1.3.), "Zelda", "Metroid" kimi məşhur seriyalar bu dövrdə yaranmışdır. Qızıl dövrdən etibarən hazırlanan oyunlar 3D qrafiklə, kompleks oyun mexanikası, multimediya və vizual effektlər kimi bir sıra texnologiya və yeniliklərin əlavə edilməsi ilə birlikdə daha cəlbedici hala gəlməyə başlamışdı. Əsasən bu oyunların yüksəlişə keçməsinin əsas səbəblərindən başlıcası, oynanış mexanikasının(gameplay) daha mürəkkəbləşməsi və 2D qrafika ilə birlikdə personaj və səhnələrin tərtibində olan realistikliyə doğru gedən inkişaf idi. 90-ci illərin sonlarına gəldiyimizdə bir çox oyunların hazırlanmasında artıq 2D qrafikadan 3D qrafikaya və Açıq Dünya(Open World) xəritəli oyunlara keçid edilməyə başlanmışdı. Verəcəyimiz bir sıra nümunələr ilə oyunların həmin dövrdə necə bir rəqabətlə, hansı istiqamətlərdə yüksəlişə keçməsini izah etmiş olacağıq. Zamanın iki böyük oyun firmalarından olan "Atari" və "Rockstar" öz məşhur seriyalarının hazırlanmasına başlayanda tarixlər 90-cı illərin sonunu göstərirdi.

Şək. 1.3. Mario oyunundan təsvir (Shigeru Miyamoto, 1981)

1999-ci ildə "Atari" və "Reflection" birlikdəliyi ilə "Driver"( Şək. 1.4.) oyunun ilk seriyası təqdim olundu. Bu oyun "Açıq dünya" oyun olmağı ilə yanaşı, qrafiki cəhətdən də zamanın irəlisində idi. Baxmayaraq ki, "Rockstar"ın tanıtmış olduğu "GTA"(Şəkil 6) oyunun birinci seriyasının çıxışı həmin ilə təsadüf edirdi, bu oyun quş baxışı oynanılan və "Driver ilə müqayisə edilməyəcək dərəcədə qrafiki əksiklikləri olan bir oyun idi.

Şək. 1.4. Driver oyunundan bir an (Allister Brimble, 1999)

90-ci illərin sonuna qədər hazırlanmış oyunların böyük bir hissəsinin oxşar xüsusiyyəti, onların müəyyən bir hekayə anlatmaması idi. Bir sıra oyunlar zamanın texnologiyasıı ilə əlaqəli olaraq hekayəli oyunlar deyil, ani oynanıla biləcək, rəqabətcil və ya müəyyən tapşırıqları olan oyunların hazırlanması üstün tutulurdu. Çünki belə oyunları konsollarda və ya müəyyən məkanlarda olan kompüterlərdə müəyyən zaman daxilində dəfələrlə oynanıla bilinəcək və bir dəfə oynanıldıqdan sonra yenidən maraq yaradacaq xarakterə malik olması idi. Lakin belə bir ortamda hekayə xarakterli oyunların yüksəlişi qaçınılmaz idi. Buna təsir edən yan amillər də var idi. Bunlardan ən başlıcası kompüterlərin artıq hər evdə olacaq qədər yayğın hala gəlməsi idi. Kompüterlərdə istifadə olunan komponentlərin(RAM, ROM, CPU, GPU, Ana plata və.s) parametrlərinin artmasının da buna təsiri var idi. Oyunlarda istifadə olunan texnologiyalar onun yaddaş həcminin artması ilə nəticələnir, qrafik göstəriricilərin artması isə birbaşa aparat təminatına bağlı olaraq irəliləyirdi. “Driver” oyununun ilk seriyasının oynayanlar tərəfindən sevilməsinin əsas səbəblərindən biri 3D qrafikalı olmasıdır. Real maşın modelləri, şəhər məkanları, personajlara hazırlanmış animasiyalar yeniliklərdən bir neçəsi idi. Əsasən maşın mexanikası və oyun hissiyatı zamanın yeniliklərindən biri olmaqla yanaşı, realistikliyə doğru gedən yolun təməllərini atmış oldu. Həmin zamanın bir digər yeniliyi də Açıq Dünya(Open World) xəritə ilə hazırlanmış olması idi. Əvvəlki oyunlara nisbətdə, oyunçulara sərbəstlik tanıyan ilk oyunlardan biri olmuşdur. Müəyyən tapşırıqları yerinə yetirməkdən əlavə olaraq xəritənin istənilən yerinə oyunçular kəşf edə bilirdi. Bu oyun dünyasını kökündən dəyişəcək yeniliklərdən biri kimi sayıla bilər. Çünki oyunçu azadlığı deyə adlandıra biləcəyimiz bu məsələdə ilk addımlar atılırdı. Bu tip oyunlarda olan

xəritənin oyunçu tərəfindən kəşfi, müəyyən ərazilərdə təyin edilmiş tapşırıqlar, tərtibatçılar tərəfindən yerləşdirilmiş yüzlərcə sirli məqamlar artıq oyun dünyasının ayrılmaz bir parçasına çevriləcəkdi. İstifadə olunan bir digər yeniliklərdən biri də süni intellekt idi. Təbii ki, tək oyunçulu oyunlarda, istifadə olunan rəqiblərə müəyyən dərəcədə əmrlər verilməsi əvvəldən var idi. Lakin süni intellektin əmrlərinin genişləndirilməsi, müəyyən şəraitlərdə fərqli qərarlar verərək oyundakı axışı dəyişə bilməsi bu illərə təsadüf edir. Məsələn, oyundakı bir polisin cinayətkar təqibi buna nümunədir. Bu texnologiyanın sərhədlərinin günümüzdə hansı yerlərə çatdığına isə bir azdan toxunacağıq. NPC(Non-Player Character) deyə adlandıracağımız bu süni intellektlərdən personajlarda, maşınlarda, öz personajımıza birlikdəlik tanıyacaq bir digər personajın istifadəsində, xəritədə olan heyvanlarda və s. kimi yerlərdə rastlamaq mümkündür. 1997-ci ildə GTA seriyasının ilk oyununun 3D qrafika ilə çıxmamasına, qarşısında Ubisoft kimi bir rəqib olmasına baxmayaraq, 2001-ci ildə təqdim olunan GTA 3, 2002-ci ildə GTA: Vice City, 2004-cü ildə GTA: San Andreas, özünün bu janrda rəqibtanımaz hala gətirməsi ilə nəticələndi. Bu yerdə vacib qeyd olunmalı bir digər məsələ, Atari-nin Rockstar ilə böyük rəqabətə girərək Driver oynunun bir digər seriyasını çıxarmaq üçün böyük pullar və oyunçulara vədlər verərək bunları yerinə gətirə bilməməsi ilə hüsrana uğradı. Mövzuya digər bir tərəfdən baxsaq Atari bu rəqabətdə yalnız maddi gücünü səhv idarə edərək itirmədi, bizim də bəhs edəcəyimiz oyun texnologiyaları və yaradılma mərhələsində olan oyunun düzgün formada test mərhələlərinin idarə edilməməsi oldu. Məsələnin üstünü örtmək üçün Atari şirkəti bir neçə böyük media qurumunu pulla satın alaraq oyunu yüksək qiymətləndirmə ilə Rockstar-dan üstün göstərməyə çalışsa da Driver seriyası ilk oyunu kimi bir daha bu rəqabətə girə bilmədi. Kompüter oyunları ilə yanaşı 20-ci əsrin sonuna doğru mobil telefonların gündəlik istifadədə yayğınlaşması və çoxməqsədli forma alması ilə birlikdə mobil oyunların inkişafı qaçınılmaz oldu. İlk başlarda mini konsollar şəklində olan bu cihazlarda bir və ya bir neçə oyun var idi. İlk mobil xarakterli oyunun tarixi tam dəqiq olaraq bilinməsə də, Hewlett Packard-ın HP 95LX(Şək.1.5.) daşınabilən

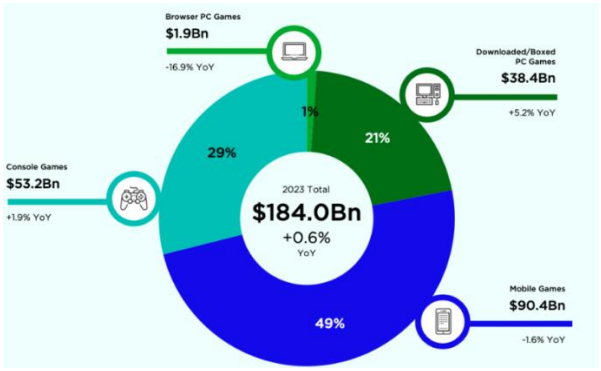
kompüterində hazırlanmış “Tetris” oyunu sayılır. Bu tip daşınabilən mini kompüter və ya konsol deyə adlandıracağımız cihazlar mobil oyunların hazırlanmasında öncü texnologiyalar sayıla bilər. Çünki belə cihazların istifadəsi oyun məqsədli olduğu üçün ilk telefonlardan böyük olan ekran və məqsədyönlüyü oynanılan oyunların inkişafına daha aşkar yol verdi.

Şək.1.5. HP 95LX daşınabilən kompüterində oynanılan Tetris oyunu (Hewlett-Packard, 1991)

Bu tendensiya uğurlu bir siyasət izləyərək oyun texnologiyasının inkişafında böyük rol oynadı, hansı ki, hal hazırda da müəyyən firmalar öz konsollarını və konsollar üçün oyunlarını hazırlayaraq bu inkişafın davamını gətirirlər. Mobil oyunların inkişafının isə 90-ci illərin sonuna doğru başladı. 1997-ci ildə Nokia 6110 telefonunun çıxışı ilər birlikdə həm telefon bazarının inkişafı onunla paralel, mobil oyunların inkişafının təməlləri atılmış oldu. Bu telefonda olan oyun isə hər birimizin həyatında ən az bir dəfə gördüyü və ya oynadığı “Snake” oyunu olmuşdur. Əsasən 2006-ci ilə, yəni smartfonların yüksəlişinə qədər olan mərhələdə istehsal olunmuş telefonlara yüklənilə bilən və ya daxilində gələn oyunlar qrafik resursları məhdud, yaddaş həcmi kiçik, saylı mərhələli və proqramlaşdırma dillərində hazırlanmış oyunlar idi. 2007-ci ildən etibarən smartfonların istehsalının kütləvi xarakter alması ilə birlikdə həmin telefonların çoxfunksionallığını nəzərə alsaq, oyunların bu zamandan sonra necə bir yüksəlişə keçəcəyini təxmin etmək çox da çətin məsələ olmamalıdır. 2007-2008-ci illərdən başlayaraq Android, 2007-ci ildən Apple istehsalı olan Iphone cihazlarında olan IOS əməliyyat sistemli telefonlar üçün, günümüzə qədər saysız oyun hazırlanmış və təkmilləşdirilmişdir. İlk başlarda bu oyunların məzmununa baxsaq

əsasən zaman və çıxış tarixlərinə görə heç də küçümsənməyəcək dərəcədə qrafiki resursları və oynanış hissi axıcılığı ilə seçilən oyunlar olduğunu görərik. Eyni zamanda 90-cı illərin ortalarından başlayaraq Sony firmasının Playstation, 2000-ci ildən etibarən isə Microsoft-un Xbox kimi oyun konsollarının çıxışı oyunların çoxşaxəli olaraq fərqli istiqamətlərdə təkmilləşməsinə gətirəcəkdi. Günümüzdə həm kompüter, həm telefon, həm də konsol oyunları maddi və texnoloji olaraq inanılmaz bir yol qət edib. Təbii ki, bunda texnologiyanın inkişafının, yəni kompüterlərin təkmilləşməsinin, smartfonların böyük ölçüdə ildən-ilə yeni texnologiyalar ilə təqdim olunmasının, prosessorlar və videokartlar kimi texnologiyaların gəldiyi son nöqtənin böyük dəstəyi var. Mobil oyunların inkişaf tarixinin günümüzə yansımasına nəzər yetirsək, belə oyunların hazırlanmasında həm qrup, həm də tək olaraq inkişaf etdirmək mümkün olduğundan bu tipli oyunlar bazarda daha sürətli və tez hazırlanır. Kompüter oyunların hazırlanması zamanı böyük ölçüdə müəyyən edilmiş bir neçə komanda işlədiyindən, platforma olaraq, konsol, kompüter kimi bazarlara hazırlandığından onlarda olan keyfiyyət də digərlərindən seçilməlidir. Mobil oyunlarda isə qrafiki keyfiyyətlərin məhdud olmasından dolayı, əsasən oyunun mexanikasının seçilməsi ilə üstün tutulur. 2010-cu ildən günümüzə kimi hazırlanmış mobil oyunların məzmununa nəzər yetirsək əsasən “Endless Runner” dediyimiz, platforma üzərində “sonsuz” qaçış olan oyunlar üstünlük təşkil edir. Burada digər önəmli məsələ oyunçu psixologiyasını düzgün anlamaqdır. Mobil oyunların əsas məqsədi müəyyən zaman aralığında(qısa və orta) oyunçunu sıxmadan axıcı keyfiyyət verməkdir. Lakin son zamanlarda kompüter oyunlarının telefonlara inteqrasiyasını nəzərə alsaq küçümsənməyəcək dərəcədə kütləsinin olduğunu görə bilərik(məsələn PUBG Mobile, COD, League of Legends, Fortnite). Lakin bu tip oyunların hazırlanması o qədər də asan məsələ deyil. Kompüter oyunlarının günümüzdə gəldiyi nöqtə isə həm mexaniki, həm də qrafiki cəhətdən reallıqdan seçilməsi güc olub. 2023-cü ilin statistikasına görə oyun sənayesinin dünya üzərində ümumi büdcəsi 500 milyarda yaxındır(Şək.1.6.). Hazırlanması zamanı tələbata uyğun olaraq platforma seçimi, məzmunun

müəyyənləşməsi, əsas problem olaraq göstərilən, oyunun çıxış tarixinin müəyyən olunması və s. kimi bir çox amil bu sektora yatırılan maddi vəsaitə təsir edir. Büdcənin böyük hissəsində mobil oyunlar başçılıq edir. Bu da mobil oyunlara tərtibatçı və şirkətlər tərəfindən olan marağın artması deməkdir.




