Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
(повна назва)

Кафедра медіаінженерії та інформаційних радіоелектронних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*
(позначення документа)

Дослідження методів створення 3D об'єктів

(тема)

Виконав:
студент 2 курсу, групи *СТМм-21-1*
Ігор КРАМСЬКИЙ
Спеціальність *171 Електроніка*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системи, технології і комп'ютерні засоби мультимедіа*
(повна назва освітньої програми)

Керівник *проф. Олег СИТНІК*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Володимир КАРТАШОВ
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформації
Кафедра Медіаінженерії та інформаційних радіоелектронних систем

Рівень вищої освіти другий (магістерський)
Спеціальність 171 Електроніка.
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма "Системи, технології і комп'ютерні засоби мультимедіа"

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Крамському Ігорю Андрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження методів створення 3D об'єктів*
затверджена наказом по університету від " *21* " *11* *2022 р. №* *1503 СТ*
2. Термін подання студентом роботи *08.12.2022 р.*
3. Вихідні дані до проекту (роботи)
1. Проаналізувати принципи створення 3D-моделей
2. Розробити 3D - модель ігрового персонажу
3. Провести аналіз методу створення 3D-моделі ігрового персонажу

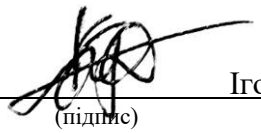
4. Перелік питань, що потрібно опрацювати в роботі
ВСТУП
1. Аналітичний огляд принципу створення 3D-моделей
2. Аналітичний огляд та аналіз існуючих видів 3D-моделювання
3. Аналіз етапів створення 3D-моделі
4. Розробка 3D-моделі ігрового персонажу
ВИСНОВКИ
ПЕРЕЛІК ПОСИЛАНЬ
ДОДАТКИ

5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій:
1 Актуальність роботи; 2 Сфери застосування 3D-графіки; 3 Що таке 3D графіка; 4 Історія створення 3D-моделювання; 5 Створення 3D моделі; 6 Методи створення 3D-моделей: твердотільне моделювання; 7 Методи створення 3D-моделей: скульптинг; 8 Методи створення 3D-моделей: полігональне моделювання; 9 Методи створення 3D-моделей: воксел; 10 Методи створення 3D-моделей: сплайнове моделювання; 11 Методи створення 3D-моделей: nurbs моделювання; 12 Створення 3D-моделі персонажа: референс; 1 3 Створення 3D-моделі персонажа: блокінг; 14 Створення 3D-моделі персонажа: деталізований драфт; 15 Створення 3D-моделі персонажа: деталізована highpoly; 16 Створення 3D-моделі персонажа: ретоплена lowpoly та UV; 17 Створення 3D-моделі персонажа: запікання; 18 Створення 3D-моделі персонажа: текстурування та рендер; 19 Висновки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Аналітичний огляд принципу створення 3D-моделей	21.11.22–28.11.22	
2.	Аналітичний огляд та аналіз існуючих видів 3D-моделювання	21.11.22–28.11.22	
3.	Аналіз етапів створення 3D-моделі	23.11.22–02.12.22	
4.	Розробка 3D-моделі ігрового персонажу	01.12.21–05.12.22	
5.	Графічна частина роботи	07.12.21–08.12.22	
6.	Перевірка керівником	07.12.22-08.12.22	
7.	Перевірка на академічний плагіат	08.12.22-09.12.22	
8.	Перевірка завідувачем кафедри, рецензування	09.12.22-10.12.22	

Дата видачі завдання *21.11.2021 р.*

Студент  Ігор КРАМСЬКИЙ
(підпис)

Керівник роботи _____ *проф. Олег СИТНІК*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи має: 111 с., 99 рис., 6 табл., 2 додатків, 47 джерел.

ЛОУПОЛІ, ХАЙПОЛІ, ПОЛІГОН, РОЗГОРТКА, ЗАПІКАННЯ, ВЕРТЕКС, МОДЕЛЬ, НОРМАЛЬ, ОБ’ЄКТ, СІТКА, ПАЙПЛАЙН, ПІКСЕЛІ, ТРИКУТНИКИ, РЕНДЕР

Об'єкт дослідження – методи 3D моделювання.

Предмет дослідження – особливості типів 3D моделювання, роботи графічного процессора та етапів створення 3D моделі .

Мета кваліфікаційної роботи роботи – розробити 3D модель.

Методи дослідження – теоретичний аналіз, статистична обробка даних, аналіз роботи GPU.

У даній роботі наведено класифікацію методів 3D моделювання, типів полігонів, класифікації полігональних сіток, детальний аналіз роботи графічного процесору при рендері полігонів, методи оптимізації моделей, огляд принципу блокінгу моделей, наведені правила ретопології, UV-розгортко та запікання текстурних карт, розробка 3D моделі персонажа, опис функціонування фізично коректних текстур, розроблено алгоритм створення 3D моделі.

ABSTRACT

he explanatory note of the qualification work has: 111 pages, 99 figures, 6 tables, 2 appendices, 47 sources.

LOW POLY, HIGH POLY, POLYGON, SNAP, BAKE, VERTEX, MODEL, NORMAL, OBJECT, GRID, PIPELINE, PIXELS, TRIANGLES, RENDER

The object of research - 3D modeling methods.

The subject of the study - the peculiarities of the types of 3D modeling, the operation of the graphics processor and the stages of creating a 3D model.

The purpose of the qualification work - to develop a 3D model.

Research methods – theoretical analysis, statistical data processing, GPU performance analysis.

This work provides a classification of 3D modeling methods, types of polygons, classification of polygon meshes, a detailed analysis of the graphics processor when rendering polygons, methods of optimizing models, an overview of the principle of model blocking, the rules of retopology, UV scanning and baking of texture maps, development of a 3D character model , a description of the functioning of physically correct textures, an algorithm for creating a 3D model was developed.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забеспечення;

GPU - graphics processing unit;

PBR - physically based rendering.

3D – тривимірна графіка

2D – двовимірна графіка

ШВЛ – штучна ветиляція легень

Т.і. – таке інше

API – Application Programming Interface

ВУ – Вершинне уявлення

FPS – frames per second

UV – осі координат текстури U та V

ЦП – центральний процесор

N-gon – полігона з п’ятьма або більше вершинами

ЧПК – числове програмне керування

NURBS – Non-Uniform rational B-spline

AAA – A lot of time, A lot of resources, A lot of money

CAD – Computer-aided design

AO – Ambient occlusion

ID - індифікатор

ЗМІСТ

ВСТУП

3D моделювання - це проектування тривимірної моделі за заздалегідь розробленим кресленням або ескізом. Для побудови об'ємної моделі предмета використовуються спеціальні програмні продукти візуалізації та апаратні пристрої у вигляді комп'ютерів, планшетів та оргтехніки. При моделюванні важливим етапом є рендеринг - перетворення чорнової варіації моделі на приємний для очей формат.

3D моделювання відіграє важливу роль у житті сучасного суспільства. Сьогодні воно широко використовується у сфері маркетингу, архітектурного дизайну та кінематографії, ігрових технологіях, не кажучи вже про промисловість. 3D -моделювання дозволяє створити прототип майбутньої споруди, комерційного продукту в об'ємному форматі для ректами або промо акцій, проведенні презентації та демонстрації будь-якого продукту чи послуги, с медицині та навчанні для наглядності, у будівництві та складних промислових процесах. Важливу роль 3D моделювання відіграє під час реставрації та збереженні історичних пам’яток та артефактів.

На данний момент 3D моделювання використовуєть таких областях:

- створення різноманітних моделей персонажів. Зазвичай це використовується при створенні мульфільмів і при проектуванні сучасних комп’ютерних відоігр;
- 3D візуалізація будівель. Цим займаються проектні організації, які бажають оцінити для замовника конструктивні особливості майбутнього об'єкта;
- Створення 3Д моделей предметів інтер'єру. Найчастіше їх виконують дизайнерські компанії з метою демонстрації естетичних властивостей представлених експозицій;
- Реклама та маркетинг. Часто потрібні нестандартні об'єкти для рекламування. Важливу складову тривимірна графіка грає під час

демонстрації будь-якої послуги. Це дозволяє справити ефектніше враження на зацікавлених осіб;

- Виготовлення ексклюзивних прикрас. Професійні художники та ювеліри використовують спеціальні програми, які дозволяють створити оригінальний та неповторний ескіз;
- Виробництво меблів та комплектуючих. Виробничі меблеві компанії часто використовують розробку тривимірної моделі для розміщення своєї продукції в електронних каталогах;
- Промислова галузь. Сучасне виробництво неможливо уявити без моделювання продукту компанії. Кожну деталь або повноцінний об'єкт простіше збирати за готовою та продуманою 3D-моделлю;
- Медична сфера. Наприклад, при проведенні пластичної операції або хірургічному втручанні, все частіше використовують тривимірну графіку для того, щоб наочно продемонструвати пацієнтові, як проходитиме процедура, і яким буде результат.

Завдяки появі та популяризації 3D-друку 3D-моделювання перейшло на новий рівень і стало необхідним як ніколи. Кожна людина вже може надрукувати намальований ним самим або завантажений з інтернету 3D-об'єкт, чи то дизайнерська модель, чи то персонаж улюбленого мультфільму. Звичайно, не всі знаються на -програмах і вміють моделювати об'ємні об'єкти. Як приклад, можна навести за останні три роки при пандемії та військовій агресії наші співвітчизники використтовують 3D принтери для виготовлення захисту для лікарів, елементів для ШВЛ, різні елементи для використання дровнів або тепловізорів та т.і. І все це показує актуальність теми моєї кваліфікаційної роботи.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРИНЦИПУ СТВОРЕННЯ 3D-МОДЕЛЕЙ

1.1 Огляд основних принципів створення 3D моделі

Розглянемо основні принципи створення 3D моделі Для того щоб намалювати тривимірний об'єкт на плоскій поверхні, необхідно створити модель цього об'єкта у просторових координатах X, Y та Z. Тобто необхідно описати кожну точку на поверхні цього об'єкта – вказати його координати. Таким чином, потрібно описати нескінченну кількість точок на поверхні об'єкта, щоб при будь-якому масштабуванні якість картинки не втрачалася. На практиці для опису тривимірної моделі використовується груба сітка, що складається з полігонів [1]. З курсу вищої математики ми знаємо, що чим детальніше сітка, тим більше полігонів і тим реалістичніша модель, як на рис. 1.1 [1]. Але тим більше потрібно ресурсів комп'ютера для розрахунку моделі та побудови 3D-графіки.