Şək.1.6. Oyun gəlirləri statistikası (Taleh Məmmədli, 2023)

## 1.2. Oyun mühərrikləri və mövcud texnologiyalar

İlk zamanlarda oyunların hazırlanmasında proqramlaşdırma dillərindən istifadə olunurdu. Bu metod günümüzdə də istifadə olunur. Lakin bunun yanı sıra oyun mühərriklərindən də istifadə olunur. Proqramlaşdırma dillərindən oyunların mexanikasının hazırlanmasında, daha optimal olmasında istifadə oluna bilər. Lakin hazırlanan oyunun zaman aralığı, qrafiki olaraq realistik olması və bir çox komponentin eyni anda bir proqram üzərindən istifadə olunmasında oyun mühərriklərindən istifadə olunur. Günümüzdə çox sayda mühərrikdən istifadə olunur. Hazırlanan oyunun janrından, çıxacağı platformadan asılı olaraq mühərriklər fərqlilik göstərə bilər. Mühərriiklərin hazırlanmasında müxtəlif proqramlaşdırma dillərindən istifadə olunur. Unreal Engine-də C++, Unity-də C#, RAGE-də C++, CryEngine-də C++, C#, Lua dillərindən istifadə olunub. Jani H, Game Development with Unreal Engine 4(2020, 10p) məqaləsində Unreal Engine daxilində istifadə olunan Event-lər, Macros-lar, animasiya daxilində State Machine daxilində blueprint üzərindən proqramlamadan bəhs etmişdir. Mühərriklər daxilində zamandan qənaət etmək, vizual olaraq optimizasiya və bərabərində realistikliyə gedəcək bir texnologiya kimi mühərriklərdən istifadə qaçınılmazdır. Lakin bunun yanı sıra

mühərriklərin arxasında istifadə olunan proqramlaşdırma dillərində və bunun təkmilləşməsində çalışan tərtibatçıların böyük önəmi var. Mühərrikləri önəmli edən digər məsələ, onların açıq mənbə və ya əksinə olmasıdır. Günümüzdə oyun tərtibatçılarının üz tutduğu mühərriklərin başında Unity və Unreal Engine durur. Amma bu mühərriklərin yenilənməsində məsuliyyətli şirkətlər qərarlarının doğruluğu və ya yanlışlığı onlardan istifadə edən tərtibatçıların fərqli mühərriklərə üz tutmağına səbəb olur. Pelechano G, Núria, Andújar G, Carlos A, Synthesising character animation for real time crowd simulation systems in Unreal Engine(Pelechano G, 2018,) məqaləsində günümüzdə istifadə edilən açıq mənbə oyun mühərriklərindən və onların versiyalarından bəhs edirdi. Unity-in haqq sahibliyinin tərtibatçılardan illik abunə haqqı, hazırlanan oyunların satışından gələn pulun müəyyən faizinin şirkət tərəfindən kəsintisi kimi amillərdən dolayı tərtibatçılar uyğun açıq mənbə proqramlarına üz tutmalı olurlar. Açıq mənbə oyun proqramlarının digər üstün tərəfləri böyük şirkətlərin işə alım zamanı çox seçimi olmasıdır. Əlavə olaraq şirkətlərin öz mühərriklərinin yaratmasında mənfi cəhəti, oyun tərtibatçılarının işə qəbulundan sonra həmin mühərriyə uyğunlaşması, sahib olduğu texnologiyaların işləmə prinsipinin öyrənməsinin aldığı vaxt itkisidir. Bunun yanında açıq mənbə mühərriklərdə belə problemlərin olmaması ilə yanaşı, mühərrik daxilində olan səhvlərin kütləvi bildirilməsindən dolayı səhvlərin daha tez aradan qaldırılmasına səbəb olur.

Julia V, Comparison between Vizard VR Toolkit and Unreal Engine 4 as platforms for virtual experiments in pedestrian dynamics using the Oculus Rift(Julia V, 2015) məqaləsində Unreal Engine üzərindən VR texnologiyasının istifadə vurğulamışdır.

Oyunların janrları və hazırlanan platformalara görə mühərriklər əvvəlcədən seçilir. Buna əsasən şirkətlər öz konseptlərini əvvəlcədən seçir və ona əsasən bir yol xəritəsi izləyir. 2D mobil oyunların hazırlanmasında Unity-dən, 3D mobil, kompüter və konsol oyunlarının hazırlanmasında açıq mənbə olaraq Unreal Engine-dən istifadə olunur. Bunun yanı sıra

Rockstar RAGE-dən, Crytek CryEngine-dən, 2 və 3D oyunların hazırlanmasında isə Godot-dan istifadə edilir.

Bu elmi işdə danışacağımız olan Unreal Engine, 90-cı illərdən təkmilləşdirilməyə başlanmışdır. Təbii ki, interfeys olaraq ilk illərdə indikindən tamamilə fərqlənən bu proqram ağırlıqlı olaraq mexanikanın kodlaşdırmasında C++ dilindən istifadə olunurdu. 2014-cü ildən etibarən, yəni Unreal Engine 4 versiyası istifadəyə verildiyi gündən isə "Blueprint" texnologiyasından istifadə olunur. Tri Hai Ha, Game development with Unreal Engine(Tri Hai, 2022) məqaləsində Blueprint texnologiyasının daha sürətli və vizual olduğu üçün C++ dilindən bu cəhətlərinə görə üstünlük qurduğunu düşündüyünü deyirdi. Bu texnologiya heç bir proqramlaşdırma dili bilmədən oyun hazırlamaq imkanı tanıyan bir yenilik idi. Təbii ki, bunla da yekunlanmır, günümüzdə istifadə olunan bir sıra proqramlarda da buna bənzər texnologiyadan istifadə olunur. Hansı ki, oyun təkmilləşdirilməsi sırasında bir proqramlaşdırma dilinə nisbətdə çox da az hazırlanma vaxtı tələb edir, başqa sözlə artıq kod yazma əziyyətindən bizi qurtarır. Reece A, Implementing Reinforcement Learning in Unreal Engine 4 with Blueprint məqaləsində Blueprint(Reece A , 2017) üzərində massiv və riyazi ifadələrinin şəkilli izahını istifadə etmişdir. Hər biri Blueprint skriptindən ibarət olan bu hissəciklər arxasında C++ üzərində yazılmış koda bağlı olub, hər hansı bir əməliyyatı sətirlərlə kod yazmadan, bir skriptlə həmin əməliyyatı icra etməyə imkan tanıyır. Lakin bu texnologiyanın üstünlükləri və çatışmamazlıqları var. Az öncə qeyd etdiyimiz kodlaşdırma bilməyən birinin bundan istifadəsi, zamandan qənaət, Blueprint olaraq qurulmuş sistemin rahatlıqla C++ koduna inteqrasiyası, təməl olaraq işlədilən ardıcıllıqların bir neçə Blueprint ilə bitirilməsi və Unreal Engine topluluğunun Blueprint fokuslu olması, bunun da çətinliklər olduğu və ya öyrənmə materialları zamanı rahatlıqla bir çox məzmunun tapılması üstünlüyü olaraq qeyd oluna bilər. Roope P, Virtual Reality Multiplayer in Unreal Engine 5 with C++( Roope P, 2022) məqaləsində C++ üzərindən optimizasiya qaydalarından bəhs etmişdir. Hesablama tələb edən yerlərdə Blueprint-dən istifadə, müəyyən siniflər daxilində yazılmış

olan Blueprint-lərə müraciətin əlçatanlığın çətinliyi, üst versiyada yazılmış olan Blueprint skriptinin aşağı səviyyə versiyalarda hər zaman düzgün işləməməsi, bəzən tamamilə xəta verməsi, Blueprint-lərin öz öz aralarında "Behavior Tree" deyə də ifadə edə biləcəyimiz ierarxiyada xətalar çıxarması onun mənfi tərəfləri kimi göstərilə bilər. Lakin bu o demək deyil ki, C++ tamailə çıxış yoludur. Blueprint ilə C++ kodunun uyğun yerlərdə işlədilməsi üçün oyun qurulumu zamanı əvvəlcədən bunun təyin olunması lazımdır. Blueprint istifadəsində oyun performansının azalmasına səbəb olacaq hissələr əsasən hesablamaların olduğu kod hissələrinin skript üzərində yazılan yerləridir. Məsələn "Debug" hissələrində, oyun üzərində ölçüm və ya sayım olan hissələrdə C++ kodundan istifadə daha münasibdir. Lakin geri qalan hissələrdə isə Blueprint kodundan istifadə məqbul tutulur. Yəni peşəkar oyun proqramlanmasında Blueprint kodu və C++ kodu bərabər yazılması daha uyğundur. Performans dəyərləndirməsinə gəlməmişdən əvvəl son olaraq deyə bilərik ki, Blueprint üzərində aparılan təkmilləşdirmələr C++ kodunu əvəzləyəcək dərəcədə irəli səviyyədə olub, aradakı fərqi hiss etmək üçün böyük həcmli proyektlər üzərində işləmək lazımdır. Kiçik proyektlərdə isə bu fərq hiss olunmayacaq. Marcos R, Brenden S, Blueprint Visual Scripting for Unreal Engine 5(Marcos R, 2022) məqaləsində Unreal Engine daxilində riyazi ifadələrin blueprint üzərindən izahını vermişdir. Bu ifadələr müxtəlif yerlərdə müxtəlif məqsədlər üçün istifadə olunur.

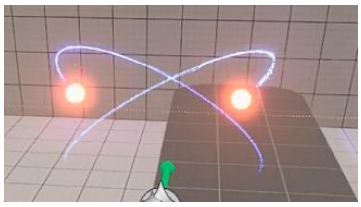
### 1.2.1. Vizual effekt(VFX)

Vizual effektlər oyunlarda, reklamlarda, kinolarda və.s kimi yerlərdə istifadə olunan və rəng qatan komponentdir. Hazırlanması üçün müxtəlif platformalardan istifadə olunan bu effektlər oyunlar üçün, mühərriklərin daxilində də hazırlana bilinir. Əlavə olaraq Houdini, Blender kimi tətbiqlərdə də 3D animasiya və VFX hazırlanmasında istifadə olunur. Bu kimi proqramlar daxilində olan əsas məsələlər "render", "shader" kimi aparat təminatı ilə birbaşa əlaqəli məsələlərdir. Unreal Engine daxilində "Cascade" və "Niagara" Hissəcik sistemindən(Particule System) istifadə olunur. Unreal Engine 5 versiyasından etibarən Niagara sistemi ağırlıqlı olaraq istifadə edilir. Əlavə olaraq realistiklik üçün Ninja Fluid

kimi “plugin”lər istifadə olunur. Vizual effektlər GPU daxilində render olunur. Render nədir? Səhnədə olan 3D modellər, effektlər işıqlandırma və başqa komponentlərin ekranda görüntülənməsindən cavabdehdir. Ying Wu, A New Exploration based on Unreal Engine4 Particle Effects of Unreal Engine in 3D Animation Scenes məqaləsində hissəciklərlə bulud effektinin hazırlanmasından bəhs edirdi ki, burada optimizasiyanın önəmi də çox mühüm məsələdir. Render və shader-lərin səhnədə vizuallaşdırılması qarışıq riyazi hesablama və səhnədəki yerində görünməsi üçün mühərrik daxilində olan tənzimləmələrlə bağlıdır. Lakin bunun yanı sıra çox sayda olmayan Vizual effekt CPU daxilində render həyata keçirilir. Particle System daxilində effektdən asılı olaraq hissəciklərin sayını artıra və ya azalda bilərik. Buna uyğun olaraq Unreal Engine mühərriyi bizə GPU-un işə salınması üçün bildiriş verəcək.

VFX ilə istənilən effekti hətta bir çox cisiimlərin toplu hərəkətini də simulyasiya etmək mümkündür. Bunlara quşların, balıqların, həşəratların sürü ilə hərəkətini nümunə göstərmək mümkündür.

Tədqiqat zamanı isə bir sıra riyazi modellərin simulyasiyası həyata keçirilmişdir ki, bunlardan ən başlıcası 3 cisim problemi olaraq riyaziyyatda bilinən cisimlərin öz kütlələrinin bir birlərini çəkməsi ilə gedəcəyi traektoriyanın nəmalumluğunu bildirir(Şək. 1.7.).



Şək. 1.7. Particle effect (Taleh Məmmədli, 2024)

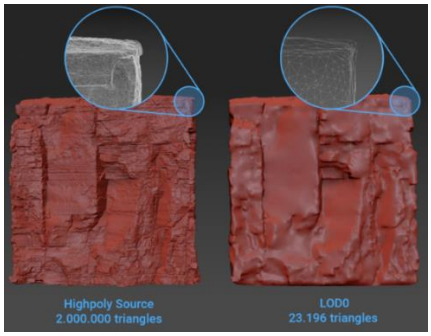
Bunun riyazi modeli çox qarışıq və təyi edilməsi çətin olduğundan 2 cisim ilə izah olunmuşdur. Əsasında da bu cisimlər hissəcik olaraq kütlə verilmiş və bir birlərinin ətrafında hərəkət etmişdir. Bu hərəkət zamanı cisim olaraq VFX daxilində göstərilmiş

hissəciklərə, İmpact Point, yəni təsir hissələri verilmiş, əlavə olaraq bu hissəciklər daxilində bir birlərinin ətrafında hərəkət etməsi üçün kütlə verilmişdir. Kütlələr təxmini olaraq verilsə də bir birlərinə yaxın kütlələr nəzərdə tutulmuşdur. Digər bir effekt isə Shock Wave adlandırılan Şok dalğası effektidir. Bu effekt daxilində də fizika hissəsində İmpact Point qeyd edilərək ona kütlə verilmişdir. Bunun nəticəsində isə effekt zamanı müəyyən hissədə olan obyektlərə bu dalğa təsir edir.

### 1.2.2. BSP, Geometriya. Polygon.

3D modelləmədə geometriyadan geniş şəkildə istifadə olunur. Əsasən modellərin hazırlanmasında oyun mühərriklərin daxilindən deyil, fərqli tətbiqlər, Blender, 3D Max kimi proqramlardan istifadə olunur. Əsasən modellərin hazırlanmasında nəzərə alınmalı məsələ "Polygon" sayısıdır. Polygon sayısı modellərin hazırlanmasında və formalaşdırılmasında istifadə olunur. Polygon sayısına əsasən modellər üzərində olan ayrıntılar daha aşkar olur. Lakin bu o demək deyil ki, istədiyimiz qədər polygon sayısını istədiyimiz qədər artıra bilərik.

Polygon sayısının artması(Şək. 1.8.) ilə əlaqəli render problemləri ortaya çıxır. Bunun yanı sıra optimizasiya məsələləri ortaya çıxır.




Şək. 1.8. Polygon sayısının dəyişməsi(Taleh Məmmədli, 2024)

Modeli hazırlayan zaman proqram daxilində polygon sayısına əsasən onun şəbəkəsinin(grid) sıxlıq dərəcəsini görə bilərik. Şəbəkə üzərində görünən hər bir x,y,z

nöqtələrinin birləşməsində istifadə olunan komponent Vertex(verteks) adlanır. Verteks nöqtələri birləşərək 3 bucaq əmələ gətirir, yəni triangles adlanır. Triangles polygon-un kənarları, verteks nöqtələrini birləşdirən cizgilərdir. Triangles-in səthini müəyyən etmiş olub, kənarların uzunluğu, dərəcəsini triangles-in forması müəyyən edir. Triangles polygon-un səthi triangles-in kənarları tərəfindən müəyyən olunan bir parça kimi başa düşülə bilər. Hər bir triangles-in bir normal vektoru var ki, bu vektorlar da triangles-in hansı tərəfə istiqamətləndiyini göstərir. Dedik ki, hər bir triangles-in bir normal vektoru var, bu normallar səthə perpendikulyar istiqamətlənir. Hər bir normalın səthə toxunduğu hissəyə perpendikulyar istiqamətlənmiş Tanget olur. Bunun yanı sıra hər bir Tanget-in normala toxunduğu nöqtəyə perpendikulyar olaraq binormal endirilir. Normal, tanget və binormal-ın praktiki istifadə olunduğu yerlər vardir. Jaakko S, Interactive Architectural Visualization Using Unreal Engine(2022, 15p) məqaləsində Unreal Engine üzərində obyektlərin UV xəritəsi göstərilmişdir. Zbrush, Blender kimi proqramlarda modellər hazırlayarkən bunları daha aydın şəkildə görmək mümkündür. Hər bir modelin bir neçə səviyyədə təyin olunmuş LOD(level of Detail) modeli olunur. Bu isə tamamilə optimizasiyaya yönəlmiş məsələdir. High-polygon bir modelin alt səviyyələrdə təyin olunmuş sıfırdan başlayaraq bir neçə dərəcədə LOD modeli olur. Aralarında fərq onların polygon və triangles sayılarıdır. Daha realistik görüntülər üçün high-polygon modellərdən istifadə olunur. Bunları əsasən reklam, film, vizuallaşdırma kimi ani səhnələrin təqdimatında daha realistik render almaq üçün istifadə edirlər. Oyun kimi daha optimizasiya tələb edən yerlərdə isə LOD(0,1,2,3 və.s) modellərdən istifadə olunur. High-polygon modelin 2 milyon polygonu olduğunu saysaq, həmin modelin LOD0-cı elementində 23 min polygona qədər bu dəyər enir. LOD1, LOD2 və digərlərində bu dəyər getdikcə daha da enir. Ona görə model hazırlayan zaman yerindən asılı olaraq 3D Artist(3D modelyer) bunların hər birini nəzərə almalıdır.

### 1.2.3. Süni İntellekt(Artificial Intelligence)

Süni intellekt oyunların müxtəlif hissələrində geniş istifadə olunur. Süni intellekt öz daxilində də müxtəlif texnologiyalardan istifadə edir. Unreal Engine daxilində istər C++ üzərindən, istərsə də Blueprint üzərindən müxtəlif süni intellekt texnologiyaları var. Əsasən Blueprint üzərində vizual olaraq bir çox tənzimləmələr üzərindən bunları qeyd edə bilərik.

### 1.2.4. Behavior Tree

Unreal Engine daxilində süni intellekt yaratmaq istədiyimiz zaman Pawn Class daxilindən personaj yaratdıqdan sonra sadə BP kodlarından istifadə edə bilərik. Lakin daha mürəkkəb AI hərəkətləri, özünü səhnədə necə aparmalı olduğunu, birdən çox yöndə hərəkətlərin gedişinin irəliləməsini idarə etmək üçün Behavior Tree istifadə olunur. Məsələn, hansı yerdə personajın hansı animasiyası icra olunsun kimi xüsusiyyətlər buradan idarə edilir. Behavior Tree daxilində "Root"( Şək.1.8.) başlanğıc "İnput" olaraq başa düşmək olar. Selector, Sequence və Simple Parallel hərəkətlərin hansı yöndə idarə olunmasını tənzimləyəcək və bunlar kompozitlər adlanır. Kompozitlər öz daxilində hərəkətin idarə edilməsində True və False olaraq dəyərlər çıxarır. Bu dəyərlərə əsasən kompozitlər daxilində olan tapşırıqlar icra edilir və ya tam əksi.


Şək.1.8. BT daxilində Root (Taleh Məmmdli, 2024)

Selector bir hərəkət ierarxiyası olaraq düşünülsə, bu ierarxiya altında icra olunan tapşırıqlar, yəni Task-lar qoşulur. Ferus Abu-Abed, Sergey Zh, New Game Artificial Intelligence Tools for Virtual Mine on Unreal Engine (2023) məqaləsində BT daxilində olan funksiyalardan bəhs edərkən soldan sağa doğru işləməsindən bəhs edirdi. Task

burada icra olunan hərəkətdir. Məsələn, animasiyanın işə düşməsi, səsin icrası, süni intellektin dayanması və s. kimi. Selector(Şək.1.9.) daxilində hazır şəkildə istifadə edə biləcəyimiz bir çox hadisə var. Əlavə olaraq C++ kodu olaraq və ya BP daxilində susmaya görə sıfırdan özümüz də Task- hazırlaya bilərik. Selector-a qoşulu olan Task-lar daxilində olan kodda, **FINISH EXECUTE** pini hər iki Task-a True dəyər versək, yalnız Selector-un sağında olan Task icra olunacaq. Amma hər iki **FINISH EXECUTE** pinində False dəyər versək, hər iki Task icra olunacaq. Çünki Selector ancaq True dəyərə görə kodu icra edir.

Sequence isə bu dediklərimiz əksinə icra olunur. Sequence üzərinə birləşdirilən Task-lar daxilində **FINISH EXECUTE** pinində True dəyər versək, hər iki Task icra olunacaq.

Simple Parallel(Şəkil 17) üzərində isə birdən çox hərəkətin birləşməsi də adlandırıla bilər. Simple Parallel üzərində olan bənövşəyi pində yalnız Task-lar olur. Digər tərəfdə isə kompozitlər Task birləşdirmək olar. Prioritet olaraq sol tərəfdəki pin həmişə birinci icra olunur. Əsas hissədə olan Task-ın dəyərindən asılı olmayaraq arxa planda olan çıxış həmişə icra olunur.


Şək.1.9. Simple Parallel pini (Taleh Məmmdli, 2024)

Yalnız süni intellektin yaradılmasında BT tək başına kifayət etmir. BT bir ierarxiyanın idarə edilməsində istifadə olunur. Muhammad M, Applied Artificial İntelligence in 3D-game (HYSTERİA) using UNREAL ENGİNE 4 məqaləsində Blueprint üzərində süni intellektin imkanlarının üstünlüyünü vurğulamışdır.

### 1.2.5. Crowd System

Unreal Engine mühərriyində botları idarə etmək üçün istifadə olunan bir neçə texnologiya var. Bunlardan AI Controller və Crowd System nümunə göstərmək olar. Lakin aralarında fərqlər vardır. Kütləvi botların idarə edilməsi üçün Crowd System-dən istifadə olunur. AI Controller botların idarə edilməsində istifadə olunsa da böyük kütlələrin idarə edilməsində istifadə olunması çox da üstün tutulmur. AI Controller tək istifadə olunan botların idarə edilməsində istifadə olunur. Tutaq ki, səhnədə tək başına gəzən, bir-birlərilə çox da əlaqədə olmayan botlar istifadə etmək istəsək, AI Controller Class vasitəsilə yaradılan sinifdən istifadə edərək botları yarada bilərik. Lakin bu botların sayını artıraraq, eyni yerdə istifadə etsək, onlara verilən tapşırığın icrası zamanı bir-birlərini gözləməyəcək, verilən kodun icrasını yerinə yetirmək üçün bir-birlərini qabaqlamağa çalışacaq. Burada isə ortaya yaxşı görüntü çıxmayacaq. Bunun yaşanmaması üçün Crowd System-dən istifadə olunur. Crowd System, DetourCrowdAIController sinifi ilə idarə olunur. Eden Li, Real Time Crowd Flow(2018, 8 p) məqaləsində obyektlər arasında botların gəzməsi üçün yerlərini təyin edən Volume-dən istifadəsinin şəkilli izahını vermişdir. Daxilində yazılan kodlar AI Controller ilə eyni olsa sinfin yaradılması zamanı arxasında işləyən kod böyük kütlələrin eyni zamanda, eyni yerdə istifadəsi üçün nəzərdə tutulub. Mərkəzə doğru, eyni anda yönəlməyə çalışan botlar mərkəzə doğru yönəldikcə aralarında olan məsafə azalsa da bir-birlərini kobud desək, tapdalamadan, gözləyərək mərkəzdə verimiş məsafəyə yönəlməyə çalışacaq. Əgər mərkəzə tez çata bilməyən bot olarsa arxada olan hər hansı yerdə öz yerini tutaraq dayanacaq. Bunu Shivang S, Bernard T, Helen C, AWideAreaMultiview Static Crowd Estimation System Using UAVand3DTraining Simulator(2018, 17 p) məqaləsində dron ilə yerdə gəzən kütləvi botların skanlamasını həyata keçirmişdir.

## 1.3. Oyunların yaradılması zamanı qrafik resurslar

Qrafik resurslar, oyunların yaradılması zamanı nəzərə alınması vacib olan ən mühüm məsələlərdəndir. Yaradılması və inkişaf prosesində ən çox təkmilləşdirilən

texnologiyalardan başlıcası qrafiki resurslara təsir edən texnologiyalardır. Optimizasiyaya təsir edən amillər ən çox yeniləmələrə getməli omuşdur..

### 1.3.1. Render

Render prosesi GPU daxilində baş verdiyindən, əlavə olaraq növündən asılı olaraq bu prosesin tezliyi, keyfiyyəti fərqlilik göstərə bilər. Render prosesi bir neçə mərhələdə icra olunur.

1. Səhnə Qrafiki. Səhnə açılan zaman səhnədə olan obyektlərin yeri, ölçüləri əvvəlcədən təyin olunmuş formada icra olunur.

2. Geometriyanın hazırlanması. 3D modellərin yaddaşda olan məlumatları hazırlanır. Buna “culling” də deyilir.

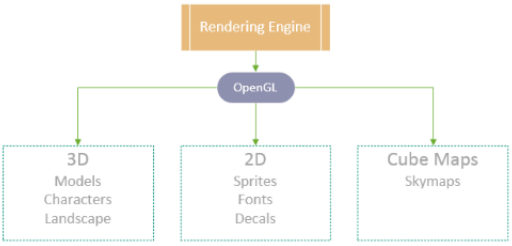
3. İşıqlandırma. İşıqlandırmanın hesablanması çox mürəkkəb prosesdir. İşıq qaynaqlarının müxtəlif olması və ümumi işıqlandırmanın səhnəyə təsirinə əsasən bu proses baş verir. İşıqlandırmanın hesablanması zamanı kölgələr, əks olunmalar, əks olunma zamanı əks olunan işığın digər obyektlər üzərindəki təsiri bunlara daxildir.

4. Görünüş(Visibility). Səhnənin açılması ilə birlikdə səhnədə personajın görünüş dərəcəsinə daxil olan obyektlərin görünməsi bu mərhələlərdən biridir. Təbii ki, Açıq Dünya oyunu yaradan zaman böyük bir xəritə oyunun ilk açıldığı zaman render olunsa da, onun üzərində olan obyektlər və xəritənin tamamı personajın görünüş dərəcəsində görünməsi mümkünsüzdür. Çünki bu dərəcədə ağır bir proses hər hansı bir telefonun və ya kompüterin, qrafik prosessoruna əlavə iş yükü deməkdir.

5. Modellərin yumuşaqlıq sərtlik görünüşü. Modellər üzərində olan ayrıntıların, sərtliyin, çıxıntıların, və materialın verilməsi bu mərhələdə baş verir.

6. Shader. Shader işlənməsi son mərhələlərdəndir. Bu mərhələ işıqlandırma ilə birbaşa əlaqəlidir. İşıqlandırmadan asılı olaraq kölgənin yumşaqlıq və sərtliyi təyin olunur. Təbii ki burada material və modelin növü də təsir edir.

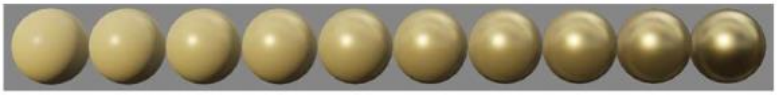
Evgeny Pukki isə bu prosesi Unreal Engine 4 for Automation(2021) məqaləsində aşağıdakı kimi təsvir etmişdir(Şək. 2.0.).



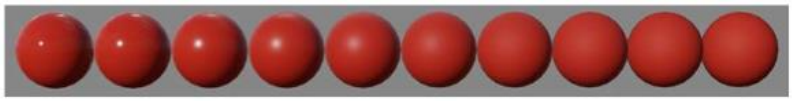

Şək. 2.0. Rendering Engine analizi (Evgeny Pukki, 2021)

### 1.3.2. Material

Material, səhnədəki obyektlərin üzərindəki rəngi, teksturanı və müəyyən illuziya yaradacaq effektləri müəyyən edir. Materialın daxilində kök modulda(Şəkil 18) olan kanallar vasitəsilə real həyatda rastladığımız metal, parıldama və ya əksi, rəngin udulması çıxıntıların verilməsi və.s kimi məsələlər idarə oluna bilər. Materialın yaradılması həm mühərrik daxilindən, həm də fərqli proqramlarda edilə bilər. Unreal Engine daxilində Material yaradan zaman constant, scalar, 2, 3, 4 ölçülü dəyər və tekstura ilə Base Color üzərinə əsas material olacaq kanalı təyin edə bilərik. Bu obyektin əsas görünüşünü təyin edəcək olan kanaldır. Metalic, materiala metal effekti vermək üçün istifadə olunur. Standart dəyər 0.5 olub, 0-1 arası dəyər qəbul edir. 0 dəyər qeyri-metal olacaqsa, 1 metalı təmsil edəcək. Specular, materiala parlaqlıq effekti verir(briliant kimi). 0-1 arası dəyər qəbul edir və birə yaxınlaşdıqca parlaqlıq artır. Specular kanalı Metalic kanalla eyni zamanda işləmir. Miro V, 3D Game Environment in Unreal Engine 4(2014) məqaləsində Metalic və Roughness arasındakı fərqi, dəyərləri dəyişdirərkən aşağıdakı şəkillərlə izah etmişdir(Şək.2.1, Şək. 2.1.1).