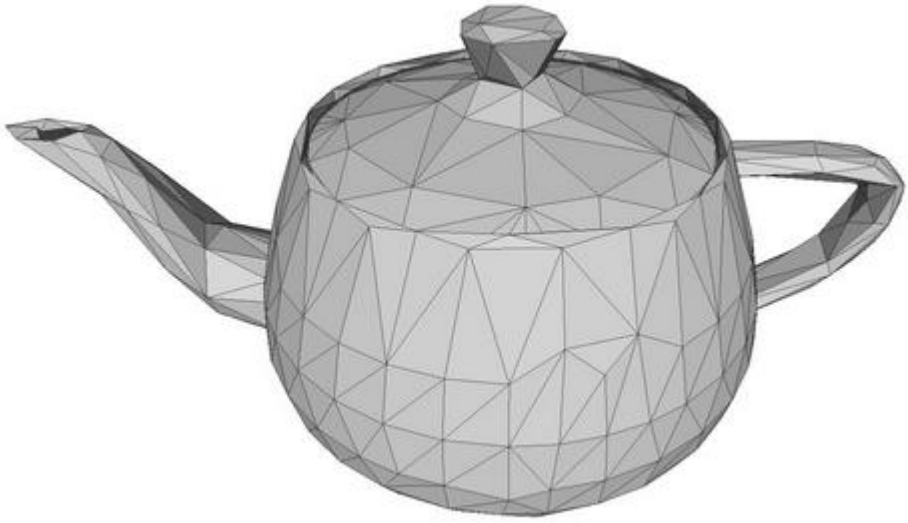


Рисунок 1.1 - Модель чайника у вигляді полігонної сітки [1]

Спочатку, коли комп'ютери та відеокарти не були такими потужними як зараз, кожен полігон ділився на трикутники, так як за допомогою

трикутника можна однозначно описати положення невеликої ділянки поверхні та обчислити на ньому такі необхідні параметри, як освітленість та відображення світла, що падає. Сукупність багатьох таких невеликих трикутників дозволяє створювати реалістичне тривимірне зображення об'єкта (рисунок 1.2 [1]). Тут і далі полігон і трикутник виступатимуть синонімами, оскільки уявити трикутник набагато простіше, ніж полігон із N вершинами.

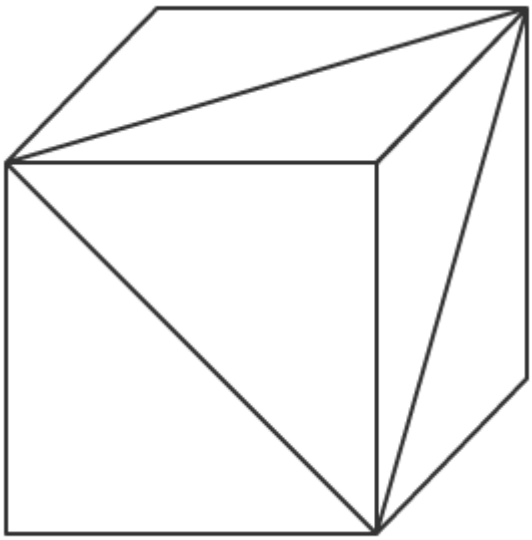


Рисунок 1.2 - Куб, складений із трикутників [1]

Таким чином, для створення тривимірної моделі об'єкта достатньо описати координати кожної вершини трикутника, щоб обчислити координати кожної точки об'єкта, навіть якщо сам об'єкт переміщається в просторі або змінюється позиція спостерігача. Вершини трикутника називаються вертексами (vertex), що з'єднують їх відрізки називаються ребрами (edge), а поверхня, укладена між відрізками, називається гранню (face). Знаючи розташування трикутника в просторі, ми можемо за законами лінійної алгебри побудувати до неї нормаль (вектор, який виходить з поверхні і перпендикулярний їй), і таким чином обчислити, як світло, що падає на межу, від джерела буде фарбувати поверхню і відбиватися від неї (рис. 1.3 [1]).

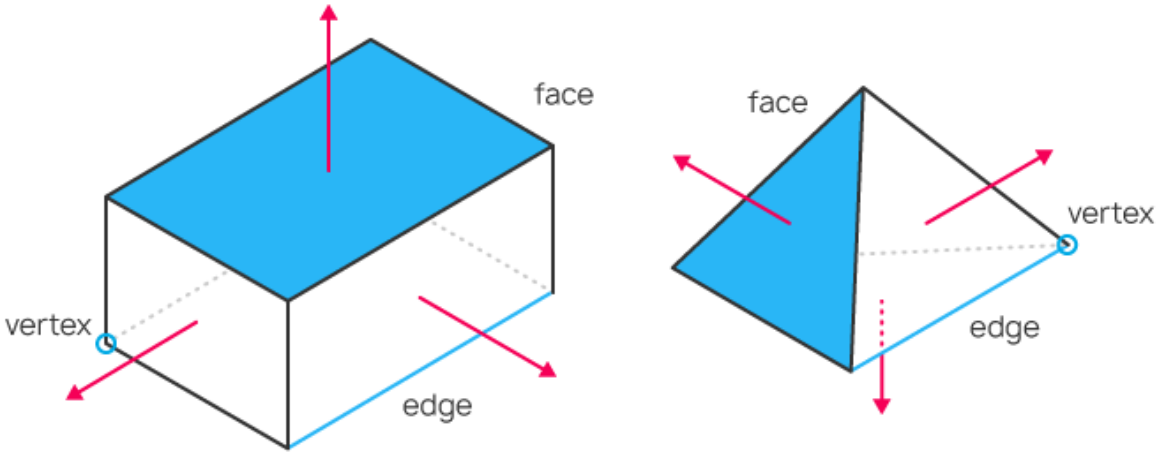


Рисунок 1.3 - Приклади простих об'єктів з вершинами, ребрами, гранями та нормалями. Нормаль – стрілка червоного кольору [1]

Створити модель об'єкта можна у різний спосіб, топологія описує те, як саме полігони формують 3D-модель (mesh). Правильна топологія дозволяє використовувати мінімальну кількість полігонів для опису об'єкта і в деяких випадках робить простішим переміщення і поворот об'єкта в просторі (рисунок 1.4 [1]).

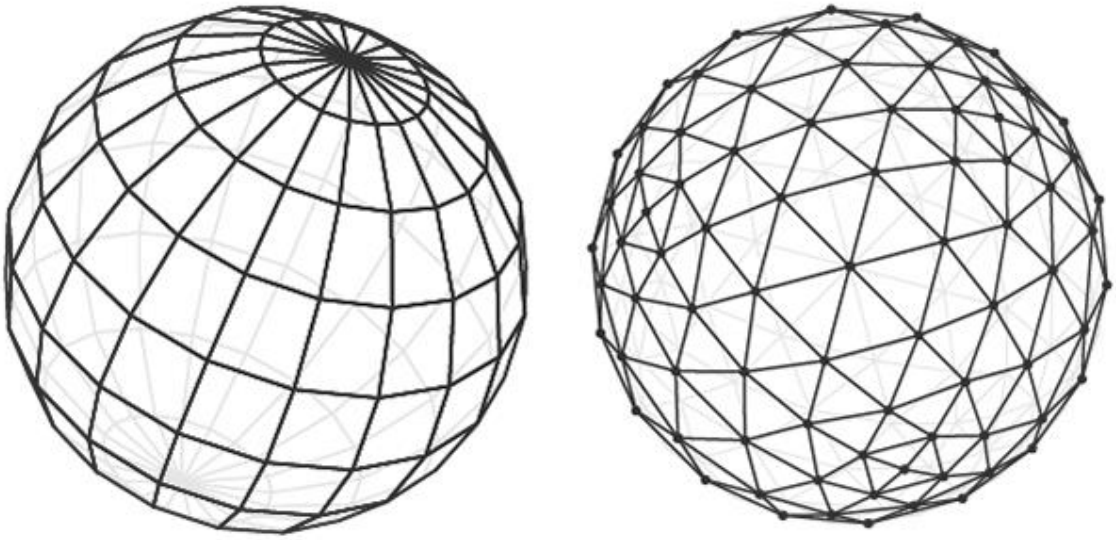


Рисунок 1.4 - Модель сфери у двох топологіях [1]

Гра світла і тіні на полігонах об'єкта робить його об'ємним, у цьому полягає завдання тривимірної комп'ютерної графіки — обчислити для кожної

точки об'єкта його розташування в просторі, розрахувати її колір і освітленість, щоб потім спроектувати на екран монітора.

Полігональна сітка - це сукупність вершин, ребер та граней, які визначають форму багатогранного об'єкта в тривимірній комп'ютерній графіці та об'ємному моделюванні. Гранями зазвичай є трикутники, чотирикутники або інші прості опуклі багатокутники (полігони), так як це спрощує рендеринг, але сітки можуть також складатися з найбільш загальних увігнутих багатокутників, або багатокутників з отворами [2].

Вчення про полігональні сітки - це великий підрозділ комп'ютерної графіки та геометричного моделювання. Безліч операцій, що проводяться над сітками, може включати булеву алгебру, згладжування, спрощення та багато інших. Різні уявлення полігональних сіток використовуються для різних цілей та програм. Для передачі полігональних сіток по мережі використовуються мережеві уявлення, такі як «потокові» та «прогресивні» сітки. Об'ємні сітки відрізняються від полігональних тим, що вони явно уявляють і поверхню та обсяг структури, тоді як полігональні сітки явно становлять лише поверхню, а не об'єм. Так як полігональні сітки широко використовуються в комп'ютерній графіці, для них розроблено алгоритми трасування променів, виявлення зіткнень та динаміки твердих тіл.

Математичний еквівалент полігональних сіток – неструктуровані сітки – вивчаються методами комбінаторної геометрії.

1.2 Огляд складових та елементів моделювання сітки

Об'єкти, створені за допомогою полігональних сіток, повинні зберігати різні типи елементів, такі як вершини, ребра, грані, полігони та поверхні. У багатьох випадках зберігаються лише вершини, ребра та або грані, або полігони. Рендерер може підтримувати лише тристоронні грані, так що полігони повинні бути побудовані з їхньої множини (рисунок 1.5 [2]). Однак багато рендерерів підтримують полігони з чотирма і більше сторонами, або

вміють тріангулювати полігони в трикутники на льоту, роблячи необов'язковим зберігання сітки в тріангульованій формі. Також у деяких випадках, таких як моделювання голови, бажано вміти створювати і три- і чотиристоронні полігони.

Вершина – це позиція разом з іншою інформацією, такою як колір, вектор нормалі та координати текстури.

Ребро – це з'єднання між двома вершинами.

Грань - це замкнута безліч ребер, в якому трикутна грань має три ребра, а чотирикутна - чотири.

Полігон - це набір компланарних (лежачих в одній площині) граней. У системах, що підтримують багатосторонні грані, полігони та грані рівнозначні. Однак, більшість апаратного забезпечення для рендерингу підтримує лише грані з трьома або чотирма сторонами, тому полігони представлені як безліч граней. Математично, полігональна сітка може бути представлена у вигляді неструктурованої сітки або неорієнтованого графа, з додаванням властивостей геометрії, форми та топології.