Şək.2.1. Metalic (Mehman Şəfiyev, 2024)



Şək. 2.1.1. Roughness (Mehman Şəfiyev, 2024)

Roughness materiala matlıq və parlaqlıq verir. 0-1 arası dəyər qəbul edən bu kanal, 0 dəyərində mat, 1-ə doğru isə parlaqlıq effekti verməyə başlayır. Emmisive Color vasitəsilə materiala materialın işıqlandırmaq olar. Məsələn günəşin rəngini ağ olaraq qəbul etsək həmin ağ rəngin, parıltı dərəcəsini bu kanal vasitəsilə tənzimləmək olar. Opacity vasitəsilə şəffaf material hazırlamaq mümkündür. Lakin şəffaf material hazırlamaq üçün kanal daxilində Translucent(şəffaf) rejim aktiv olunmalıdır. 0-1 arası dəyər qəbul edir. 0-da tam şəffaf, 1-də qeyri-şəffaf olur. Opacity Mask, Opacity-in iş prinsipi kimi işləyir. Arasındakı fərq isə Opacity kimi şəffaflığı tənzimləyə bilmirik. Ya 0, ya da 1 qəbul edir. Opacity Mask-ın üstün tərəfi şəffaf hissələrində dinamik kölgə verə bilməsidi. Normal kanal materiala qabarıqlıq vermək üçün istifadə olunur. Səhnədə olan düz relyefin girinti-çıxıntı olaraq görünməsini buradan tənzimləyə bilərik. Henk V, Wilhelm Ogterop, Unreal Engine 5 Character Creation, Animation, and Cinematics(2022) kitabında Normal Map-in bir illuziya yaratmaq istədiyini yalançı işıqlandırma istifadə edərək səthin qabarıqlaşdırması olaraq təsvir edirdi. Hər bir material hazırlanan zaman onun Normal Map adlanan teksturası yaradılır. Normal Map-da olan qabarıqlıq işıqlandırma və kölgə hesabına verilir. Burada olan əsas məsələ qabarıqlığın illuziya effekti olaraq verilməsidir, yəni poliqonlar şişirdilmir. Tangent vasitəsilə verilmiş teksturaya işıqlandırma və intensivlik verilir ki, bununla saç, axan çay kimi materialları hazırlamaq mümkündür. Backlit səthin arxa fonunun işıqladırma intensivliyidir. İkiqat material istifadə etmək üçün Clear Coat-dan istifadə edilir. İlk qat intensivlik, ikinci isə Roughness-i idarə edir. Materialın işıqlandırılmayan sahələrinə verilmiş teksturaya görə ləkələr vermək üçün Ambient Occlusion-dan istifadə olunur. Bitki örtüyü kimi materiallarda istifadəsinə rast gəlmək olar. Materialın kameraya olan məsafəsinə əsasən verilmiş dəyərə rənglənməsinin istifadəsi üçün Pixel Depth Offset

istifadə olunur. İşığın şüşədə sınma effektini almaq üçün Refraction kanalından istifadə olunur. Təbii ki, sınma dərəcəsi hər şüşə və ya cisimdə eyni olmadığından, bunu tənzimləmək üçün müxtəlif dəyərlərdən istifadə olunur. Material üzərində olan kanalların idarəsindən əlavə teksturanın düzgün seçilməsi də mühüm məsələlərdəndir. Hər hansı bir tekstura yaradılan zaman material daxilində olan kanallarla işləsin deyə, həmin teksturanın ayrılıqda müxtəlif kanalları yaradılır(Metalic, Base Color, Normal və.s kimi).

Su materialının hazırlanması, material daxilində riyazi ifadələr istifadə olunur. Bu riyazi ifadələr hesabına materialın rəngi, dalğalanma, kənar səthlərə dəydiyi zaman dalğaların əks qayıtması kimi bir sıra məsələlər tənzimlənir. Su materialının hər hansı bir səthə dəydiyi zaman yaranan dalğalanması Sine və Cosine ilə həll olunur. Sine və Cosine X, Y oxu üzərində qiymət dəyişimi zamanı uyğun olaraq, Sine 0-dan başlayaraq 1-ə və -1-ə geriləyir. Cosine isə 1-dən başlayaraq -1-ə və oradan yenidən başa qayıdır. Bu uyöun olaraq material daxilində də baş verir və dalğalanmanın haradan başlayaraq necə forma alacağına yön verir. UE 5 versiyadan istifadə edilən Substrate Material isə qat-qat material da dediyimiz, funksionallığı və realistikliyi ilə normal materialdan çox seçilir. Bu tip materialların istifadəsi zamanı əvvəllər Clear Cost kanalı imitasiya edirdisə, həmin materialın istifadəsində susmaya görə iki funskiya işləyirdi.

### 1.3.3. İşıqlandırma

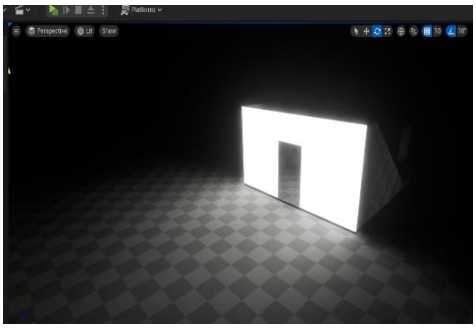
İşıqlandırma səhnənin görünüşünə təsir edən ən başlıca amillərdən öndə gələn təsirlərdəndir. İşıqlandırma demək olar bir çox mühərriklərdə oxşar prinsiplərlə işləyir. Lakin Unreal Engine-də işıqlandırma digər mühərriklərdən daha realistik və optimizasiyalı olaraq işləyir. Buna yeni texnologiyaların da təsiri böyükdür. Unreal Engine üzərində işıqlandırmanın 3 növü var. Bunlar Static, Stationary, Dynamic və ya Movable. Static işıqlandırma 1 dəfə mühərrik tərəfindən işlənilir, buna bake ya da built-də deyilir. Kölgələnmə də buna əsasən hesablanır. Məsələn səhnədə olan obyektlərin üzərinə static işıq mənbəyindən işıqlandırma mühiti qursaq, daha sonra obyektlərin yerini

dəyişsək, işıq mənbəyi static olduğuna görə obyektlərin ilk olduğu yerlərdə kölgə izlərini səhnədə görə bilərik. Statonary tipli işıqlandırma istənilən anda dəyişdirilə bilən işıqlandırmaya deyilir. Bu tip işıqlandırma Static ilə Dynamic işıqlandırmanın arasında keçid kimi də düşünülə bilər. Dynamic isə adından göründüyü kimi tam hərəkətli işıq qaynağıdır(məsələn, günəş). Bu tip işıqlandırma zamanı işıq mənbəyindən gələn şüalar yer dəyişdiyindən real zaman ərzində işıqlandırma və bərabərində kölgələnmə hər periodda hesablanır. Təbii ki, Static işıqlandırma ilə işləyən səhnə daha az resurs tələb edəcək. Bunun üçün mobil oyunlarda bu tipli işıqlandırmadan istifadə olunur. Dynamic işıqlandırma daha çox resurs tələb etsə də səhnə bir o qədər real və gerçəkci görünəcək. Bir daha qeyd edək ki, static işıqlandırma səhnə başlayan zaman bir dəfə hesablanır, və səhnəyə obyektlərin yeri götürülərək həkk olunur. Bunun üçün də bu tipli işıqlandırma daha optimizasiyalı işləyir. Lakin Unreal Engine 5 versiyasından etibarən susmaya görə mühərrik static işıqlandırmadan istifadə etmir. Bunun səbəbi Lumen texnologiyasının istifadəsidir.

### 1.3.4. Lumen

İşıqlandırmanın 2 əsas növü var. Bunlar, fiziki korrekt olunmuş işıq şüalarının trasirovkasıvə işıq şüalarının təxmini trasirovkası(approksimasiyası). Fiziki korrekt olunmuş işıq şüaları işıq mənbəyindən çıxan işıq şüalarının hər bir düşəcəyi səthə toxunur və daha sonra əks olunan səthdən yenidən digər səthlərə ötürülür. Belə işləyən işıqlandırma daha realistik və gözoxşayan olur ki, buna Ray Tracing deyilir. Approksimasiya olunmuş işıq şüaları isə təxmini səthlərə toxunub işıq mənbəyinə doğru hərəkət edir. İşıqlandırma Lightmap adlanan vasitələrlə(asset) işləyir. Bunlar özlərində işıqlandırma haqqında olan bütün məlumatları saxlayırlar. İşıqlanmanın əks olunma məlumatları isə Cubemap-lara yazılır. Screenspace Reflection ilə əks olunma kameranın görmə bucağında hesablanır. Digər hissələrdə isə bu baş vermir. Unreal Engine 4 versiyalarında işıqlandırma yalnız 1 dəfə hesablanır, sərt diskdə saxlanılırdı. Daha sonralar isə real vaxt ərzində işıqlanma, Ray Tracing inkişaf etdirilməyə başlanıldı.

Şüaların daha inkişaf etmiş metodu isə Path Tracing-dir. Burada multiəks olunma baş verə bilər. Yəni eyni zamanda birdən çox əks olunma baş verir. Məsələn, bir güzgü səthindən digər güzgü səthi əks oluna bilir. Belə başa düşmək olar ki, Ray Tracing çox resurs tələb edən lakin realistik verən bir metod idi, lightmap-lar isə az resurs tələb edən lakin çox da realistik görüntüyə sahib olmayan bir işıqlanma idi. Həm realistik olub həm də az resurs tələb edən texnologiya isə Nvidia-nın VXGL(Voxel Global Lightning) oldu. Trasirovka burada konuslardan istifadə edirdi. Hər bir şey voxel-lərə bölünür, daha sonra rənglər korreksiya olunurdu. İşıqlanma və kölgələnmə bu texnologiya ilə əmələ gəlirdi. İşıq mənbəyi burada bir pikselden deyil, konusdan atılırdı. Unreal Engine 5 ilə Lumen texnologiyası gətirildi. Üstün cəhəti isə real işıqlanma ilə birlikdə az resurs tələb etməsidir. Trasirovka tam olaraq həyata keçməsə də, tam şəkildə trasirovkanı istifadə etmək mümkündür. UE 5-də iki növ trasirovka var. Hardware və software trasirovka. Hardware trasirovka uyğun trasirovkanı dəstəkləyən videokart tələb edir. Software trasirovka isə uyğun API-lər dəstəyini tələb edir. Lumen-in iş prinsipinə baxsaq, bir işıq mənbəyi ilə səhnəni işıqlandırsaq, işıq şüalarının perpendikulyar səthə doğru düşdükdən sonra müəyyən bucaq altında digər səthlər üzərində yansımasını görə bilərik(Şək. 2.2.).



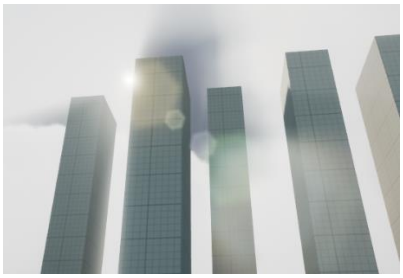
Şək. 2.2. Lumen ilə işıqlandırılmış səhnə (Mehman Şəfiyev, 2024)

Lumen məlumatları sərt diskə deyil, keş yaddaşa yazır. Buna görə həm realistlik, həm də sürətli işləmə eyni anda əldə edilir. Bundan əlavə, material bölümündə danışdığımız Emissive Color, özü də işıq şüaları saçdığından Lumen texnologiyası buna da təsir edir. Lakin Lumen şəffaf materiallarla işləmir.

Jiashuai Du, Liping Su, Dong Chen, Innovative Design for Interactive Display of Ancient Architectural and Cultural Heritage using UE5 Virtual Engine (2023. 9p) məqaləsində işıqlandırmada yeni texnologiyarın istifadəsi zamanı VR, AR texnologiyalarına da təsirindən bəhs etmişlər.

### 1.3.5. Post Process Volume

Post Process Volume səhnədə işıq mənbəyindən çıxan işığın kamera perspektivində necə görünməsi gərəkdiyini, əlavə olaraq müəyyən məkanlarda işıq şüalarının görünürlüyünü dəyişmək, termal və gecə görüş kamerası hazırlamaq və.s kimi bir çox yerlərdə istifadə olunur. Post Process vasitəsilə səhnədə olan abu-havanı dəyişmək olar. Məsələn qorxu janrında olan havadakı soyuqluq və ya isti havada olan havadakı açıqlıq buna misal ola bilər. Real həyatda çox işıqlı bir mühitdən qaranlıq mühitə keçid etdikdə və ya günəşə baxıb anidən fərqli yerə baxdığımız zaman gözümüzün qaralaraq həmin mühitin işığına adaptasiya olunmasını Post Process daxilində olan Exposure bölümü vasitəsilə etmək mümkündür.Günəş işığının gözə və ya kameraya yansıyaraq lens-dən əks olunması effektini də bu hissədən, Lens Flares(Şək. 2.3.) vasitəsilə tənzimləyə bilərik. Post Process-in ən çox istifadə olunan funksionallıqlarından biri də qapalı mühitlərdə çətin işıq alan künclərin işıqlanmasını tənzimləmək üçün Ambient Occlusion ilə tənzimləməsidir.



Şək. 2.3. Post Process vasitəsilə Lens flares (Mehman Şəfiyev, 2024)

İstənilən işıq mənbəyinin öz ölçüsü olur. İşığın birbaşa düşə bilmədiyi hissələrdə 3 əsas zona əmələ gəlir. İlk olaraq Umbra, yəni kölgə olan hissə, ikinci Penumbra, yəni kölgənin kənarları, son olaraq Antumbra, yarı kölgələnən ərazi. Əgər işıq mənbəyi düşəcəyi səthdən uzaq və ya işıq mənbəyi kiçikdirsə kölgənin kənar xətləri bir o qədər sərt olacaq. Amma

nə qədər yaxın olarsa və ya işıq mənbəyi nə qədər böyük olarsa kölgə bir o qədər yumuşaq olacaq. Məsələn, aydın havada kölgələri səthdən asılı olaraq sərt işləmək lazımdır. Lakin tutqun havada tam əksinə.

### 1.3.6. Nanite

Unreal Engine 5-də ən innovativ yeni texnologiyalardan biri olan Nanite-dır. Bu texnologiya oyun sənayesi və ümumilikdə kompüter qrafikasını kökündən dəyişəcək bir texnologiya sayılır. Əvvəllər oyun tərtibatçıları səhnədə çox sayda obyekt yerləşdirmək üçün həmin obyektləri optimizasiya etmək məcburiyyətində idilər. İlk olaraq yüksək detallaşdıırlmış HighPoly modellər düzəldilirdi. Daha sonra bu yüksək detallaşdırılmış modellərin cizgiləri normal map adlı texturalara köçürülürdü. Daha sonra eyni modelin aşağı deltallı eynisi işlənilirdi və modelin lazım olan sahələrində retopologiya işləri görüldükdən sonra həmin normal map-in köməyi ilə yüksək detallaşdırılmış modellərə bənzər modellər düzəldilirdi. Bir növü illuziya yaradılırdı. Amma bir problem yananırdı. Səhnədə eyni zamanda çox sayda HighPoly modellər yerləşdirmək lazım olurdusa bu zaman məsələni necə isə həll etmək lazım olurdu. Bu zaman isə LOD (Level of Details) texnologiyası işlənildi. Aleksi Karppinen, Creating foliage for Video Games məqaləsində 200 min poliqondan ibarət bir ağacları modellərindən ibarət səhnədə Nanite ilə işləyən səhnə LOD-a nisbətdə çox üstün performans verdiyini təyin edir deyə yazırdı. LOD texnologiyası ona əsaslanırdı ki, eyni modelin bir necə versiyası işlənilirdi və sıfırıncı, yəni ən birinci model ən yüksək detallaşdırılmış model olurdu. Ən sonuncu modelə getdikcə isə həmin modelin poliqon sayı müəyyən interval ilə azalırdı. Bu dəyişimi isə kameranın həmin modeldən olan məsafəsinə görə tənzimlənirdi. Beləliklə kameradan daha uzaqda olan modellər ən aşağı detallı modellər istifadə olunurdu və səhnədə yüklənmənin qarşısı alınırdı. Amma Unreal Engine 5-də yaradılan Nanite texnologiyası bütün bu artıq işlərdən tərtibatçıları xilas etdi. Nanite texnologiyasının üstünlüyü ondadır ki, burada modelin detallaşdırılmış və poliqon sayı ilə bağlı limit olmur və ən əsası bu texnologiya işləyən zaman normal map-a ehtiyac olmur. Nanite texnologiyası detallaşdırılmanı özü avtomatik

olaraq oyunçudan(personajımızdan) olan məsafəyə görə tənzimləyir. Amma burada hamını maraqlandıran əsas bir sual ortaya çıxır. Əgər oyunda HighPoly modellər istifadə etsək bu zaman oyunumuzun və ya proyektimizin həcmi nə qədər olacaq? Burada da Nanite texnologiyasının üstünlüyü var. Məsələ orasındadır ki, Nanite virtuallaşdırılmış geometriya şəbəkəsidir. Bu geometriya şəbəkəsi yeni format sayılır. Hansı ki, burada hər bir detalı avtomatik olaraq həyata keçirilir. Nanite formatlı məlumatlar isə öz növbəsində çox sıxılmış (compress) halda olurlar. Nanite texnologiyası necə işləyir? Modeli mühərrik daxilinə köçürən (import) zaman və ya mühərrik daxilində olan zaman modeli Nanite texnologiyasına konvert (çevirmək) etdiyimiz anda modelin geometriya şəbəkəsinə analiz olunur və klasterlərə (böyük qruplara) bölünür. Bu klasterləri poliqonlar qrupu təşkil edir. Daha sonra render konveyerində (rendering piline) həmin modelin kameradan məsafəsinə görə klasterlər işə düşür və öz yerlərini dəyişməyə başlayır. Burada bir növü LOD texnologiyasının işləmə prinsipinə bənzərlik ola bilər. Amma Nanite daha qəliz və daha innovativ texnologiya olduğu üçün bu müqayisə düzgün olmaz. Nanite ayrı bir patokda render olunur. Buna görə də ənənəvi drawcall texnologiyasından tamami ilə fərqli texnologiya ilə işləyir. Nanite texnologiyası PlayStation 5, XBOX series və DirectX11-12 versiyanı dəstəkləyən personal kompyuterlərdə işləyir. Nanite texnologiyası daha çox poliqon sayı istifadə olunan modellərdə tətbiq olunur. Eyni zamanda Nanite texnologiyasını səhnədə eyni modelin daha çox nümunəsi istifadə olunan zaman aktivləşdirmək məsləhətdir. Misal üçün ağaclar, otlar, binalar və s. Nanite texnologiyası statik obyektlər ilə işləyir. Beləki, skeletal obyektlərdə animasiya istifadə olunduğuna görə Nanite texnologiyasını paralel istifadə etmək məqsədə uyğun sayılmaz. Çünki modelin animasiyası zamanı poliqon sayı çoxaldıqca və ya dəyişdikcə animasiyanın dəqiq işləməsi çətin olar. Ona görədə tərtibatçılar skeletal model hazırlayarkən polqon sayına xüsusi nəzarət edirlər ki, animasiya zamanı işləmək mümkün qədər rahat olsun. Skeletal və statik mesh-lər haqda danışacayıq. Beləliklə Nanite texnologiyasında böyük məlumat axınını nəzərə alıb bu texnologiyanı SSD yaddaş qurğusu ilə işlətmək daha məsləhətdir. Tianshu

Li, real–time performance comparison of environments created using traditional geometry rendering versus unreal nanite technology in virtual reality(2024, 35 p) məqaləsində Nanite texnologiyası ilə fps dəyərlərinin əvvəlki texnologiyalar ilə müqayisəsini göstərmişdir.

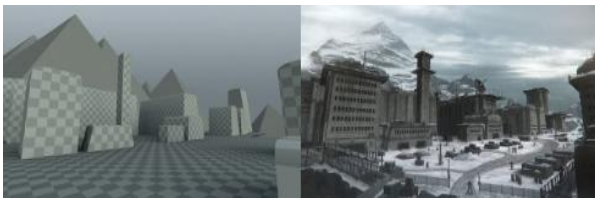
Nanite texnologiyasında istifadə olunan geometriya şəbəkəsinin sıxlığına baxmayaraq , Nanite-a konvert olunmuş modeller yaddaşda daha az yer tutur. Burada bir poliqon yaddaşda 14.4byte yer tutur. Bu isə o deməkdir ki, 1milyon poliqonu olan modeli Nanite-a konvert etsək bu zaman model yaddaşda 13.8mb yer tutacaq. Bu isə ancaq geometriyanın-poliqonların həcmidir. Nəzərə almaq lazımdır ki, modelin materialı, texturası, sheyderlər və s. ola bilir. Təsəvvür edin adi base color texture 4k ölçüdə 8.2mb yaddaşda yer tutur. Eyni 4k ölçüdə normal map isə 21.85mb yaddaşda yer tutur. Yuxarıdakı göstəriciləri nəzərə alaraq 12min poliqonlu bir modelin normal map və digər texturalarla birlikdə aşağı detalizasiyada 31.04mb yaddaşda yer tutacaq. Eyni modelin yüksək detalizasiyalı növü isə yaddaşda 49.69mb yer tutacaq. Amma eyni modelin yüksək detalizasiyalı və bütün texturalar ilə Nanite-a konvert olunmuş növü isə yaddaşda 27.83mb yer tutacaq. Fikir verdinizsə eyni modelin aşağı detalizasiyalı növü Nanite texnologiyasına konvert olunmuş yüksək detalizasiyalı növündən daha çox yaddaşda yer tutur. Stephen R, LODs and Nanite within Unreal Engine 5: The Future of 3D Asset Creation for Game Engines adlı məqaləsində Nanite texnologiyasının LOD ilə müqayisəsini və aralarında işləmə prinsipi olaraq fərqliliklərini qeyd etmişdir.

# II FƏSİL. OYUNLARIN QURULMASINDA TEST MƏRHƏLƏSİ

## 2.1. Alpha Test mərhələsi: Daxili Test olunma və səhvlərin fikslənməsi

Oyunlar bazara satışa çıxmazdan əvvəl müxtəlif test mərhələlərdən keçir. Bu mərhələlər oyunların fərqli istiqamətlərdə inkişafına yol açа, məzmunda kiçik dəyişikliklərə gedə bilər. Başlanğıc zamanı səhvlərin, mexaniki xətaların, animasiyaların bir-birləri ilə konflikt yaratmasını bu hissədə görə bilərik.

Ən başdan başlasaq, oyun hazırlanmasında müxtəlif qruplar arasında bağlantı olur. Bunlar VFX Artist, SFX Artist, Animator Artist, Environment Artist, UI Artist, Material Artist, Technical Artist, kimi qruplar bir-birlərilə sıx əlaqədə olurlar. İlkin olaraq landscape hazırlanması ilə başlanılan mərhələdə ətraf mühit dizayn edilir. Paralel olaraq əgər Shooter və ya multiplayer oyundursa personaj dizaynı da bərabərində olur. İlk ətraf mühit dizaynı zamanı hər şey Grey Box(Şək. 2.4.) adlanan mərhələdə olur.



Şək. 2.4. Grey Box səhnənin son hala olan dəyişimi (Mehman Şəfiyev, 2024)

Grey Box adlanan bu mərhələdə ətraf mühitdə olan obyektlər sadə formada olacaq şəkildə cisimlərlə əvəzlənir. Bu hissə sonradan dəyişikliyə gedəcək olan cisimlərin ilkin təyini kimi başa düşülə bilər. Məsələn hər hansı konteyner bir kubla, maşın bir konusla və.s kimi. Alpha test də öz daxilində bir neçə mərhələyə ayrılır. Təbii ki, göstərəcəyimiz mərhələ və nümunələr bir qaydalar toplusu olaraq deyil də, bir yol xəritəsi olaraq başa düşülə bilər. Şirkətlərin öz daxilində olan qaydalara, işləmə yöntəmlərinə əsasən bunlar kiçik dəyişikliklər göstərə bilər. Early deyə adlandırılan ilkin mərhələdə oyunun mexanikası, strukturu qurulur. Bu mərhələdə əvvəlcədən təyin olunmuş ilk işlər, personajın seçilməsi və əsas animasiyaları hazırlanır. Paralel olaraq ssenari üzərində yazılmış mərhələlərin kod

hissəsinin yazılması başlanılır. Paralel olaraq animatorlar, bu mərhələlər zamanı olacaq digər animasiyaların hazırlanması ilə məşğul olarkən, xəritənin hazırlanması prosesi davam edir. Xəritənin üzərində olacaq obyektlər məzmuna uyğun seçilərkən, bu obyektlərin materialları isə Material Artist tərəfindən hazır olunur. Landşaftın və obyektin materiallarının düzgün seçilməsi bundan sonrakı mərhələlərin ləngiməməsi üçün önəmlidir. Daxili test adlandırılan mərhələ zamanı proqramçılar öz aralarında işin müəyyən faizi görüldükdən sonra test etməyə başlayırlar. Nataliia F, Maksim P, Iryna B, Svitlana V, Yaroslava D, Research On Calculation Optimization Methods Used In Computer Games Development(2023, 34 p) məqaləsində obyektlərin optimizasiyası üçün görünən nöqtə, LOD hissə və ya gizlənən hissə, LOD hissədən çixiş kimi bir bənzətmə modelindən bəhs etmişlərdir ki, bu da optimizasiya üçün önəmli bir məsələdir.

Bəzi böyük şirkətlər tərəfindən idarə edilməyən və profesional karyerası olmayan proqramçıların bu mərhələlərdə etdiyi bir qrup səhvlər var. Clean Code deyə adlandırılan, başa düşülən, aydın, kiçik, təkrarlara yol verilməyən, hər bir kodun açıqlaması üzərində olan, test olunmaya hər daim uyğun olan, ierarxik olaraq düzgün qurulmuş və ən əsası optimizasiyalı yazılmış koddan istifadə olunmalıdır. Bir çox proqramçının sonradan bu xətaları düzəldərəm deyə başladığı yolda, xətaların həddən artıq çoxalması iş yükünü artıran amillərdəndir. Oyunun performans testlərinin olduğu mərhələdə kodlarda olan xətalar və ya kodların alternativ çıxışlarının olması yolunda bir sıra dəyişikliklərə gedilə bilər. Mexanikada olan konfliktliklər, oyunun çökməsi kimi halların olmaması üçün bu mərhələ çox vacibdir. Testerlərə ilkin geri dönüşlərin alınması üçün oyun verilməzdən əvvəl, proqramçı qruplar komanda liderləri və Technical Artist-lər dəyərləndirmələr edirlər. Personaj dəyişimi, oyunun mərhələlərində qısa məzmun dəyişimi kimi halların baş verməsi normal sayılır. Daha sonra testerlər oyun haqqında ilkin rəylər vermək üçün oyunu sınaqdan keçirərlər. Təbii ki, fikirlərin obyektiv olması üçün testerlər qruplara ayrılır. Burada mexanikada, xəritədə, UI elementlər üzərində olan səhvlərin aşkar olunması kimi nüanslar daxildir. Gələn geri dönüşlərə əsasən proqramçılar ilkin alpha

mərhələsinin sonunda oyunu bir daha nəzərdən keçirərək, xətaların düzəldilməsi və yeniliklərin əlavə edilməsini aparırlar.

### 2.1.1. Beta Test mərhələsi: Xarici Test olunma və rəylərin toplanması

Beta test mərhələsi daha geniş bir kütlə üzərində sınaqdan keçirilir. İkinci test mərhələsi sayılan bu mərhələdə oyun bazara Beta versiya olaraq çıxarılır və yerləşdirilən platformaya əsasən oyunçular oynanılması üçün dəvət olunur. Buna misal olaraq yaxın zamanlarda çıxan Gray Zone Warfare buna misal göstərmək mümkündür. Platformalara yerləşdirilməyində əsas məsələ oyunçulardan rəy yığılmasıdır. Anketlərlə, oyun daxilindən reportlarla proqramçılara xətaların və təkliflərin bildirilməsi burada baş verir. Əgər Multiplayer bir oyundursa, performans artırılması, online serverlərdə olan xətaların düzəldilməsi beta versiyaya təsadüf edir.

### 2.1.2. Parlatma(Polishing) və Optimizasiya

Oyun keyfiyyətinin artırılması üçün istifadə olunan 2 mərhələ oyunun tam versiyası çıxdıqdan sonra ümid verici bir oyundursa proqramçılar oyun üzərində yaxşılaşdırmalara gedərək oyunu təkmilləşdirməyə davam edərlər. Polishing adlanan mərhələdə oyunun qrafik və effektlərin təkmilləşdirilməsi baş verər. Animasiyalar, personaj dəyişikliklərinə gedilə bilən bu mərhələdə, oyun üzərində mexanikanı daha axıcı etmək üçün oyunun sürətləndirilməsi aparıla bilər. Səs effektlərinin realistik olunmasına doğru gedilən yeniləmələr ola, UI üzərində dəyişikliklər mütləq şəkildə aparılır. Oyunların bir çoxunda UI elementləri göz yormaması üçün zaman zaman ən çox dəyişikliyə gedən elementlərdəndir.

Optimizasiya oyun performansının artırılması üçün istifadə olunan bir ifadədir. Buraya kodun optimizasiya olunması, lazımsız kod hissələrinin azaldılması, kod hissələrində əlavə alternativlərin gətirilməsi ola bilər. Qrafik yaxşılaşdırılmarın aparılması da buraya daxildir. Məsələn, Unreal Engine 5 versiyası üzərində Lumen işıqlandırmaya keçilməsi bina nümunə ola bilər. Loading Screen dediyimiz, gözləmə ekranı olaraq bilinən oyunun

yükləmə ekranında kod hissəsində olan yaxşılaşdırmalar buna ən böyük nümunədir. Məsələn, Unreal Engine üzərində BP üzərindən hələ ki, Loading Screen hazırlamaq mümkün deyil, bu əməliyyat C++ kodu üzərindən hazırlanır, əlavə olaraq isə plugin istifadə oluna bilər.