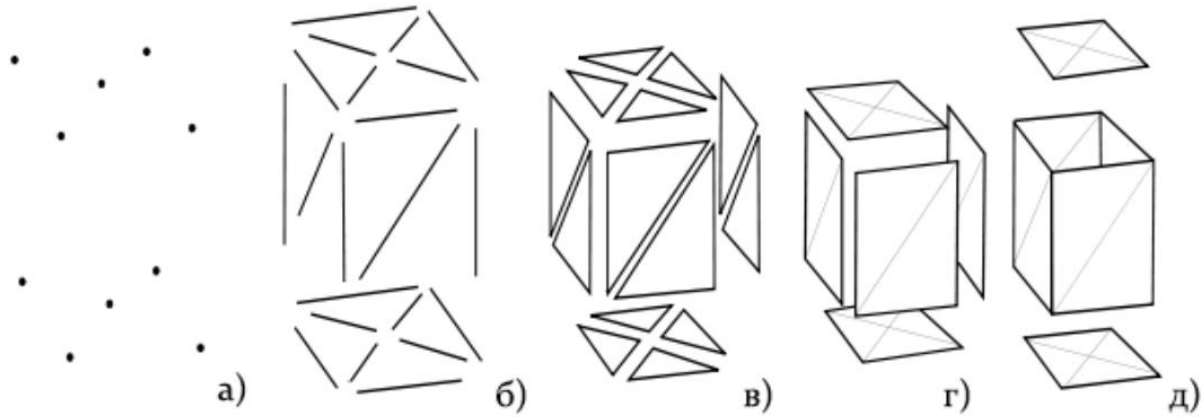


Рисунок 1.5 - Складові 3д-моделі а) вертекси (точки); б) еджі (ребра); в) фейси (грані); г) Полігони; д) Поверхні [2]

Поверхні, частіше звані групами згладжування, корисні, але з обов'язкові для групування гладких областей. Уявіть собі циліндр із кришками, такий як бляшана банка. Для гладкого затінення сторін всі

нормалі повинні вказувати горизонтально від центру, тоді як нормалі кришок повинні вказувати в +/-(0,0,1) напрямках. Якщо рендерити як єдину, затінену за Фонгом поверхню, вершини складок мали б неправильні нормалі. Тому, потрібен спосіб визначення де припиняти згладжування для того, щоб групувати гладкі частини сітки, також як полігони групують тристоронні грані. Як альтернатива наданню поверхонь/груп згладжування, сітка може містити іншу інформацію для розрахунку тих же даних, така як кут, що розділяє (полігони з нормалями вище цієї межі або автоматично розглядаються як окремі групи згладжування, або по відношенню до ребра між ними застосовується будь-яка техніка , наприклад поділ або скошування). Також, полігональні сітки з дуже високим дозволом менш схильні до проблем, для вирішення яких потрібні групи згладжування, тому що їх полігони настільки малі, що потреба в них зникає. Крім того, альтернатива існує в можливості просто від'єднання самих поверхонь від частини сітки, що залишилася. Рендерери не намагаються згладжувати ребра між безмежними полігонами.

Формат полігональної сітки може визначати інші корисні дані. Можуть бути визначені групи, які задають окремі елементи сітки та корисні для встановлення окремих подібних об'єктів для скелетної анімації або окремих суб'єктів нескелетної анімації. Зазвичай визначаються матеріали, що дозволяє різним частинам сітки використовувати різні шейдери при рендері. Більшість форматів сітки також припускають UV координати, які є окремим двовимірним уявленням полігональної сітки, «розгорнутим», щоб показати, яка частина двомірної текстури застосовується до різних полігонів сітки.

1.3 Аналіз методів представлення полігональних сіток

Полігональні сітки можуть бути представлені безліччю способів, використовуючи різні способи зберігання вершин, ребер та граней. У них входять:

Список граней: опис граней відбувається за допомогою покажчиків до списку вершин.

«Крилате» уявлення: у ньому кожна точка ребра вказує на дві вершини, дві грані та чотири (за годинниковою стрілкою та проти годинникової) ребра, які її стосуються. Крилате уявлення дозволяє обійти поверхню за постійний час, але має великі вимоги до пам'яті зберігання.

Напівреберні сітки: спосіб схожий на «крилату» уявлення, за винятком того, що використовується інформація обходу лише половини грані.

Чотириреберні сітки, які зберігають ребра, напівребра та вершини без будь-якої вказівки полігонів. Полігони прямо не виражені у виставі, і можуть бути знайдені обходом структури. Вимоги щодо пам'яті аналогічні напівреберним сіткам.

Таблиця кутів, які зберігають вершини в певній таблиці, такий, що обхід таблиці неявно задає полігони. По суті, це «віяло трикутників», що використовується в апаратному рендерингу. Подання більш компактне і більш продуктивне для знаходження полігонів, але операції їх зміни повільні. Понад те, таблиці кутів не представляють сітки повністю. Для представлення більшості сіток необхідно кілька таблиць кутів (віялок трикутників).

Вершинне уявлення: представлені лише вершини, що вказують інші вершини. Інформація про грані та ребра виражена неявно в цьому поданні. Проте, простота уявлення дозволяє проводити над сіткою безліч ефективних операцій.

Кожне з уявлень має свої переваги та недоліки.

Вибір структури даних визначається застосуванням, необхідною продуктивністю, розміром даних, операціями, які виконуватимуться. Наприклад, легше мати справу з трикутниками, ніж з багатокутниками загального вигляду, особливо в обчислювальній геометрії. Для певних операцій необхідно мати швидкий доступ до топологічної інформації, такої як ребра або сусідні грані; для цього потрібні складніші структури, такі як

«крилате» уявлення. Для апаратного рендерингу необхідні компактні, прості структури; тому API низького рівня, такі як DirectX і OpenGL зазвичай включена таблиця кутів (віяло трикутників).

1.3.1 Аналіз вершинного уявлення

Вершинне уявлення (ВУ) описує об'єкт як безліч вершин, з'єднаних коїться з іншими вершинами. Це найпростіше уявлення, але воно не широко використовується, оскільки інформація про грані та ребра не виражена явно. Тому потрібно обійти всі дані, щоб згенерувати список граней для рендерингу. Крім того, нелегко виконуються операції на ребрах та гранях.

Однак, сітки ВУ отримують вигоду з малого використання пам'яті та ефективної трансформації. Рисунок 1.6 [2] показує приклад паралелепіпеда, зображений з використанням ВУ сітки.

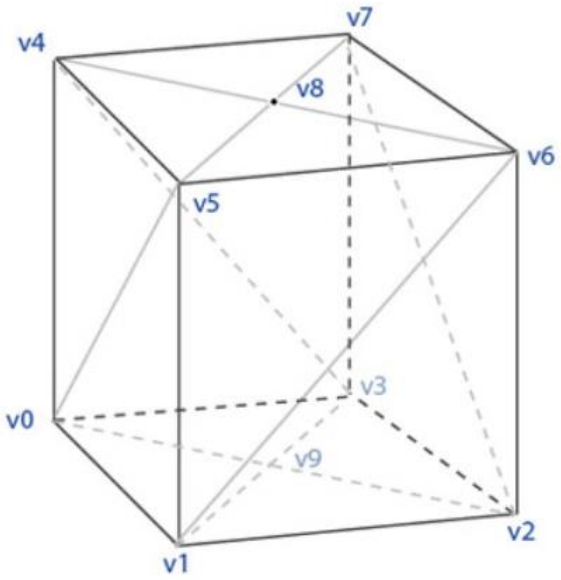


Рисунок 1.6 – Вершинне уявлення паралелепіпеда [2]

Кожна вершина індексує сусідні вершини (таблиця 1.1).

Таблиця 1.1 - Список вершин

v0	0, 0, 0	v1 v5 v4 v3 v9
v1	1, 0, 0	v2 v6 v5 v0 v9
v2	1, 1, 0	v3 v7 v6 v1 v9
v3	0, 1, 0	v2 v6 v7 v4 v9
v4	0, 0, 1	v5 v0 v3 v7 v8
v5	1, 0, 1	v6 v1 v0 v4 v8
v6	1, 1, 1	v7 v2 v1 v5 v8
v7	0, 1, 1	v4 v3 v2 v6 v8
v8	.5, .5, 0	v5 v6 v7 v8
v9	.5, .5, 1	v0 v1 v2 v3

Зауважте, що останні дві вершини, 8 і 9 зверху і знизу паралелепіпеда, мають чотири пов'язані вершини, а не п'ять. Головна система повинна справлятися з довільним числом вершин, пов'язаних з будь-якою даною вершиною [3].

1.3.2 Аналіз уявлення списку граней

Сітка з використанням списку граней представляє об'єкт як безліч граней та безліч вершин. Це найширше уявлення, будучи вхідними даними типово прийнятими сучасним графічним обладнанням.

Список граней краще для моделювання, ніж вершинне уявлення тим, що він дозволяє явний пошук вершин грані та граней оточуючих вершину. Рисунок 1.7 [2] показує приклад паралелепіпеда у вигляді сітки з використанням списку граней. Вершина v5 підсвічена, щоб показати межі, що її оточують. Зауважте, що у цьому прикладі у кожної грані обов'язково 3 вершини. Однак це не означає, що у кожної вершини одна і та ж кількість навколишніх граней.

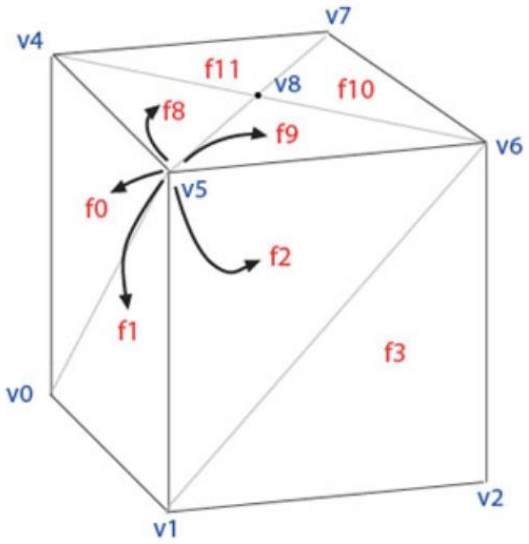


Рисунок 1.7 – Куб за список граней [2]

Для рендерингу грань зазвичай посилається в графічний процесор як безліч індексів вершин (таблиця 1.2), і вершини посилаються як позиція/колір/структури нормалей (на рисунку дано лише позицію). Тому зміни форми, але не геометрії, можуть бути динамічно оновлені, просто переславши ці вершини без оновлення пов'язаності граней (таблиця 1.3).

Таблиця 1.2 - Список вершин

v0	0, 0, 0	f0 f1 f12 f15 f7
v1	1, 0, 0	f2 f3 f13 f12 f1
v2	1, 1, 0	f4 f5 f14 f13 f3
v3	0, 1, 0	f6 f7 f15 f14 f5
v4	0, 0, 1	f6 f7 f0 f8 f11
v5	1, 0, 1	f0 f1 f2 f9 f8
v6	1, 1, 1	f2 f3 f4 f10 f9
v7	0, 1, 1	f4 f5 f6 f11 f10
v8	.5, .5, 0	f8 f9 f10 f11
v9	.5, .5, 1	f12 f13 f14 f15

Моделювання вимагає легкого обходу всіх структур. З сіткою, яка використовує список граней, дуже легко знайти вершини грані. Також список вершин містить список усіх граней, пов'язаних з кожною вершиною. На відміну від вершинного уявлення, і грані та вершини явно представлені, так що знаходження сусідніх граней та вершин постійно за часом.

Таблиця 1.3 - Список граней

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Однак, ребра не задані явно, тому пошук все ще потрібен, щоб знайти всі грані, що оточують задану грань. Інші динамічні операції, такі як розрив або об'єднання грані також складні зі списком граней.

1.3.3 Аналіз «Крилатого» уявлення сітки

Представлене Брюсом Баумгартом в 1975, «Крилате» уявлення явно представляє вершини, грані та ребра сітки. Це представлення широко використовується в програмах для моделювання для надання найвищої гнучкості динамічної зміни геометрії сітки, тому що можуть бути швидко виконані операції розриву та об'єднання. Їхній основний недолік — високі вимоги пам'яті та збільшена складність через зміст безлічі індексів.

"Крилате" уявлення вирішує проблему обходу від ребра до ребра і забезпечує впорядковане безліч граней навколо ребра. Для будь-якого заданого ребра число вихідних ребер може бути довільним. Щоб спростити це, «крилата» вистава надає лише чотири, найближчі ребра за годинниковою та проти годинникової стрілки на кожному кінці ребра. Інші ребра можна обійти поступово. Тому інформація про кожне ребро нагадує метелика, тому уявлення називається «крилатим». Рисунок 1.8 [2] показує приклад паралелепіпеда в «крилатому» поданні.

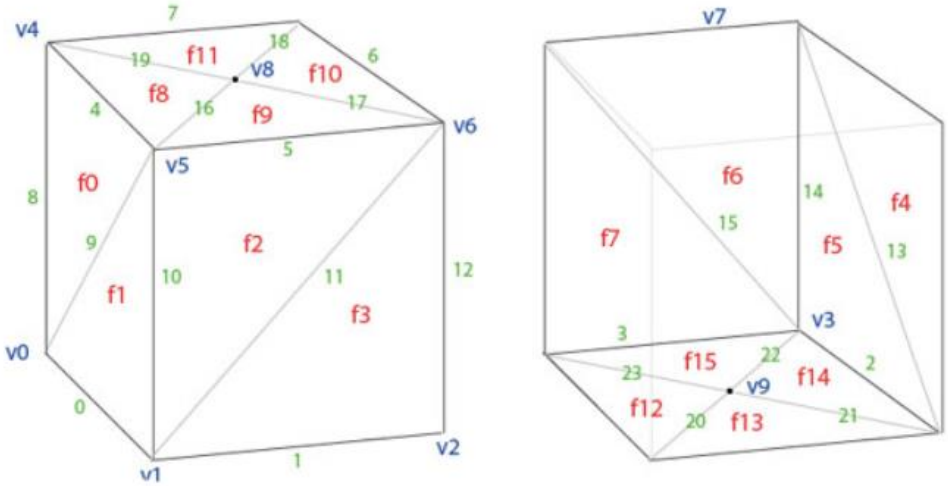


Рисунок 1.8 - «Крилате» уявлення куба [2]

Повні дані з ребра складаються з двох вершин (кінцеві точки) (табл. 1.6), двох граней (по кожну сторону) (табл. 1.4), і чотири ребра («крила» ребра) (табл. 1.5), як на рисунке 1.9 [2].

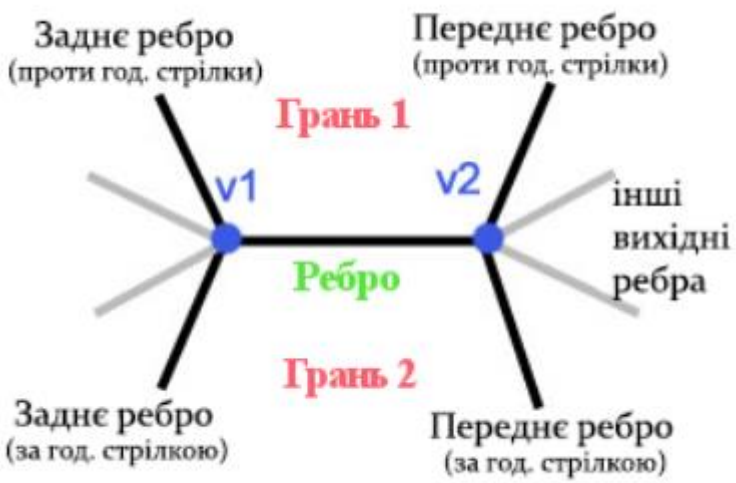


Рисунок 1.9 - Структура «крилатого» уявлення [2]

Рендеринг «крилатого» подання графічним обладнанням вимагає створення списку індексів граней. Зазвичай це робиться тільки тоді, коли змінюється геометрія. «Крилате» уявлення ідеально підходить для динамічної геометрії, такої як підрозділ поверхонь та інтерактивне моделювання, оскільки зміни сітки можуть відбуватися локально. Обхід навколо сітки, що може стати в нагоді для виявлення зіткнень, може бути ефективно виконано [4].

Таблиця 1.4 - Список граней

f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15

f8	4 16 19
f9	5 17 16
f10	618 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
15	3 22 23

Таблиця 1.5 - Список ребер

e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 14 22
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19

e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

Таблиця 1.6 - Список вершин

v0	0, 0, 0	8 9 0 23 3
v1	1, 0, 0	10 11 1 20 0
v2	1, 1, 0	12 13 2 21 1
v3	0, 1, 0	14 15 3 22 2
v4	0, 0, 1	8 15 7 19 4
v5	1, 0, 1	10 9 4 16 5
v6	1, 1, 1	12 11 5 17 6
v7	0, 1, 1	14 13 6 18 7
v8	.5, .5, 0	16 17 18 19
v9	.5, .5, 1	20 21 22 23

1.3.4 Аналіз інших уявленнь

Потокові сітки зберігають грані впорядковано, але незалежно, щоб сітку можна було пересилати частинами. Порядок граней може бути просторовим, спектральним або базованим на інших властивостях сітки. Потокові сітки дозволяють рендерувати дуже великі сітки навіть тоді, коли вони завантажуються.

Прогресивні сітки передають дані про вершини і грані з рівнем деталізації, що підвищується. На відміну від потокових сіток прогресивні сітки дають загальну форму цілого об'єкта, але на низькому рівні деталізації.

Додаткові дані, нові ребра та грані, прогресивно збільшують деталізацію сітки.

Нормальні сітки передають поступові зміни сітки як безліч зсувів нормалей від базової сітки. За допомогою цієї техніки, ряд текстур відображає бажані зміни, що наростають. Нормальні сітки компактні, оскільки вираження усунення потрібно лише одне скалярне значення. Однак, техніка вимагає ряд складних трансформацій, щоб створити текстури зсуву.

1.4 Аналіз методів оптимізації для GPU

Від кількості полігонів у меші зазвичай залежить швидкість його рендерингу. Однак незважаючи на те, що кількість полігонів часто корелює з частотою кадрів за секунду (FPS), ви можете виявити, що навіть після зниження кількості полігонів меш, як і раніше, рендерується повільно. Скорочення кількості полігонів дозволяє більш ефективно заповнювати простір текстурних атласів та трохи знижує обчислювальне навантаження [6]

Для того, щоб досягти найкращої продуктивності, необхідно оптимізувати ігри на всіх етапах розробки – вибирати той варіант, який зберігатиме найбільшу привабливість з найменшими витратами ресурсів пристрою [7].

1.4.1 Аналіз методу представлення даних полігонів

У комп'ютерній графіці і 3д-художники, і програмісти під рендерингом розуміють створення плоскої картинки – цифрового растрового зображення із 3д сцени, процес отримання зображення за допомогою комп'ютерних програм [8].

У разі 3D-ігор першим етапом рендерингу є збір інформації про вершини 3D-об'єкта шейдером – програмою, яка виконується на відеокарті. Атрибути вершин збираються в пакет, який обов'язково включає позицію, а

також може містити інформацію про колір і нормалі. Дані обробляються відповідно до інструкцій вершинного шейдера і виводять нові начення позиції вершин та іншу інформацію. Він потрапляє в розтеризатор, який переносить положення вершин моделі координати екрана пристрою. Дані передаються у фрагментну або, як її ще називають – піксельну частину, де задаються такі значення як колір та прозорість. Це дуже ресурсомісткий процес, особливо сцен з великою кількістю моделей, тому 3D-художники максимально зменшують кількість полігонів і по можливості запікають деталізацію в карти нормалей. Однак цього часто буває недостатньо [9].

Разглянемо структуру даних, яка використовується для опису полігонів. Полігон складається з набору точок, званих вершинами, та посилань. Вершини часто зберігаються як масиви значень, наприклад, подібно до рисунку 1.10 [9].

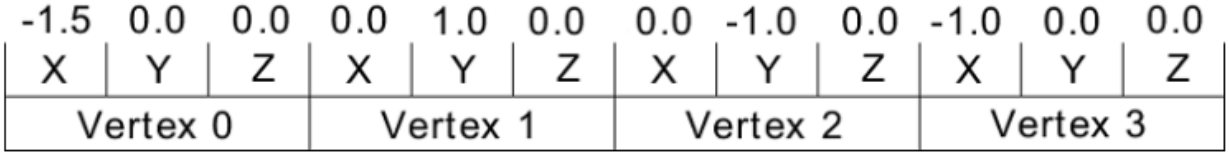


Рисунок 1.10 - Масив значень звичайного полігона [9]

У цьому випадку чотири вершини у трьох вимірах (x, y та z) дають нам 12 значень. Для створення полігонів другий масив значень описує самі вершини, як показано рисунке 1.11 [9].

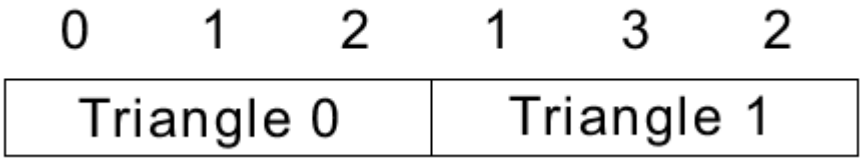


Рисунок - 1.11 - Масив посилань на вершини [9]

Ці вершини, з'єднані разом, утворюють два полігони. Зауважте, що два трикутники, у кожному з яких три кути, можна описати чотирма вершинами,

тому що вершини 1 і 2 використовуються в обох трикутниках. Щоб ці дані міг обробити GPU, передбачається, кожен полігон є трикутним. GPU очікують, що ви працюєте з трикутниками, тому що вони призначені саме для їхнього малювання.

Використані в попередньому прикладі дані вершин тривимірні, але це необов'язково. Вам може бути достатньо двох вимірювань, але часто необхідно зберігати й інші дані, наприклад, UV-координати для текстур і нормалі для освітлення.

1.4.2 Аналіз методу відображення полігону

При малюванні полігону GPU насамперед визначає, де потрібно малювати полігон. Для цього він обчислює позицію на екрані, де мають три вершини. Ця операція називається перетворенням (transform). Ці обчислення у GPU виконує невелика програма під назвою «вершинний шейдер».

Вершинний шейдер часто виконує інші типи операцій, наприклад, обробку анімацій. Після обчислення позицій всіх трьох вершин полігону, як на рисунке 1.12 [9].

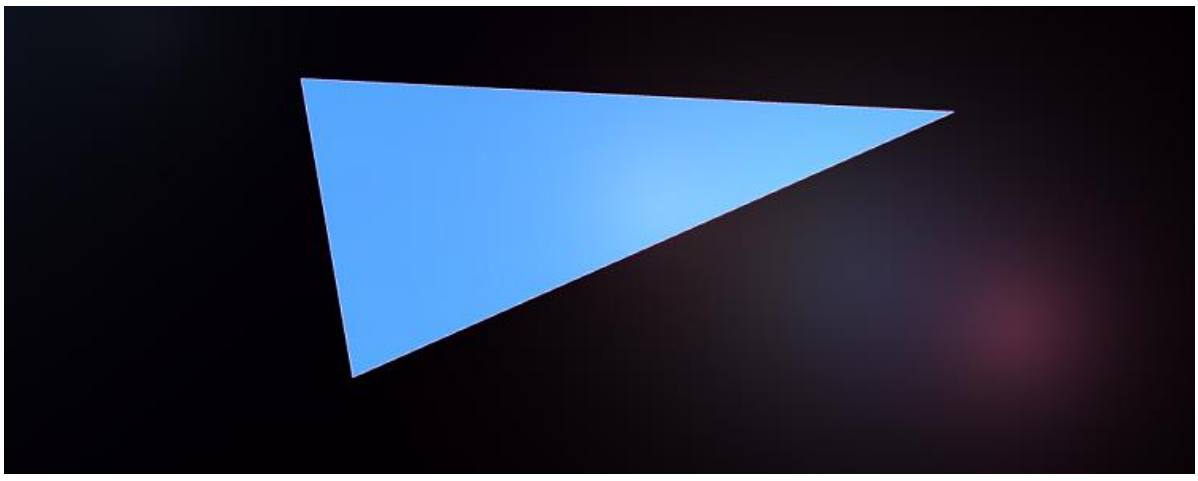


Рисунок 1.12 – Один полігон намальований на екрані [9]

На рисунку 1.13 [9] показаний порядок дій, що виконується GPU під час малювання полігону.