Mühərrik daxilində yaradılan komponentlərin optimizasiya olunması mühüm məsələlərdəndir. Bu hissəyə qədər danışdığımız texnologiyalar təkcə yaradılması ilə kifayətlənmir, çünki səhnəyə qoyulan hər bir obyekt, həm proyektin həcmini artırır, həm də lazımsız parametrlərin yüksək dəyərlərdə işləməsi optimizasiyanı korlayır. Bu texnologiyaların optimizasiya olunmasına baxaq.

**Polygon sayı.** Obyektlərin polygon sayısı bəzən gərəksiz olacaq şəkildə çox ola bilər. Səhnədə vaciblik dərəcəsindən asılı olaraq bu obyektlərin polygon sayı azaldıla bilər. Tutaq ki səhnədə çox istifadə olunan daş, ağac, ev kimi obyektlərin polygon sayısını lazımsız dərəcədə çox olsa, obyektlərin çox istifadə edilməsinə görə FPS enmələrilə qarşılaşa bilərik. Bunun qarşısını almaq üçün, Blender və ya Unreal Engine üzərində polygon sayısını azalda bilərik. Bu cür obyektlərin polygon sayısının çox olmasının çox da əhəmiyyəti yoxdur. Çünki, diqqətli olaraq incələnməyən obyektlərin polygon sayısının oyunlarda çox olmaması lazımdır. Lakin kino və realistik səhnələrin hazırlanmasında səhnənin ola bildiyi qədər reala yaxın görünməsi üçün polygon sayısında azalmaya gedilmir. Bu cür səhnələrdən yalnız render alınır. Modellərin UV kanallarına baxaraq bunları daha yaxşı anlamaq olar.

**Cull Distance Volume.** Səhnəni tam hazırladıqdan sonra müəyyən hissələrində obyektləri yerləşdirdikdən sonra həmin obyektləri hər zaman görməyimizə ehtiyac olmur. Məsələn, ev daxilində olan əşyaları yalnız evə daxil olduğumuz zaman görəcəyiksə 100-200 metr uzaqdan həmin evin içərisində olan obyektləri görməyimizə və görməsək də həmin obyektlərin orada görünürlüyünü saxlamağa ehtiyacımız yoxdur. Bunun üçün LOD sistemdən istifadə olunur. Unreal Engine 5 versiyadan isə Nanite texnologiyası bunu

əvəzlədi. LOD necə istifadə olunur. Hierarchical LODVolume-dan istifadə olunur. Elmi işin Polygon sayısı olan hissəində bəhs etdiyimiz kimi, 3D modelyer, modeli hazırlayan zaman həmin obyektin LOD modellərini də yaradır. LOD modellərin rəqəmi yüksəldikcə Polygon sayısı düşməyə başlayır. Cull Distance Volume daxilinə qoyduğumuz və polygon sayısından asılı olmayaraq modelləri istədiyimiz məsafədə görünür edə və görünməz edə bilərik. Modelin görünməz olması onun səhnədə olmaması deməkdir. Bu o deməkdir ki, səhnəni yaradan zaman 20 ədəd modeli səhnəyə qoymuşuqsa, **Cull Distance Volume** daxilinə salınmış model varsa səhnədə 19 model var deməkdir, ta ki, verdiyimiz məsafəyə daxil olub, həmin modeli görənədək. Volume doğru yaxınlaşdıqca obyektlər görünməyə başlayır. Hierarchical LODVolume da Cull Distcance Volume kimi daxilində obyektlərin qoyulduğu bir Volume olsa da əsas fərqi LOD-a əsasən obyektlərin optimizasiya edilməsidir.

**Textura-ın optimizasiyası.** Textura-lar ilə işləyən zaman Unreal Engine mühərriki susmaya görə müəyyən optimizasiya işləri aparır. Amma nəzərə alsaq ki, işlədiyimiz proyektimiz mobil üçün olduqda textura-ların optimizasiyası xüsusi əhəmiyyət kəsb edir. Çünki mobil proyektinizdə textura-ları optimizasiya etmədiyimiz zaman performansa pis təsir, proyektinizin həcmi isə çox böyük ola bilər. Optimizasiya üçün ilk növbədə textura-ların ölçüsünü tənzimləməliyik. Bunun üçün müxtəlif proqram təminatları vardır(paint.net kimi). Fərz edək ki, mobil oyun üçün partlayış effekti düzəldirik. Bu zaman təbii ki, ilk növbədə uyğun texturalarımız olmalıdır. Bir çox tərtibatçı vizual effekt və ya başqa işlər görərkən optimizasiyanı həmişə ən sona saxlayır. Bu səhv addımdır. Yuxarıdakı nümunəyə görə optimizasiyanı sonda etməyimiz düz olmaz. Ona görə də textura-nı ilk növbədə ölçüsünü tənzimləmək üçün müəyyən proqramlar ilə açırıq. Daha sonra isə textura-nı Resize edirik, yəni ölçüsünü kiçildirik. Textura-nın ölçüsünü kiçildən zaman minimum kiçik ölçülərdən istifadə olunmalıdır. Amma textura-nı elə kiçiltmək lazımdır ki, daha sonra təyinatı üzrə istifadə etdikdə textura-nın kiçildildiyi o qədər də hiss olunmasın. Bunu isə ölçüləri təyinatı üzrə özümüz tənzimləməliyik. Məsələn, vizual

effekt üçün istifadə edəcəyimiz textura 1024x1024 ölçüdədirsə bu texturanı 256x256-ya qədər kiçildə bilərik. Bu ölçüdən kiçik etsək vizual effekt yaxşı görünməyə bilər. Daha sonra mühərrik daxilində proyekt üçün əsasən png formatlı şəkillərdən və vector qrafikalı icon-dan istifadə edilməlidir. Pixel qrafikalı bir şəkil vector qrafikaya nisbətən daha çox yaddaş tutumuna malikdir. Əgər textura streaming funksiyasını dəstəkləyirsə bu çox yaxşıdır. Çünki mühərrik daxilində istifadə olunan bu textura kamera ilə paralel üzərində olan obyektdə dinamik şəkildə yüklənir. Əks halda belə textura ümumi render konveyerində iştirak etmir. Daha sonra textura-lar təyinatı üzrə istifadə edilməlidir. Məsələn, adi jpeg formatlı bir şəkili menyuda hansısa element üçün istifadə etmək məsləhət deyil. Menyu üçün xüsusi icon file olur və böyük həcmli şəkillər burada əsasən vector qrafika ilə işlənilir.

# III FƏSİL. OYUN YARADILMASI ZAMANI TƏHLÜKƏSİZLİK MƏSƏLƏLƏRİ

## 3.1. Oyun serverlərini hədəf alan DDoS hücumları.

Gəlin ilk öncə oyun serverlərinin nə olduğundan başlayaq. Multiplayer oyunları oynamaq üçün bir oyun serverinə ehtiyacımız var. Mehman.Ş, Kamil.C, Fərid.Ə tərəfindən 2024-cü ildə yazılan Unreal Engine 5(səh. 223-226) kitabında multiplayer oyunların növləri haqqında məlumat verilmişdir. Multiplayer oyunlar bəzi xüsusiyyətlərinə görə növlərə bölünürlər. Bunlara Offline Multiplayer, Dinamik Single Player və Turned Based Multiplayer oyunların adlarını çəkmək olar:

1.Offline Multiplayer oyunlar, bir oyun konsolunda və ya kompüterdə 2 və daha çox oyunçunun eyni ekranda oynadığı oyunlardır və belə oyunlarda, oyunun vizual renderi və digər arxa tərəfdə işləyən kodlar bir cihaz üzərində emal olunur.Nümunə olaraq bu oyunlara Tekken,Mortal Combat kimi konsol döyüş oyunlarının adlarını çəkə bilərik.

2.Dinamik Single Player oyunlar isə əsasən nəticələrin bir oyun serverində toplandığı oyun tipidir.Gəlin bu oyun tipinə aid nümunə çəkək.Məsələn xal yığma əsasında qurulan bir oyun oynayırsızvə bir oyun sessiyasının sonunda qazandığınız xallar hamı tərəfindən onlayn şəkildə görüləcək liderlər siyasına yazılır.

3.Turned Based Multiplayer(TBM) oyunlar.Burada 2 və ya daha çox oyuncu eyni anda bir-birlərinin hərəkətlərini görür,yəni bir oyunçunun oyun zamanı yaradığı paketlər digər oyunçularada göndərilir.Buna misal olaraq onlayn olaraq oynana bilən bilyard və şahmat oyunlarının adını çəkə bilərik.

Məsələn Turned Based Multiplayer oyun tipinə aid olan P2P Shooter oyunlarında visual və digər bütün kadrlar oyunçunun öz cihazında render edilir və qarşı tərəfə sadəcə oyunçunu xəritə üzərində harda olduğunu,hara atəş açdığını və s. bildirən paketlər göndərir.Burada Cloud Based Gaming ilə Turned Based Multiplayer oyunları qarışdıra bilərik.Lakin diqqət etməliyik ki, Cloud Based Gaming-də hər şey cloud tərəfdə emal edilir və cihaz üzərində oyunla bağlı heç bir proses işləmir.TMB tipi oyunlarda saniyədə

100-lərlə məlumat mübadiləsi olduğu üçün burada işlədilən oyun serverləri güclü serverlər olmalıdır və bu serverlərə daimi baxım edilməlidir.

Multiplayer oyunlarda qoşulma “client-server” texnologiyası əsasında qurulur.Client oyunçunun özüdür.Server isə oyunçuların qoşulduğu və məlumat mübadiləsinin aparıldığı sistemdir.Məlumat mübadiləsi TCP/İP üzərindən aparılır.Bu serverlər public serverlər olduğu üçün DDoS hücumuna qarşı açıqdırlar.İlk öncə DDoS hücumunun nə olduğunu anlayaq.DDoS(Distributed Denial-of-Service) Paylanmış Xidmətdən İmtina hücumudur.Yəni botnet şəbəkəsinə daxil olmuş cihazlar tərəfindən müəyyən bir xidmətə göndərilən çoxlu istəklərin nəticəsində, servisin bu istəklərə cavab verə bilməyərək çökməsilə nəticələnən hücumdur.Oyun serverlərinə edilən DDoS hücumlarının fərqli məqsədləri ola bilər.Bunlara oyunda haqsız üstünlük yaratmaq, oyun serverinin işləməsini dayandırmaq, digər oyunçuların oyunu oynamasına maneə yaratmaq aid edilə bilər.Oyun serverləri əsasən 3 formada DDoS hücumuna məruz qalır: kanallara edilən trafik bombardmanı, şəbəkəyə edilən paket daşqınları və OSİ modelinin L7 yəni Application səviyyəsində olan hücumlar.Multiplayer oyunlarda əsas tələb oyunun istifadəçilər tərəfindən kəsintisiz şəkildə oynanılmasıdır.Serverdə hər hansısa problem yaranan zaman bu oyunçuya təsir edir. Məsələn bir sessiyalıq oynanan və son zamanlarda məhşur olan PUBG oyununu nümunə olaraq götürək.Bu oyunda 2 əsas server var.Lobbi serveri və Oyunun öz serverləri.Lobbi serverlərində oyunçular toplanır daha sonra yetəri qədər oyunçu toplanarsa, oyun sessiyasının başlandığı əsas serverə yönləndirilir.

### 3.1.1. İstifadəçi hesabları və şəxsi məlumatlarını pozan məlumat pozuntuları.

Bildiyiniz kimi, son zamanlar kiber hücumların artması ilə insanların şəxsi məlumatları, sosisal şəbəkə hesabları, şəxsiyyətini təsdiq edən sənədlər, bank məlumatları dark web üzərindən sızdırılır və kripto valyuta ilə satışı həyata keçirilir.Hətta www.haveibeenpwned.com saytı üzərindən emailinizin nə vaxtsa sızdırılıb,sızdırılmadığını tapa bilərsiniz:Məsələn nümunə kimi bir temporary email yaradaraq yoxlayaq(Şək. 2.5.):


Şək. 2.5. Web sayt təsviri (Rəşad Məmmədov, 2024)

Yoxladığımız zaman bu emailin sızdırılmadığını görürük.Sızdırılan məlumatlar arasında sizin onlayn oyunlardarkı hesablarınız, Steam və ya Epic Games Store oyun satış platformalardakı hesablarınız ola bilər.Bu hücumlar klient və server tərəfli olaraq 2 formada baş verir.

Gəlin ilk öncə serverə edilən hücumlardan danışaq.Bu hücumlar birbaşa serverə edilir və istifadəçilərin məlumatlarının oğurlanması ilə nəticələnə bilər.SQL İnjection, Command İnjection, File Upload, Local File İnclusion/Remote File İnclusion hücumlarının nəticəsində oyun serverindən istifadəçilərin verilənlər bazasındakı məlumatları oğurlana bilər.Belə hücumlardan qorunmaq üçün input validation, input sanitization, network sanitization kimi həllərdən istifadə etməliyik.

Klient tərəfli edilən hücumlara isə əsasən dictionary və brute force hücumu, XSS,sosial mühəndislik hücumları aid edilir.Gəlin hər biri haqqında danışaq.Dictionary hücumunda hədəf araşdırılır və təyin edə biləcəyi şifrələr təxmin edilərək siyahı yaradılır və ya ictimaiyyətə açıq olan hazır siyahılardan istifadə edə bilər.Daha sonra bu siyahıdakı şifrələr ilə qarşı tərəfin istifadəçi adı birləşdirilərək yoxlanılır.Brute Force hücumuda Dictionary hücumla eynidir, sadəcə bu hücumda hərf,rəqəm,işarə kombinasiyalarından istifadə edilərək təsadüfi şifrələr yaradılır.XSS hücumları isə oyunların rəsmi saytlarında və ya oyunlardakı comment yazıla bilən sahələrdə, JavaScript kodlarının yazıla bilməsi ilə nəticələnən hücumlardır.Oyun sənayesində baş verə biləcək sosial mühəndislik hücumları isə əsasən epoçt üzərindən olan, zərərli linkərin göndərilməsi ilə baş verən hücumlardır.Məsələn belə bir nümunə çəkək.Brauzerdə “steam” platformasını açmısınız və öz istifadəçi adı və şifrənizi yazaraq daxil olmusunuz.Bu zaman siz hər dəfə səhifəni

yenilədikdə yenidən şifrə istəməməsi üçün sayt tərəfindən kuki təyin edilir və klient tərəfdə yaddaşda saxlanılır.Epoçt üzərindən gələn poçtda isə sizin kukilərinizi oğurlaya bilən link ola bilər və oğurlanan kukiləriniz ilə sizin adınızdan steam platformasına giriş edə bilərlər.Klient tərəfli hücumlardan qorunmaq üçün təxmin edilməyən və mürəkkəb şifrələrdən istifadə etmək və bilmədiyimiz epoçtlardan gələn linkləri açmamaq lazımdır.Eyni zamanda zərərli kodların cihazınızda icra edilməməsi üçün EDR/XDR kimi həllərdən istifadə etmək olar.