Рисунок 1.13 – Порядок дій GPU, який малює полігон [9]

GPU обчислює, які пікселі знаходяться у цьому трикутнику, а потім починає заповнювати ці пікселі за допомогою ще однієї маленької програми під назвою "фрагментний шейдер" (fragment shader). Фрагментний шейдер зазвичай виконується раз на піксель.

Якщо розділити трикутник на два і відобразити обидва трикутники, то порядок дій відповідатиме рисунку 1.14 [9].

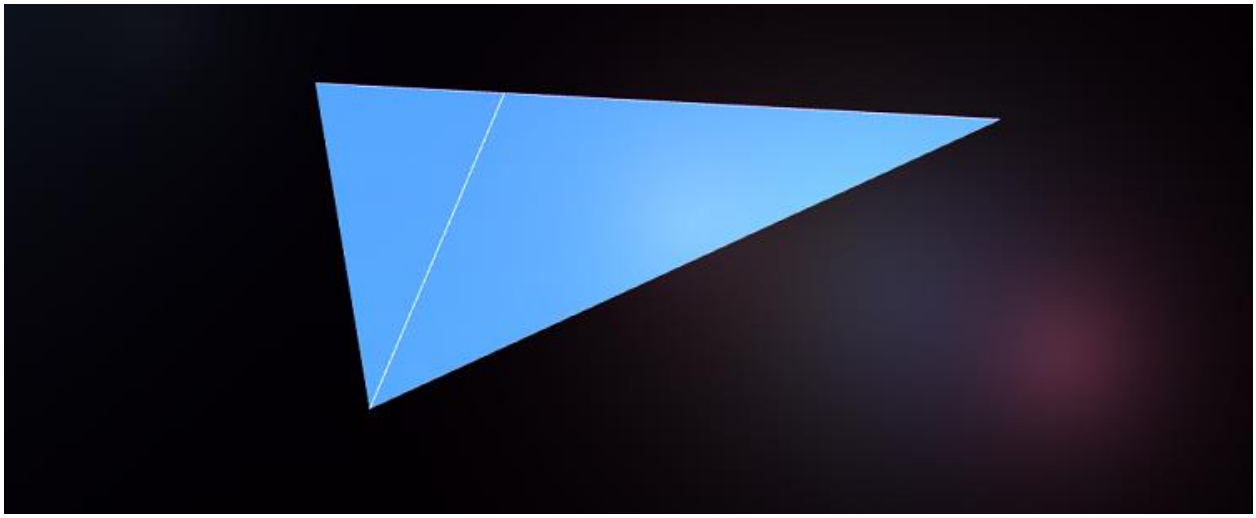


Рисунок 1.14 – Розділення полігона на два [9]

У цьому випадку потрібно вдвічі більше перетворень і підготовок, але оскільки кількість пікселів залишилася такою самою, операції не потрібно розтеризувати додаткові пікселі, як показано на рисунке 1.15 [9].

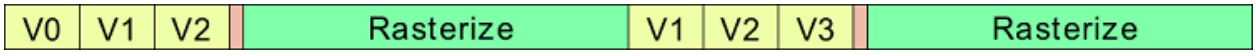


Рисунок 1.15 – Порядок дій GPU, який малює два полігони [9]

Це показує, що подвоювання кількості полігонів необов'язково подвоює час рендерингу.

1.4.3 Аналіз методу параметри вершин

Щоб вершину можна було використовувати повторно, при кожному використанні вона має бути незмінною [10]. Зрозуміло, тієї ж має залишатися позиція, але й інші параметри теж не повинні змінюватися. Параметри, що передаються вершині, залежать від використовуваного двигуна. Ось два широко поширені параметри:

- текстурні координати;
- нормалі.

При UV-накладенні на 3D-об'єкт будь-який шов, що створюється, означатиме, що вершини вздовж шва не можуть бути спільними (рисунок 1.16[10]). Тому в загальному випадку варто уникати швів.

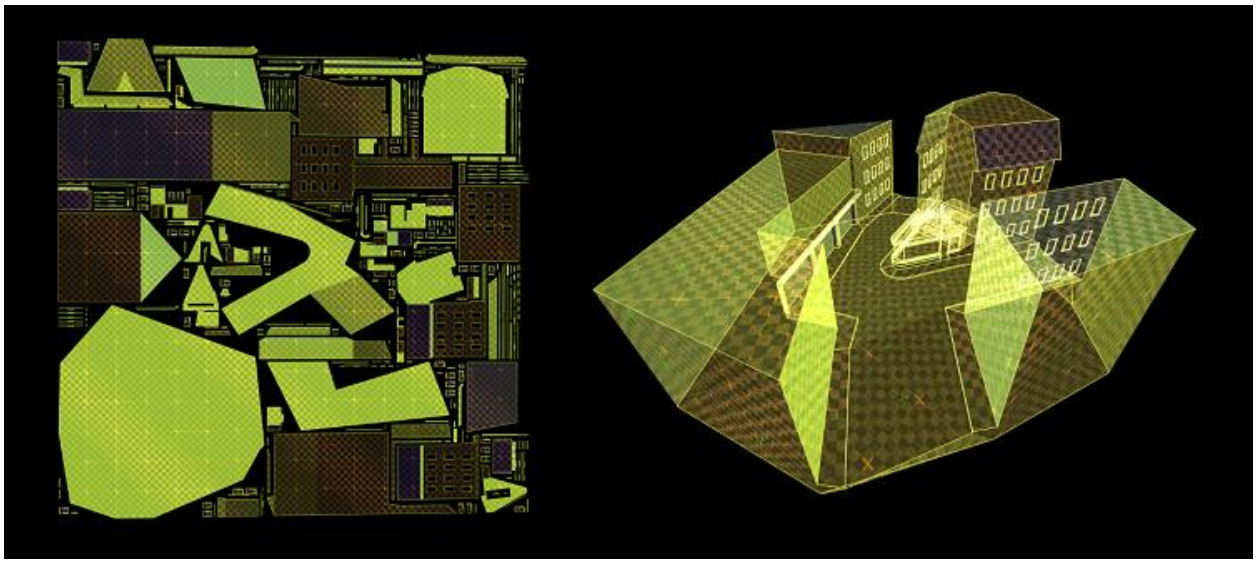


Рисунок 1.16 – UV-накладання швів текстури [10]

Для правильного освітлення поверхні кожна вершина зазвичай зберігає нормаль вектор, спрямований від поверхні. Завдяки тому, що всі полігони із загальною вершиною задаються однією нормаллю, їхня форма здається плавною. Це називається плавним затіненням (smooth shading). Якщо кожен трикутник має власні нормалі, то ребра між полігонами стають вираженими, а поверхня здається плоскою. Тому це і називається плоским затіненням (flat

shaded). На рисунку 1.17 [10] показано два однакові меші, один із згладженим затіненням, а другий — із плоским.

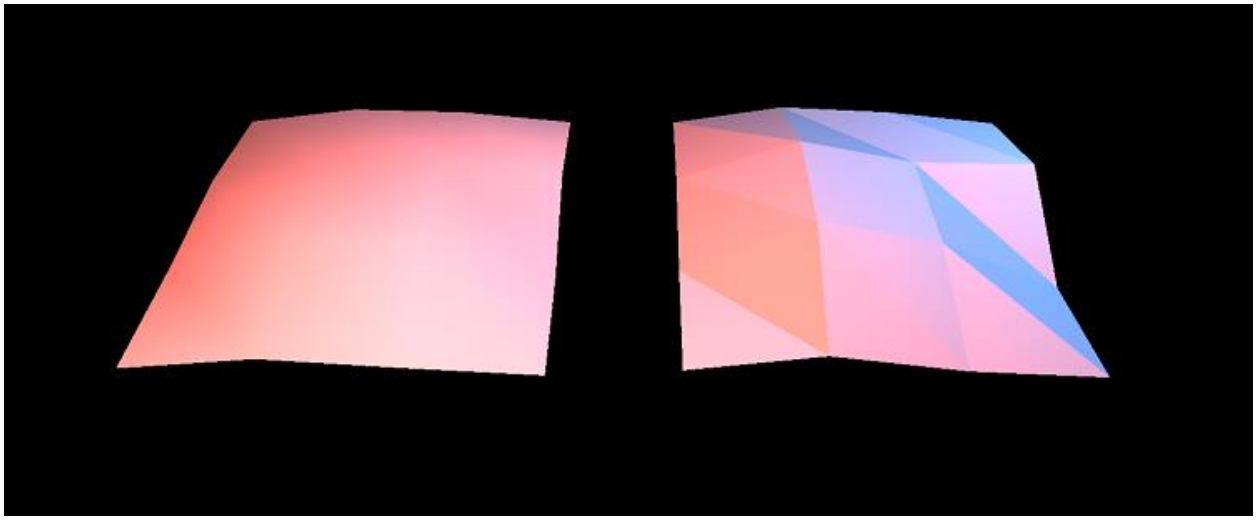


Рисунок 1.17 – Порівняння згладженого з плоским затіненням [10]

Ця геометрія зі згладженим затіненням складається з 18 трикутників і має 16 загальних вершин. Для плоского затінення 18 трикутників потрібно 54 (18 x 3) вершини, тому що жодна з вершин не є загальною. Навіть якщо два меша мають однакову кількість полігонів, швидкість їхнього відмальовування все одно буде різною.

1.4.4 Аналіз методу важливість форми

GPU швидко працюють переважно тому, що вони можуть виконувати безліч операцій паралельно. Коли GPU малює полігон, він віддає безлічі конвеєрів завдання заповнювати квадрати пікселів. Зазвичай це квадрат розміром вісім на вісім пікселів. GPU продовжує це робити, доки не будуть заповнені всі пікселі. Після обчислення всіх вершин у квадраті обладнання відкидає пікселі поза трикутником.

На рис. 1.18 [10] показано трикутник, для відтворення якого потрібно три квадрати (тайли). Більшість обчислених пікселів (блакитні)

використовують, а показані червоним виходять за межі трикутника і будуть відкинуті.

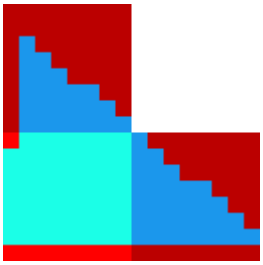


Рисунок 1.18 – Три тайли для малювання трикутника [10]

Полігон на рисунку 1.19 [10] з такою самою кількістю пікселів, але розтягнутий, вимагає для заповнення більшої кількості тайлів; Більшість результатів роботи у кожному тайлі (червона область) буде відкинуто.

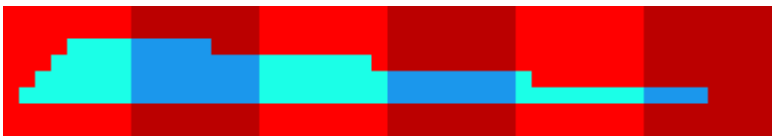


Рисунок 1.19 – Заповнення тайлів у розтягнутому зображенні [10]

Кількість пікселів, що відмальовуються, - це тільки один з факторів. Так само важливою є форма полігону. Для підвищення ефективності намагайтеся уникати довгих, вузьких полігонів і віддавайте перевагу трикутникам з приблизно рівною довжиною сторін, кути якого близькі до 60 градусів. Дві плоскі поверхні на рис. 1.20 [10] тріангульовані двома різними способами, але при рендеринг виглядають однаково.

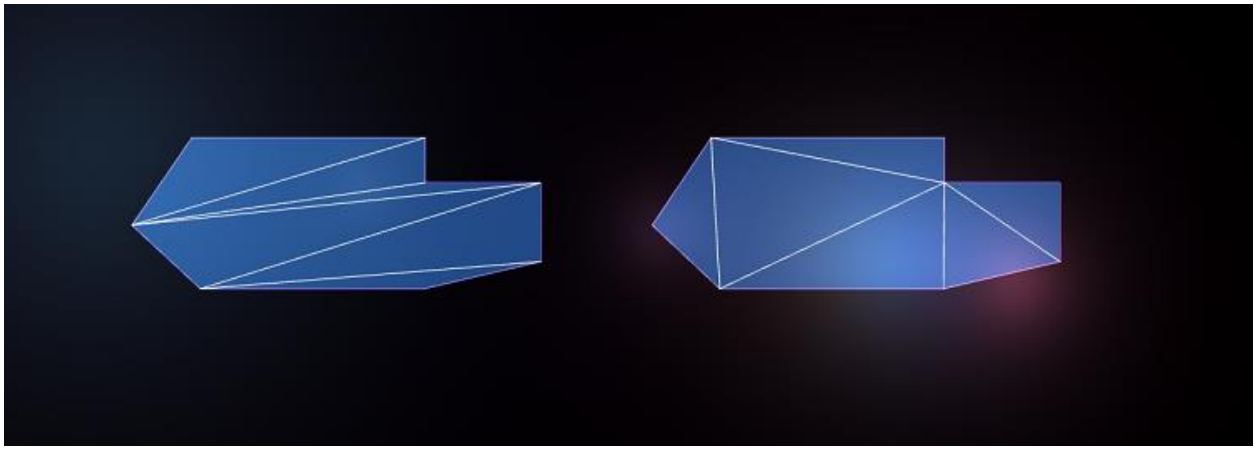


Рисунок 1.20 – Поверхні, триангульовані двома різними способами [10]

Вони мають абсолютно однакову кількість полігонів та пікселів, але так як поверхня лівого має більш довгі, вузькі полігони, ніж у правого, його рендеринг буде повільнішим.

1.4.5 Аналіз методу перемальовка

Для малювання шестипроменевої зірки можна створити меш із 10 полігонів або намалювати ту ж фігуру всього з двох полігонів, як показано на рисунку 1.21 [10].

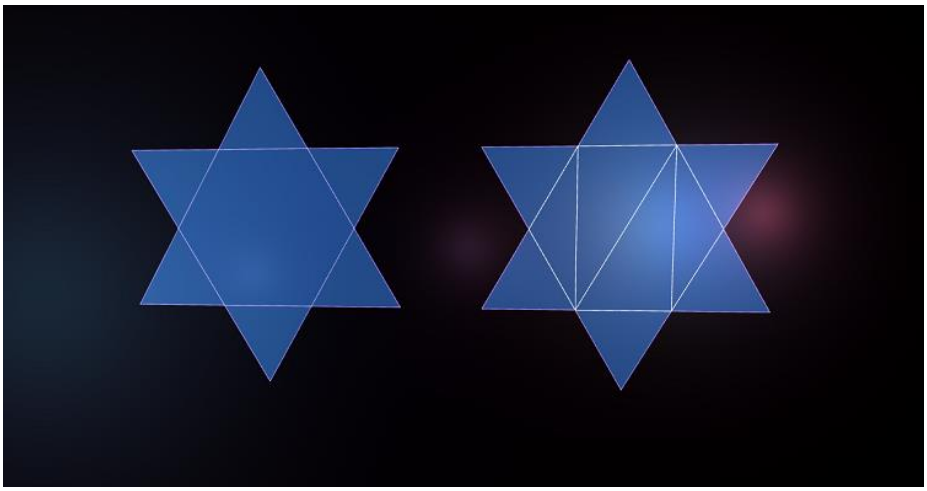


Рисунок 1.21 – Два різні способи відтворення шестипроменевої зірки [10]

Можна вирішити, що швидше відмалювати два полігони, ніж 10. Однак у даному випадку це швидше за все неправильно, тому що пікселі в центрі зірки малюватимуться двічі. Це називається перемальовкою (overdraw). Насправді воно означає, що пікселі перемальовуються більше одного разу. Перемалювання природним чином виникає у всьому процесі рендерингу. Наприклад, якщо персонаж частково прихований колоною, він буде відмальований цілком, незважаючи на те, що колона перекриває частину персонажа. Деякі двигуни використовують складні алгоритми, що дозволяють уникати малювання об'єктів, невидимих на кінцевому

зображенні, але це важке завдання. Центральному процесору часто важче з'ясувати, що потрібно малювати, ніж GPU відмалювати це.

1.5 Реалізація скриньки на підлозі

На рис. 1.22 [10] показана проста сцена: ящик, що стоїть на підлозі. Підлога складається з двох трикутників, а ящик складається з 10 трикутників. Перемальовка у цій сцені показана червоним кольором.

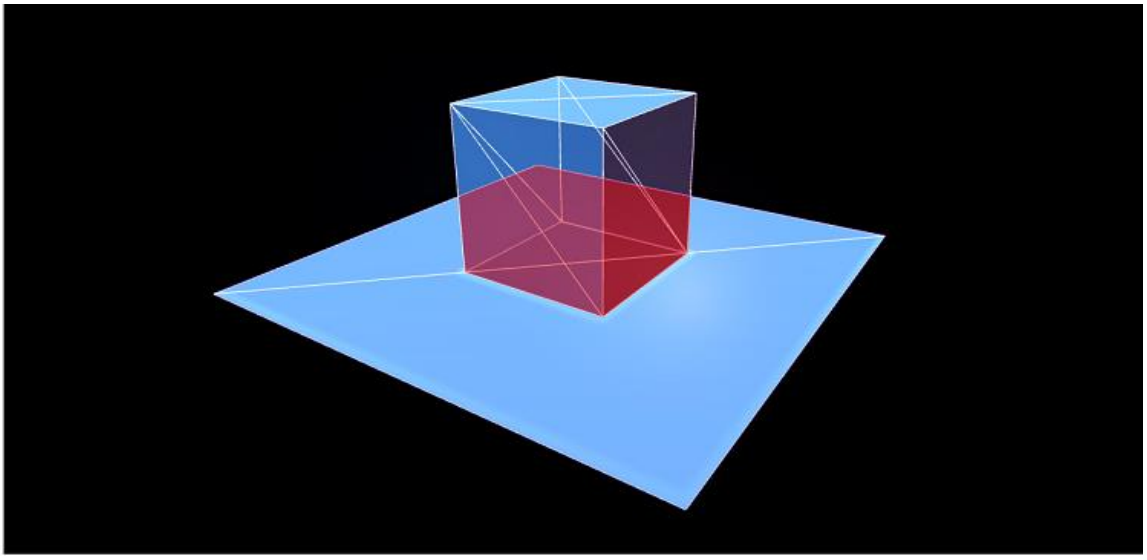


Рисунок 1.22 – Скринька, що стоїть на підлозі [10]

У цьому випадку GPU малює частину підлоги під ящиком, незважаючи на те, що її не буде видно. Якби натомість ми створили під ящиком дірку в підлозі, то отримали б більшу кількість полігонів, але набагато менше перемальовки, як видно на рисунку 1.23 [10].

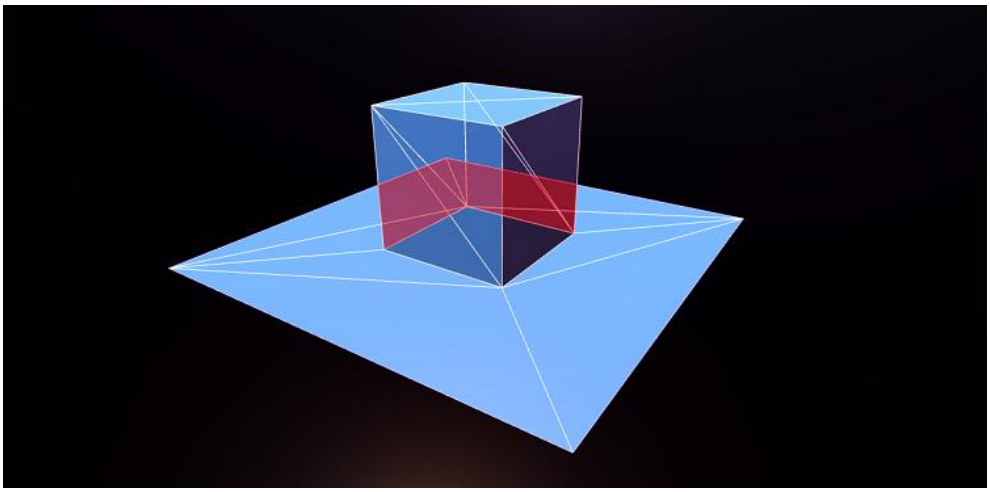


Рисунок 1.23 – Діра під ящиком, що дозволяє уникнути перемальовки [10]

У таких випадках все залежить від вашого вибору. Іноді варто зменшити кількість полігонів, отримавши натомість перемальовку. В інших ситуаціях варто додати полігонів, щоб уникнути перемальовування. Ще один приклад: дві наведені нижче фігури є однаково виглядають мішами поверхні з вістрями, що стирчать з неї. У першому меші (рисунок 1.24 [10]) вістря розташовані на поверхні.

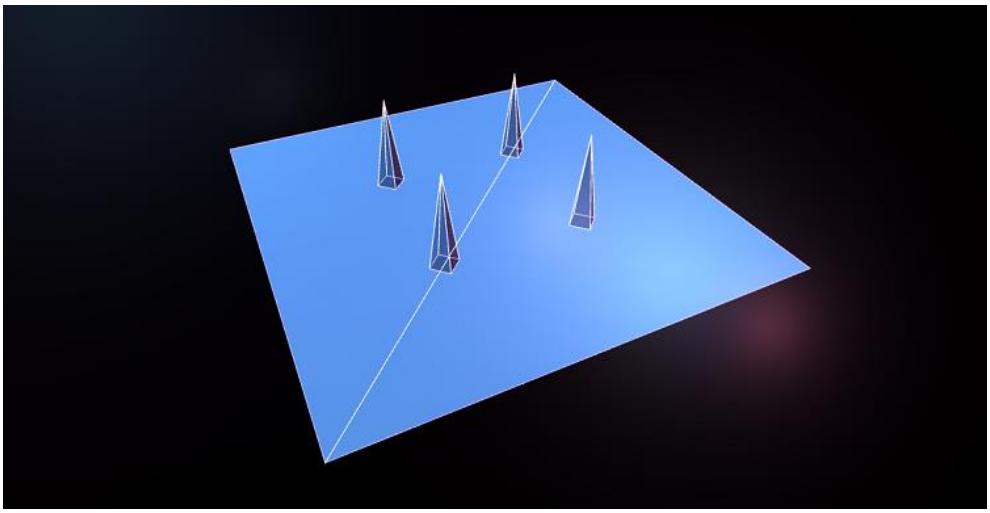


Рисунок 1.24 – Вістря розташовані на поверхні [10]

У другому меші на рис. 1.25 [10] поверхнею під вістрями прорізані отвори, щоб зменшити обсяг перемальовки.