Oyun sənayesindəki ən böyük istifadəçi məlumatı sızıntısı Sony tərəfindən olmuşdur.2011-ci ildə edilən hücumun nəticəsi ilə Sony-dən 77 milyondan çox hesab sızdırıldı.Şirkətin itkisi 171 milyon dollara yaxın idi.

### 3.1.2. Oyunu pozan fırıldaqçılıq və hücum alətləri.

İlk öncə gəlin cheat və hücum arasındakı fərqin nə olduğunu anlayaq.Cheat oyunun daxilində hiylə etmək üçün istifadə edilir.Məsələn P2P Online Shooter bir oyunda bir oyunçu digər oyunçuların yerini göstərmək üçün cheat istifadə edərək haqsız üstünlük əldə edə bilər.Cheat hər hansısa bir oflayn oyun daxilində oyunçunun oyun daxili öz resurslarını artırmaq üçün istifadə edilə bilər. Cheating üçün oflayn oyunlarda ən çox istifadə edilən toollardan biri Cheat Engine-dir.

Gəlin Cheating(fırıldaqçılıq)-in formalarından danışaq. Rens van Summeren 2011-ci ildə yazdığı Security in online gaming məqaləsində(səh. 10-13) onlayn oyunlara qarşı olan fırıldaqçılıq hallarından bəhs etmişdir. Yan və Randell() onlayn oyunlardakı fırıldaqçılığı 7 əsas formaya bölür:

1.Exploiting misplaced trust - böyük onlayn oyunların serverlərində sadəcə məlumat mübadiləsi aparılır və digər bütün hesablamalar müştərinin cihazında baş verir.Bu zaman müştərilər, müştəri tərəfindəki oyunun koduna, konfiqurasiya məlumatlarına dəyişiklik edə bilərlər.Bu zaman oyunun serverlərinə gedəcək məlumat bu zərərli oyun proqramı üzərindən gedəcək.

2.Colusion - Onlayn oyunlarda 2 və daha çox oyunçunun sövdələşərək digər oyunçular üzərində və ya ümumi olaraq oyun üzərində üstünlük qurmasıdır.Məsələn 2 nəfərlik oynana bilən yarış oyununu nəzərdən keçirək.Burada hər 2 oyunçu sıra ilə bir birinə uduzaraq öz xallarını bu şəkildə artıra bilərlər.Alternativ olaraq bir oyunçu özünə 2 fərqli hesab açaraq bunu həmin hesablar arasındada edə bilər.

3.Abusing game procedure - oyunların öz prosedurlarından sui istifadədir.Bu fırıldaqçılıq üçün heç bir texniki bilik tələb edilmir.Bu fırıldaqçılıq növünə bir nümunə çəkək.Məsələn uduzmaqda olduğunuz oyundan son anda çıxmaq.Onlayn oyunlarda bunun qarşısını almaq üçün oyun sessiyası zamanı oyunu tərk edən oyunçular cəzalandırılırlar.Məsələn buna nümunə olaraq P2P Online Shooter oyunlarını nümunə çəkə bilərik(Şək. 2.6.):Point Blank,Valorant

Şək. 2.6. P2P Online Shooter (Rəşad Məmmədov, 2024)

1. Exploiting machine intelligence - süni intellekdən istifadə edərək haqsız digər oyunçular üzərində haqsız üstünlüyün əldə edilməsi.Məsələn Chess oyununu nəzərdən keçirək.Bu oyunda 2 oyunçu şahmat oynayır.Burada oyunçulardan biri süni intellekdən istifadə edərək oyunu qazana bilər(Şək. 2.7.).
2. Modifying client infrastructure - oyunun konfiqurasiyasında dəyişlik etmədən qrafika keyfiyyətinin aşağı salınması və ya ekran kartı sürücülərinin dəyişdirilməsi ilə edilən fırıldaqlardır.Məsələn Battle Royal oyunu olan Pubg-da oyunçular yerə uzanaraq özlərini otların arasında gizlədə bilirlər.Əgər oyunçular qrafikanı artırırsa həmin oyunçu üçün otların sıxlığı artır.Qrafikanın keyfiyyətini azaltdıqda isə otların azalması nəticəsində oyunçu, yerə uzanan digər oyunçuları rahatlıqla görə bilir.

3. Exploiting a bug or loophole - oyunun kodunun və ya məlumatlarının dəyişdirilmədən, oyunda yaranan xətalardan və ya döngülərdən istifadədir.
4. Internal misuse - insayderlər tərəfindən törədilən fırıldaqçılıqdır.Məsələn oyunun proqramçıları imtiyazlarından istifadə edərək özü üçün müxtəlif haqsız üstünlüklər əldə edə bilər.

Şək. 2.7. Exploiting machine intelligence (Rəşad Məmmədov, 2024)

### 3.1.3. Oyunçulara və oyunun inkişaf infrastrukturuna hədəf alan zərərli proqramlar.

Karona pandemiyasından sonra oyunçuların sayı artmış və oyun sənayesidə bu zamandan sonra sürətli şəkildə yüksəlmişdir.Bunun nəticəsində oyunçuları və oyun infrasturukturunu hədəf alan hücumların sayı və bu infrasturuktur üçün yaradılan zərərli proqramlarında sayı artmışdır.Bu zərərli proqramlara aşağıdakılar aiddir:

Downloader - oyunları yükləmək üçün istifadə edilən 3-cü tərəf proqramlar

AdWare - oyunları yüklədikdən sonra bizim razılığımız olmadan reklam göstərən proqramlar

Trojan - hədəfin cihazında özünü qanuni proqram kimi göstərərək, məlumatları oğurlayan, cihaz üzərində komanda icrasına icazə verən zərərli proqramlar

Exploit - sistemin boşluqlarında istifadə edərək zərərli kod inyeksiya edən proqram

Oyunçuların oyunların crack(lisenziyanın sındırılması) edilmiş formada etibarsız mənbələrdən yüklədikdə, onların həssas məlumatları bu oyunların lisenziyasını sındıranlar tərəfindən oğurlana bilər. Çünki xakerlər lisenziyası sındırılmış bu oyunlara

“Şəkil 2.6”-da göstərilən zərərli proqramları inyeksiya edə bilər.Bu cür zərərli proqramlardan qorunmaq üçün oyunları sadəcə lisenziyalı mənbələrdən yükləmək lazımdır.

### 3.2. Oyunların inkişaf prosesi zamanı həsas nöqtələr

#### 3.2.1. Oyun kodunda zəifliklərə səbəb olan etibarlı olmayan kodlaşdırma təcrübələri.

Oyunların inkişafı zamanı secure coding texnikalarından istifadə edilməsi önəmli məsələlərdən biridir.Bunun təmin edilməsi proqramçıların üzərinə düşür.

1.Bu texnikalardan biri olan İnput Validation(Giriş Doğrulaması) və İnput Satization(giriş sanitarizasiyası)-dır.İstifadəçidən alınan məlumat həmişə doğrulanmalıdır.Yəni əgər oyun daxilində istifadəçidən məlumat daxil etməsini istədiyimiz hər hansısa bir forum varsa, burada daxil ediləcək hərflər,rəqəmlər və xüsusi işarələr doğrulanmalı və sanitarizasiya olunmalıdır.Giriş doğrulamasına nümunə olaraq, rəqəm daxil etməli olduğumuz hissəyə sadəcə rəqəm daxil edə bilməyimizi deyə bilərik.İnput sanitarizasiyası isə $><‘’/\; kimi işarələrin və xüsusi özlərin daxil edilməsinin qarşısının alınmasıdır.

2.Server tərəfindən istifadəçiyə qaytarılan error məlumatları arasında gizli informasiyanın verilməməsinə diqqət yetirmək lazımdır.

3.Proqramçılar məlumatların təhlükəsiz şəkildə, yəni şifrələnmiş şəkildə saxlanılmasını, şifrələrin isə heşlənmiş şəkildə saxlanılmasını təmin etməlidir.

4.Oyun daxilində isifadə edilən prokolların təhlükəsiz olduğuna əmin olunmalıdır.SSL/TLS sertifikatlarından istifadə edərək bunu təmin etmək mümkündür.

#### 3.2.2. Üçüncü tərəf kitabxanalarda və asılılıqlarda olan zəifliklər.

Oyunların yaradılması zamanı müxtəlif 3-cü tərəf kitabxanalardan və proqramlardan istifadə edilə bilər.Bunların istifadəsi oyunların yaradılmasındakı səmərərəliyi önəmli dərəcədə artırır.Öncəki fəsillərdə oyunları yaratmaq üçün oyun matorlarının istifadəsi ilə bağlı məlumat verilmişdir.İstifadə edilən bu oyun motorlarında boşluqların və zəifliklərin

ola bilməsi mümkündür.Əgər oyunumuz onlayn olacaqsa, yerləşdirildiyi serverlərdə təhlükəsizlik tələblərinə riayət olunmalıdır.Oyunun proqramçıları tərəfindən oyun kodlanarkən istifadə edilən kitabxanalara diqqət edilməlidir.Kitabxananın hər hansısa bir boşluğunun olub olmamasının araşdırılması, boşluğu olan metod və funksiyalardan istifadə edilməməsi vacibdir.

### 3.2.3. Oyun paylama platformalarında qeyri-kafi təhlükəsizlik tədbirləri.

İlk öncə oyun paylama platformalarının nə olduğunu anlayaq.Oyun paylama platformaları oyunların rəqəmsal formada satışı üçün xidmət verən platformalardır.Oyun şirkətləri və müstəqil oyun tərtibatçıları hazırladıqları oyunları bu platformalar üzərindən rəsmi olaraq satışa çıxardırlar.Bu platformalar oyunlara nəzarəti, tənzimləməri, yeniləmələri və müxtəlif kampaniyaları və endirimləri tətbiq etməyə imkan verir.Belə platformalara misal olaraq hamı tərəfindən bilinən Steam,Epic Game store kimi platformaların adlarını çəkə bilərik.Oyun satışı üçün platforma seçərkən təhlükəsizlik məsələlərinə diqqət etmək lazımdır.Oyunların bu platformalar üzərindən satılması zamanı aşağıdakı təhtidlərlə qarşılaşıla bilər:

1.Düzgün olmayan authentikasiya və authorizasiya

2.Hesab təhlükəsizliyinin düzgün şəkildə təmin edilməməsi

3.Şifrələməsi olmayan qoşulma protokollarından istifadə

Belə bir nümunə çəkək.Steam platforması oyunların satışı zamanı hər ölkə üçün fərqli qiymət siyasəti tətbiq edir.Məsələn Azərbaycanda 10 dollara satılan bir oyun,Türkiyədə 3 dollara satılır.Bu qiymət siyasətləri ölkənin iqtisadi vəziyyətinə və platforma istifadəçilərin sayına uyğun olaraq təyin edilir.Azərbaycandan olan bir istifadəçi Türkiyə hesabı açaraq və Türkiyədən alınan pul kisəsi kodları ilə Steam-ı aldada, yəni 10 dollarlıq bir oyunu 3 dollara ala bilir.Bu isə həm platforma üçün, həmdə oyunun tərtibatçıları üçün pul itkisi deməkdir.

### 3.2.4. Tərtibatçıları və oyunçuları hədəf alan social mühəndislik hücumları.

Sosial mühəndislik hücumları insanların aldadılması ilə baş verən hücumlardır.Bu hücumları insanların heklənməsidə adlandıra bilərik.Aşağıda sosial mühəndislik hücumlarının bir neçəsi göstərilmişdir:

Phishing and Spear Phishing - phishing bir nəfərə, spear phishing isə bir qrupa edilən hücumdur.Məsələn elektron poçt üzərindən bütün oyun tərtibatçılarına şirkət sahibi adından, şirkətdə yeni qaydaların tətbiq edildiyi və linkə tıklayaq bu qaydalarla tanış ola biləcəyinizi bildirən bir epoçt göndərilə bilər.

Credential Harvesting - Bu hücumda hesabınızdakı bəzi detalların yanlış olduğunu bildirilir və vəziyyəti həll etmək üçün sizə link təqdim edilir. Linkə daxil olduğunuz zaman o sizə daxil olmaq üçün saxta internet səhifəsi verir. Bu internet səhifəsində məlumatlarını daxil etdiyiniz zaman bu məlumatlar xakerlər tərəfindən oğurlanır.

Whaling - yüksək imtiyazlı şəxsələrə edilən sosial mühəndislik hücumudur.Məsələn Chief Executive Officer (CEO)-lara edilən hücumlar.

"Vishing" (Voice Phishing) - səsli əlaqə vasitəsilə baş verən hücumlara deyilir.Telefon zəngi vasitəsilə edilən hücumu buna misal olaraq deyə bilərik.

SMiShing - SMS mətn mesajlarından istifadə edir. Onlar sizdən ya vebsayta baş çəkməyinizi, ya da yüksək qiymətli zəng olduğu ortaya çıxan telefon nömrəsinə zəng etməyinizi xahiş edəcəklər.

Tailgating - kiminsə sizin arxanızca gələrək, sizin vəsiqə ilə keçdiyiniz yerlərdən arxanızca keçməsi.Bu hücumların qarşısını almaq üçün mantrap qapılardan istifadə etməliyik.

Shoulder Surfing - kiminsə arxasında dayanaraq onun həssas məlumatlarının ələ keçirilməsi ilə nəticələnən hücumlardır.Məsələn internet klublarda oyuna girmək üçün

istifadə etdiyiniz istifadəçi adının və şifrəni yazarkən klaviaturanıza baxılaraq öyrənilməsi.

Scarcity - insanların qarşısına vaxt qoyaraq edilən hücumlardır.Məsələn hər hansısa bir oyun almaq istəyirsən və sayta daxil olduğun zaman məhsulun endirimdə olduğunu və 30 dəqiqə ərzində endirimin bitəcəyini yazan mesajla qarşılaşırsan.