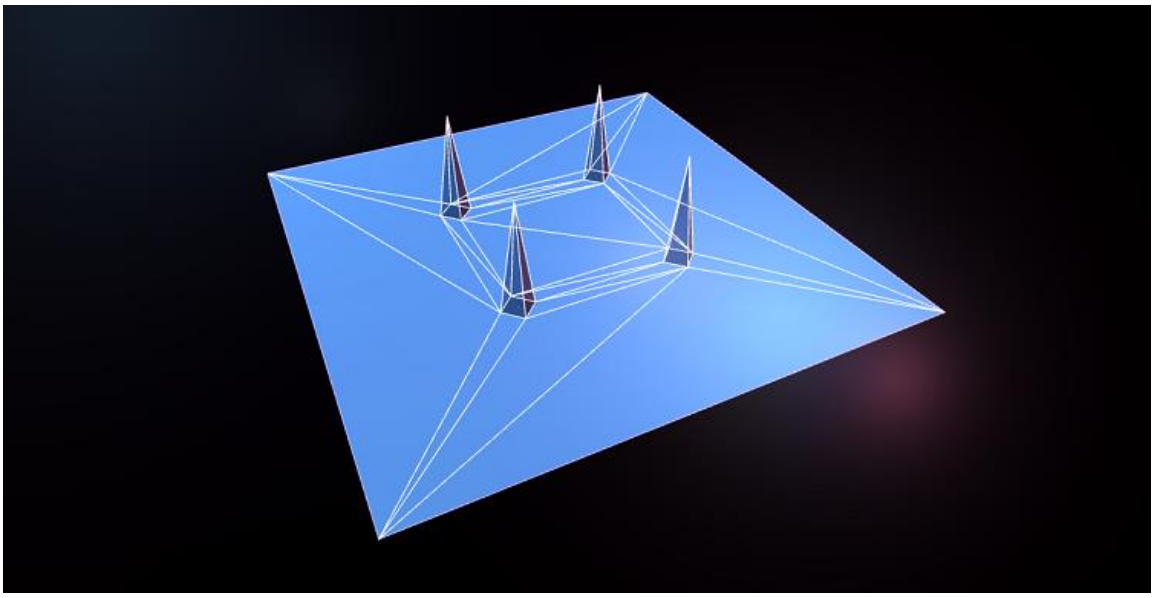


Рисунок 1.25 – Під вістрями вирізані отвори [10]

У цьому випадку для вирізування отворів було додано безліч полігонів, частина з яких має вузьку форму. До того ж поверхня перемальовки, якої ми позбулися, не дуже велика, тому в даному випадку ця техніка неефективна.

1.5.1 Використання Z-буфер та Z-конфлікт

Коли GPU малює два полігони, що накладаються один на одного, проаналізуємо як визначити, який з них знаходиться поверх іншого? Ед Кетмел у своїй статті виклав десять різних підходів [10]. В одній частині статті він зауважує, що вирішення цього завдання буде тривіальним, якщо комп'ютери мають достатньо пам'яті для зберігання одного значення глибини на піксель. У 1970-х та 1980-х це був дуже великий обсяг пам'яті. Однак сьогодні так працює більшість GPU: така система називається Z-буфером.

Z-буфер (також відомий як буфер глибин на рисунке 1.26 [10]) працює так: з кожним пікселем зв'язується значення його глибини. Коли обладнання малює об'єкт, воно обчислює, наскільки далеко від камери малюється піксель. Потім воно перевіряє значення глибини вже існуючого пікселя. Якщо він далі від камери, ніж новий піксель, новий піксель відмальовується.

Якщо вже наявний піксель ближче до камери, ніж новий, новий піксель не відмальовується. Такий підхід вирішує безліч проблем та працює, навіть якщо полігони перетинаються.

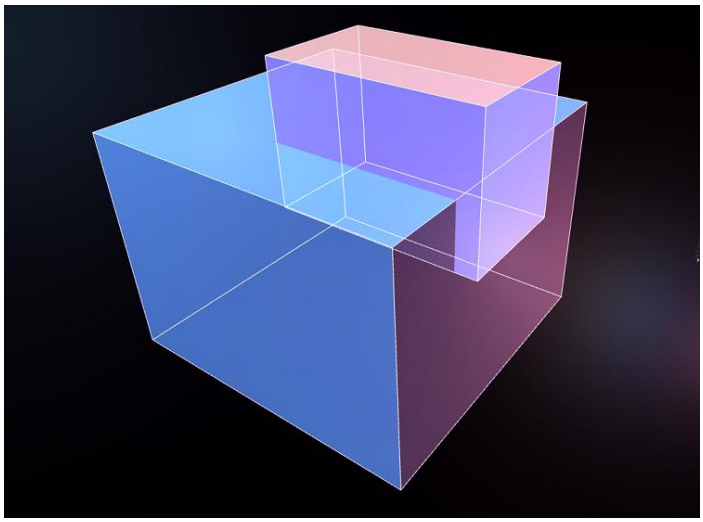


Рисунок 1.26 – Полігони, що перетинаються, оброблені буфером глибин [10]

Однак Z-буфер не має нескінченної точності. Якщо дві поверхні знаходяться майже на одній відстані від камери, то це збиває GPU з пантелику і він може випадково вибрати одну з поверхонь, як це показано на рисунку 1.27 [10].

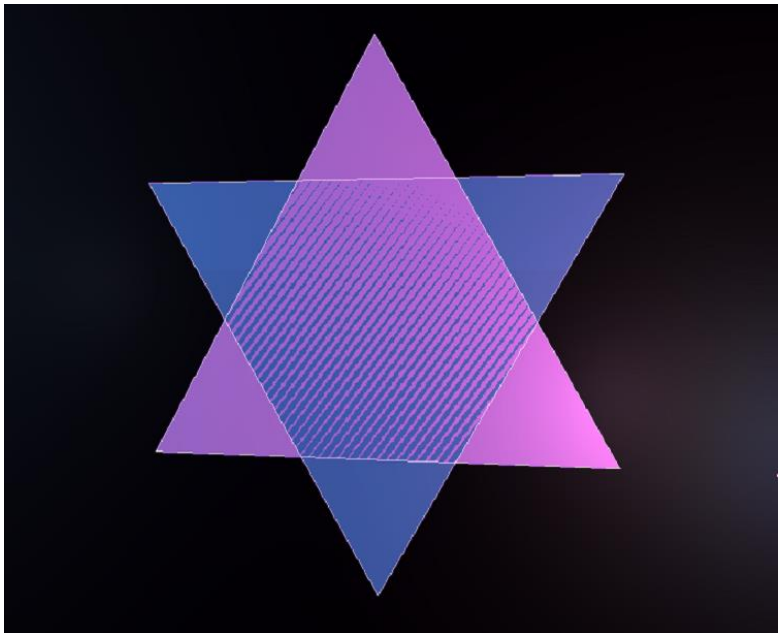


Рисунок 1.27 – У поверхонь на однаковій глибині виникають проблеми з відображенням [10]

Це називається Z-Конфліктом (Z-fighting) і виглядає дуже забаговано. Часто Z-конфлікти стають гіршими, ніж далі поверхня від камери. Розробники двигунів можуть вбудовувати в них виправлення, що дозволяють згладити цю проблему, але якщо художник створює досить близькі полігони, що накладаються один на одного, то проблема все одно може виникати. Ще одним прикладом може бути стіна з постером, що висить на ній. Постер знаходиться майже на тій самій глибині від камери, що й стіна за ним, тому дуже високий ризик Z-конфліктів. Рішення полягає в тому, щоб вирізати в стіні отвір під плакатом. При цьому також зменшиться обсяг перемальовки.

У крайніх випадках Z-конфлікт може виникнути навіть коли об'єкти стосуються один одного. На рисунку 1.28 [10] показаний ящик на підлозі, і оскільки ми не вирізали в підлозі під ящиком отвір, z-буфер може бути спантеличений поруч з ребром, де підлога зустрічається з ящиком.

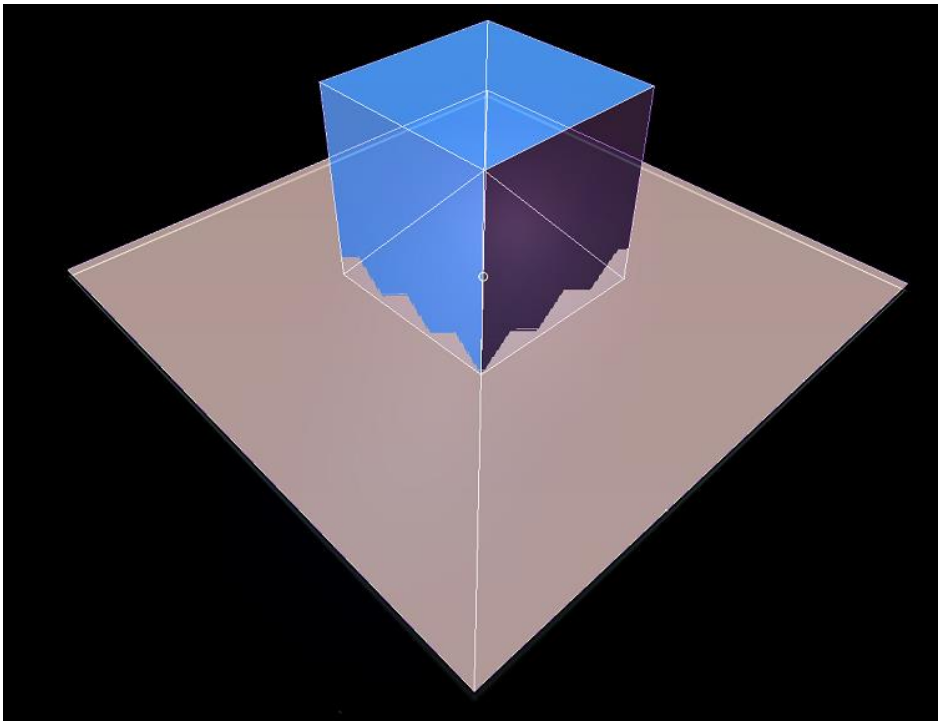


Рисунок 1.28 – Приклад Z-конфлікту полігонів, що накладаються один на одного [10]

1.5.2 Використання дзвінків малювання

GPU стали надзвичайно швидкими – настільки швидкими, що ЦП можуть за ними і не встигати [11]. Так як GPU по суті призначені для виконання одного завдання, їх набагато простіше змусити працювати швидко. Проте GPU малює лише те, що йому наказує малювати ЦП. Якщо ЦП не може досить швидко «годувати» GPU даними, відеокарта простоюватиме. Щоразу, коли ЦП наказує GPU щось відмалювати, називається викликом відмальовування. Найпростіший виклик відображення складається з одного меша, в тому числі одного шейдера і одного набору текстур.

Те, з чого складається виклик малювання, і витрати на нього сильно залежать від конкретних двигунів та архітектур. Деякі двигуни можуть об'єднати в один виклик малювання безліч мешів (виконати їх батчинг, batch), але всі меші при цьому повинні будуть мати однаковий шейдер, або можуть мати інші обмеження. Нові API на зразок Vulkan і DirectX 12 розроблені спеціально для вирішення цієї проблеми за допомогою оптимізації того, як програма спілкується з графічним драйвером, збільшуючи таким чином кількість викликів, які можна передати за один кадр. Питання по обробці та передачі зображень які використовуються в цій роботі розглянуті в публікаціях по науковим напрямкам кафедри МІРЕС [12-28].