## 3.3. Oyunlaın yazılması zamanı təhlükəsizliyini təmin etmək üçün strategiyalar və mövcud təcrübələr

### 3.3.1. Həssas məlumatları qorumaq üçün şifrələmə metodlarından istifadə.

Həssas məlumatlar dedikdə, istifadəçilərin adları, şifrələri, kart məlumatları və ya digər şəxsi məlumatları nəzərdə tutulur.Qorunmanı təmin etmək üçün şifrələmə və heşinqdən istifadə edə bilərik.Gəlin ilk öncə şifrələmədən danışaq.Server-Client formasında işləyən onlayn oyunlarda şifrələmənin istifadəsi vacibdir.

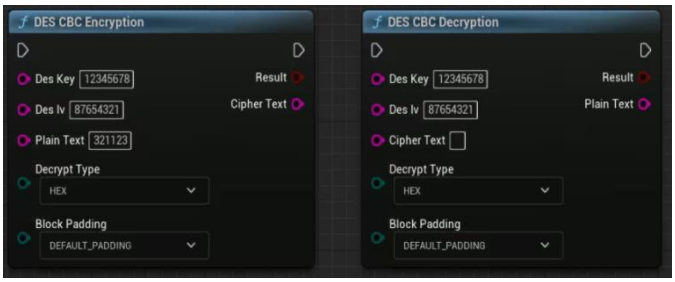
2 şifrələmə metodu mövcuddur: Simmetrik şifrələmə və assimmetrik şifrələmə.

Simmetrik şifrələmə sadəcə private key və ya shared key adlanan key-dən istifadə edir.Simmetrik şifrələmə sürətli lakin nisbətən az təhlükəsiz metoddur.Bit sayı nə qədər çox olarsa sürət o qədər aşağı, və o qədər çox təhlükəsiz olur.Simmetrik şifrələmə datanı blok şəklində ötürərkən bloklarıda şifrələyir.Bu şifrələmə metodunda ən çox istifadə edilən metod Diffie Hellman-dır(DH).DH-nin istifadəsinin məqsədi simmetrik şifrələnmiş data keçərkən təhlükəsiz tunel yaratmaqdır.

Diffie Hellman(DH) - bu barədə öncələrdə danışmışıq.Simmetrik şifrələmə üçün istifadə edilir.Məqsədi simmetrik şifrələnmiş data keçərkən təhlükəsiz tunel yaratmaqdır.Datanı ayrıca şifrələmir.DH, Internet Key Exchange (IKE) daxilində işlədiləcək key-lər yaradır və L2TP/İPSec VPN üçün secure session yaratmaq üçün UPD 500 portunu istifadə edir. İKE - İPSec də 2 -cihaz arasında təhlükəsiz və authentikasiya olunmuş kommunikasiya kanalı qurmaq üçün istifadə edilir.2 mərhələli protokoldur.1-ci mərhələdə 2 cihaz arasında

DH istifadə edilərək kanal kurulur.2-ci mərhələdə şifrələmə və heşinq parametrləri təyin edilir.Bundan sonra artıq 2 cihaz bir biri ilə təhlükəsiz şəkildə məlumat ötürə bilər.

Oyun mühərriklərində şifrələmə və deşifrələmənin istifadəsində Unreal Engine-də istifadə olunan CryptoppPluginLibrary plugin-i nümunə göstərmək olar. Bu plugin-dən istifadə edərək Blueprint üzərində yazılmış kodun şifrələmə və deşifrələməsini həyata keçirə bilərik. Plugin daxilində AES, DES, 2DES, 3DES şifrələmə və deşifrələmə metodlarından istifadə olunur(Şək. 2.8.).




Şək. 2.8. Blueprint üzərində yazılmış kodun şifrələməsi (Rəşad Məmmədov, 2024)

Assimmetrik şifrələmə Public Key İnfrastructure(PKİ) kimi bilinən 2 key istifadə edir: Public və Private Key.Bu şifrələmə prosesində ilk olaraq hər iki tərəf birbirilə public açar mübadiləsi edir. Şifrələnmə alıcının public keyi ilə olur.

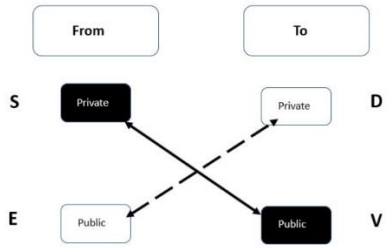
4 mərhələ var(Şək. 2.9.):

S: Sign (digital signature)

E: Encryption

D: Decryption

V: Validation




Şək. 2.9. Şifrələnma alıcının public keyi (Rəşad Məmmədov, 2024)

From To-ya məlumat göndərirsə, o zaman öz private key-i ilə göndərilən məumatı imzalayır,daha sonra qarşı tərəfin pubic keyi ilə şifrələyir.Data gedib alan tərəfə çatdıqda isə alan tərəf öz private keyi ilə datana açır və göndərən tərəfin public keyi ilə datanın tamlığını doğrulayır.

Rəqəmsal imza şəxsin identikliyini bildirən heşinq məntiqində olan bir şifrədir.Yəni göndərilən datanı göndərən öz private keyi ilə imzalayır və alan isə göndərənin public keyi ilə datanın tamlığını verifikasiya edir.

Ümumiləşdirsək, Rəqəmsal imza mesajı göndərənin həqiqiliyini və mesajın tamlığını yoxlamaq üçün istifadə edilir, Assimmetrik şifrələmə isə məxfiliyi təmin etməklə təhlükəsiz əlaqə üçün istifadə edilir.

Assimmetrik Alqoritmlər:

Rivest, Shamir and Adelman(RSA) - 1024,2046,3072,4096 bitlik şifrələmədir.

Digital Signature Algorithm(DSA) - bu key-lər rəqəmsal imza üçün istifadə edilir. 512,1024,2046 bitlik olur.

Heşing - çevrilə bilməyən birtərəfli funksiyadır.Riyazi hesablamalar nəticəsində əldə edilir.Məlumatın heşlənməsi zamanı bit sayı nə qədər çoxdursa, o qədər təhlükəsizdir.Bit sayının az olmasının üstünlüyü isə belə alqoritmlərin daha sürətli işləməsidir.Müxtəlif heşinq alqoritmləri var:

Secure Hash Algorithm Version 1(SHA1) - 160 bit, SHA2 - 256 bit, SHA3 - 512 bit və MD5 -128 bit istifadə edən alqoritmlərdir.Heş dəyər həmdə message digest adlanır.2 eyni mətn,eyni algoritmlə heşlənərsə nəticə eyni olacaq.

RİPEMD - 128 bitlik heş alqoritmidir..RİPEMD-160, RİPEMD-256, RİPEMD-320 kimi versiyaları var.

Unreal Engine mühərriki daxilində hashing istifadəsinə SHA256 HASH FUNCTION plugin-i nümunə göstərmək olar. Plugin daxilində həm String, həm faylı, həm də massiv daxilində olan elementlərin hashing-ni həyata keçirmək mümkündür.

### 3.3.2. DDoS hücumlarından qorunmaq üçün şəbəkə təhlükəsizliyi tədbirlərinin tətbiqi.

DDoS hücumlarından qorunmaq üçün aşağıdakı həllər tətbiq edilə bilər:

1.DDoS qoruma xidmətləri - DDoS hücumunu aşkar edən və onu bloklayan və ya yüngüllədən xidmətlərdir.

2.Şəbəkə trafikinin monitorinqi - Oyun serverinin və şəbəkənin trafikini analiz edərək anomal trafiki aşkar edə bilərsiniz.Bunun sayəsində trafikin mənbəyini bloklaya bilərsiniz.

3.DDoS hücumlarına qarşı, DDoS Protection dəstəkli firewall-lardan istifadə edilə bilər.

4.Hər serverə gələcək paket və trafik limiti təyin edin.Bir İP adresdən gələcək paket sayını limitləyin.

Əgər oyun serverində gözlənilməz artıq resurs istifadəsi başlayırsa, bu DDoS hücumunun nəticəsi ola bilər.

### 3.3.3. Fırıldaqçılıq aşkarlama mexanizmlərinin və icazəsiz girişdən qorunma texnologiyalarının inteqrasiyası.

Adı çəkilən mexanizmlərin inteqrasiyası oyun platformaları üçün əhəmiyyətli bir məsələdir.Bu mexanizmlərə aşağıdakıları aid etmək olar:

1.Süni İntellektin istifadəsi - Oyunçuların davranışlarını monirotinq və analiz edərək fırıldaqçılıq edən oyunçuları süni intellektin vasitəsilə tapmaq mümkündür.

2.Reporting - Onlayn oyunda digər oyunçular hər hansısa bir oyunçunun fırıldaqçılıq etdiyini fikirləşikləri zaman həmin oyunçunu Report edə bilərlər.Məsələn bir zamanların ən məhşur oyunlarından biri olan Point Blank-ı buna nümunə çəkə bilərik.Bu oyunda əgər biri sizin fırıldaqçılıq etdiyinizdən şübhələnirdisə, o zaman sizin oyundan atılmanız üçün səsvermə keçirə bilirdi.Əgər digər oyunçularda sizin fırıqdaqçılıq etdiyinizi düşünürdülərsə və yetəri qədər səs toplanırdısa, siz oyundan atılırdınız.

3. Xüsusi proqramlardan istifadə edərək fırıldaqçılıq üçün istifadə edilən kodları aşkarlamaq mümkündür.

4.Yamaq - oyun tərtibatçıları baş verən fırıqdaqçılıq hallarının qarşısını almaq üçün internetə sızdırılmış və çoxluq tərəfindən bilinən metodları analiz etməli və qarşısını almaq üçün yeniləmə və yamaq tətbiq etməlidir.

### 3.3.4. İnkişaf etdiricilər və maraqlı tərəflər arasında kibertəhlükəsizlik və ən yaxşı təcrübələr barədə məlumatlılığın artırılması.

Kiber hücumlardan qorunmağın ən yaxşı yolu oyun tərtibatçılarına kiber təlimlərin verilməsidir.Çünki hücumların səbəbi olaraq insat faktoru bu mövzuda önəmli rol oynayır.Buna görədə tərtibatçılar və oyunun yerləşdirildiyi serverləri və şəbəkəni idarə edən İT əməkdaşları mütəmadi olaraq kiber təlimlərdən keçirilməli və tərtibatçılar “Secure Coding” məntiqi və ən son hücumların metodikaları haqqında məlumatlı olmalıdırlar.

# NƏTİCƏ

Magistr dissertasiya mövzusu əlaqədar tədqiqat işlərin təhlil edilməsi ilə başlamış, qarşıya qoyulmuş bir neçə məsələ istiqamətində tədqiqatlar aparılmaqla aşağıdakı nəticələrlə yekunlaşmışdır:

- Oyun proqramlaşdırmasında ikiqat material probleminin köhnə texnologiyalar üzərində istifadəsi və alternativləri, yeni texnologiyalar ilə müqayisəsi;
- İşıqlandırma texnologiyalarının aparat təminatı ilə birlikdə inkişafı zamanı yeni texnologiyalara təkmilləşməsi nəticəsində yaranan Lumen texnologiyasının analizi aparılmışdır;
- Simulyasiya proqramlarında və oyun proqramlaşdırmasında süni intellekt texnologiyasının istifadə yerləri işləmə prinsiplərinin analizi aparılmışdır;
- Günümüz texnologiyasında oyun proqramlaşdırmasında ən alternativ optimizasiya həllərinə yönəlik analizlər aparılmışdır;

# İSTƏDƏBİYYAT

Aleksi K, Creating Foliage for Video Games. (2023). 13 p.

Andrew S, An Introduction to Unreal Engine 4. (2016). 193 p.

Brain K, Epic Games, Real Shading in Unreal Engine 4. (2014)

Devkan K, Hasan K, Serkan D, Açık Kaynak Kodlu 3D Oyun Motorları. (2012). 2 p.

Eden L, Real Time Crowd Flow. (2018). 8 p.

Evgeny P, Unreal Engine 4 for Automation. (2021). 5 p.

Ferus Abu-Abed, Sergey Zh, New Game Artificial Intelligence Tools for Virtual Mine on Unreal Engine. (2023)

Henk V, Wilhelm O, Unreal Engine 5 Character Creation, Animation, and Cinematics. (2022). 131 p.

Jaakko S, Interactive Architectural Visualization Using Unreal Engine. (2022). 15 p.

Jani H, Game Development with Unreal Engine 4. (2020). 10 p.

Jiashuai Du, Liping Su, Dong Chen, Innovative Design for Interactive Display of Ancient Architectural and Cultural Heritage using UE5 Virtual Engine. (2023). 9 p.

Julia V, Comparison between Vizard VR Toolkit and Unreal Engine 4 as platforms for virtual experiments in pedestrian dynamics using the Oculus Rift. (2015). 3 p.

Katax E, Devin Sh, Unreal Engine Physics Essentials. (2015). 28 p.

Manuel Drago Díaz-Alemán, Esteban M. Amador G, Elisa Díaz-González, Jorge de la Torre-Cantero, Nanite as a Disruptive Technology for the İnteractive Visualisation of Cultural Heritage 3D Models. (2023).

Marcos R, Brenden S, Blueprint Visual Scripting for Unreal Engine 5. (2022). 121 p.

Mehman Sh, Kamil C, Farid A, Unreal Engine 4. (2023). 94-100 p.

Mehman Sh, Kamil C, Farid A, Unreal Engine 5. (2024). 223-226 p.

Miro V, 3D Game Environment in Unreal Engine 4. (2014). 20 p.

Mittring, Martin, and Bryan Dudash, “The Technology Behind the DirectX

Muhammad M, Applied Artificial İntelligence in 3D-game (HYSTERİA) using UNREAL ENGİNE 4. (2018). 3 p.

Nataliia F, Maksim P, Iryna B, Svitlana V, Yaroslava D, Research On Calculation Optimization Methods Used In Computer Games Development. (2023). 38 p.

Pelechano G, Núria, Andújar G, Carlos A, Synthesising character animation for real time crowd simulation systems in Unreal Engine. (2018). 13 p.

Reece A, Implementing Reinforcement Learning in Unreal Engine 4 with Blueprint. (2017). 8 p.

Roope P, Virtual Reality Multiplayer in Unreal Engine 5 with C++. (2022). 47 p.

Shivang S, Bernard T, Helen C, AWideAreaMultiview Static Crowd Estimation System Using UAVand3DTraining Simulator. (2021). 17 p.

Stephen R, LODs and Nanite within Unreal Engine 5: The Future of 3D Asset Creation for Game Engines. (2024). 24 p.

Tianshu Li, Real–Time Performance Comparison Of Environments Created Using Traditional Geometry Rendering Versus Unreal Nanite Technology In Virtual Reality. (2024). 35 p.

Tri Hai Ha, Game development with Unreal Engine. (2022). 14 p.

Unreal Engine Samaritan Demo, Game Developer Conference 2011.

Ying Wu, A New Exploration based on Unreal Engine4 Particle Effects of Unreal Engine in 3D Animation Scenes. (2021). 692 p.