В цьому розділі були розглянуті основні принципи створення 3D моделей, а саме: для створення тривімірного об’єкту необхідно створити модель цього об’єкта у просторових координатах X, Y, Z, тобто описати кожну точку на поверхні цього – вказати його координати; складові моделювання сітки такі, як вершина, ребро, грань та полігон. Також було проаналізовано представлення полігональних сіток: вершинне уявленя сіток, де об’єкт описується як безліч вершин, уявлення списку граней, де об’єкт представляється як безліч граней та безліч вершин, «крилате» уявлення, явно

представляться верниши, грані та ребра сітки. Ще були проаналізовані методи для оптимізації сіток 3D моделей, щоб GPU швидшке та менш трудомістко рендерив полігони, і було виявлено, що на швидкість рендеру впливає не тількі кількість полігонів, а ще й форма цих полігонів, розположення різних об’єктів відносно один до одного та потрапляння певних полігонів у камеру рендера.

2 АНАЛІТИЧНИЙ ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ВИДІВ 3D-МОДЕЛЮВАННЯ

Тривимірна комп'ютерна графіка з'явилася порівняно недавно, проте задовго до появи персональних комп'ютерів тривала кількість часу робилися спроби відтворення тривимірного світу на фотографіях, картинах, кіноплівці.

Процес створення 3D-моделі може здійснюватися багатьма способами. Все залежить від цілей, термінів, складності виконання та інших особливостей виробництва. Зазвичай основними стадіями підготовки тривимірної графіки є: моделювання, текстурування, анімація та сам рендер.

Тривимірне моделювання - це процес створення тривимірної моделі об'єкта. Його основне завдання у тому, щоб показати візуальний обсяг, створюваного об'єкта [29]. Основними критеріями оптимізації 3D-моделі для анімації є: топологія та кількість полігонів. Полігон або полігональна сітка - це набір вершин, граней і ребер, які визначають форму багатогранного об'єкта в тривимірній комп'ютерній графіці. Говорячи простими словами, коли ми дивимося на 3D-модель, то ми бачимо ту саму полігональну сітку, адже складові полігонів і утворюють ті форми, які ми створюємо. У більшості випадків при створенні самої моделі розробники використовують режим сітки (wire-frame), щоб правильно побудувати форму об'єкта та його топологію [30].

Розглянемо види полігонів:

1. Полігон із трьома вершинами. Він має мінімально кількість вершин та сторін. Також називається як "трикутник" або "трис". При необхідності його з легкістю можна перетворити на полігон із трьома вершинами [31].
2. Полігон із чотирма вершинами. Має чотири вершини та чотири сторони, що робить його вкрай зручним для побудови тривимірних форм, а також при маніпуляціях із полігональною сіткою. Цей вид є обов'язковим при

побудові 3D-моделей, які надалі удосконалюватимуться, анімуватимуться і згладжуватимуться.

3. Полігон із п'ятьма вершинами або більше (N-Gon). Створює труднощі у вигляді різних артефактів під час текстурування, рендеру або анімації. Також із мінусів — погано піддається згладжуванню на згинальних поверхнях.

Під топологією розуміється плавна і потокова організованість полігонів. Просто кажучи, топологія — акуратність, правильність полігональної сітки і безперервність каркасу.

Топологія в 3D-моделюванні є ключовим аспектом правильної розробки тривимірного об'єкта, тому що від цього залежить, як надалі згладжувати стики полігонів, робити розгортка і анімувати об'єкт [33].

2.1 Аналіз методу полігонального моделювання

Приклад полігональної моделі на рис. 2.1 [34].

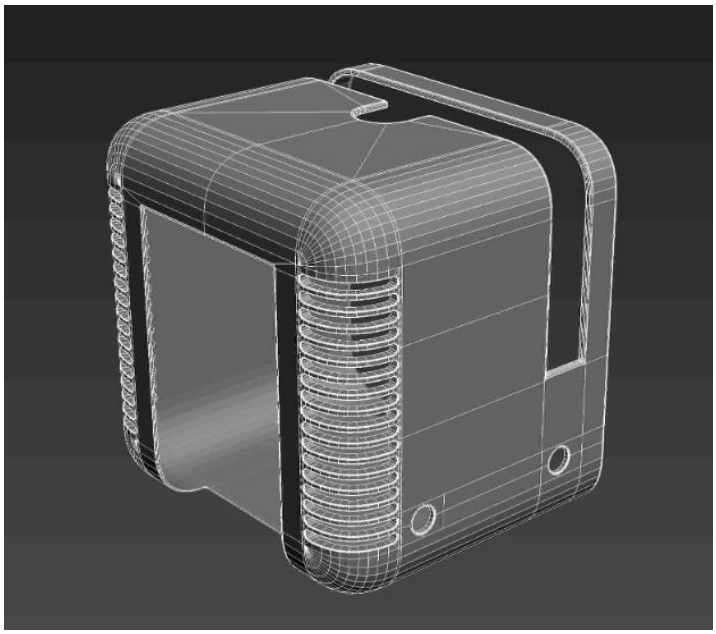


Рисунок 2.1 – Полігональний 3D об’єкт [34]

Дозволяє швидко зробити 3D-модель за дизайнерським концептом, ескізом, начерком. З полігональних 3D моделей створюється реалістичне

оточення для візуалізації. Цим способом можна створити складки тканини, наприклад. Твердотільне моделювання, за всієї інженерної точності, для таких завдань не підходить [34].

2.2 Аналіз методу вердотільне моделювання

Приклад твердотільної моделі на рисунке 2.2 [34]

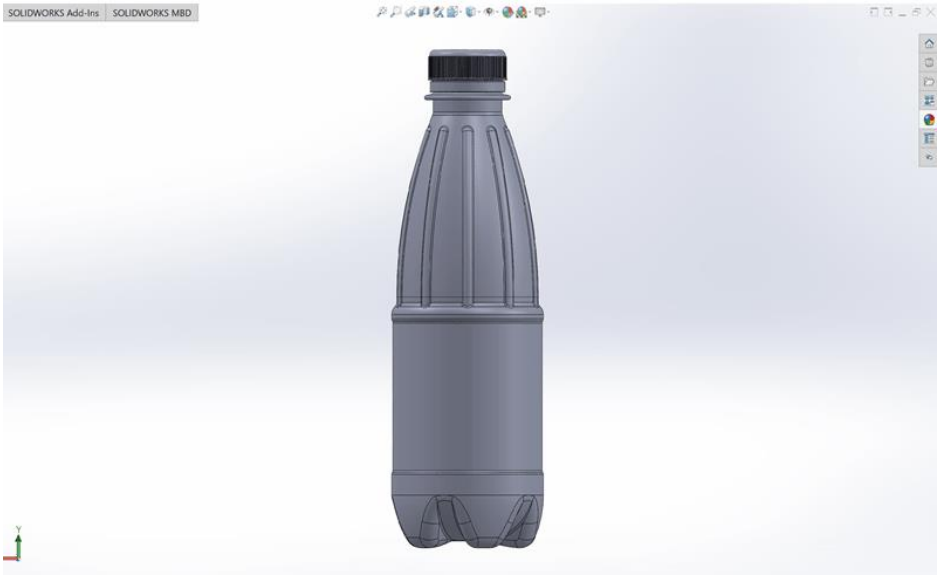


Рисунок 2.2 – Твердотільний 3D об’єкт [34]

Проводиться на усіх стадіях розробки пристрою від перших ескізів. Це і конструкція, і креслення, перший прототип, і файл для верстата ЧПК. Якщо в полігональній моделі якісь моменти можна робити «на око», твердотільна 3D-модель будується за формулами та математичними розрахунками. У Solidworks, технічні зазори та рухомі частини моделюються як елементи працюючої конструкції, а не лише як зовнішній образ.

2.3 Аналіз побудови моделі за допомогою полігонів

Приклад побудованої полгінонами моделі на рисунке 2.3 [34]

Цей спосіб створення складної моделі із геометричних примітивів. Спочатку створюється примітив - куб, наприклад, а потім його вершини, кути, грані та ребра розмножуються за допомогою полігональної сітки. Чим щільніше полігони у сітці, тим складнішу форму має модель.

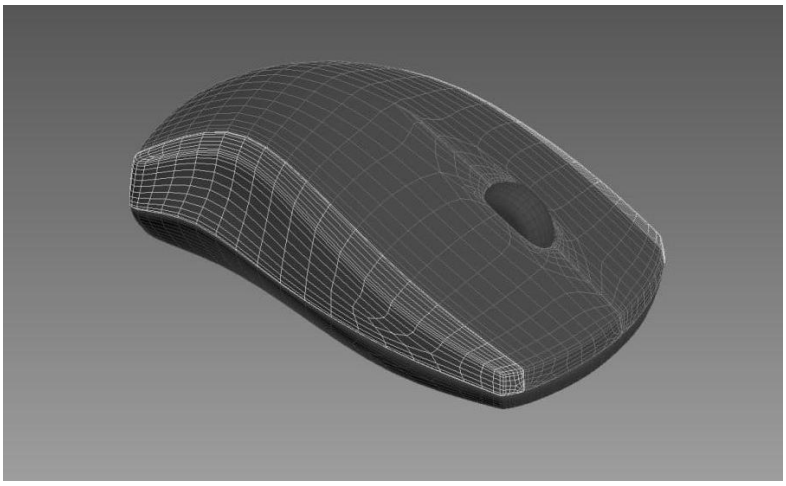


Рисунок 2.3 - Побудований полігонами 3D об’єктй [34]

При полігональному моделюванні можуть використовуватись креслення, але створити 3D-модель полігонами можна і за описом, і просто взяти образ з уяви.

2.4 Аналіз методу Hard Surface

Приклад сітки для hard surface зображен на рисунке 2.4 [34]

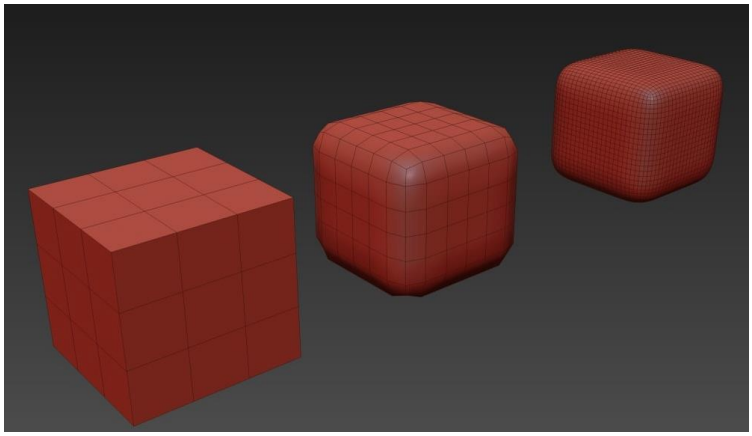


Рисунок 2.4 - сітка для Hard Surface об’єкта[34]

Спосіб полігонального моделювання, що підходить для рукотворних об'єктів. Його плюс у тому, що достатньо створити низькополігональну модель, яка автоматично згладжується модифікаторами 3D-редактора. Через це у hard surface моделей специфічна сітка – великі полігони на площинах та висока щільність сітки на гранях для надання жорсткості.

2.5 Аналіз методу сплайн

Приклад сплайн об’єкта на рисунке 2.5 [34].

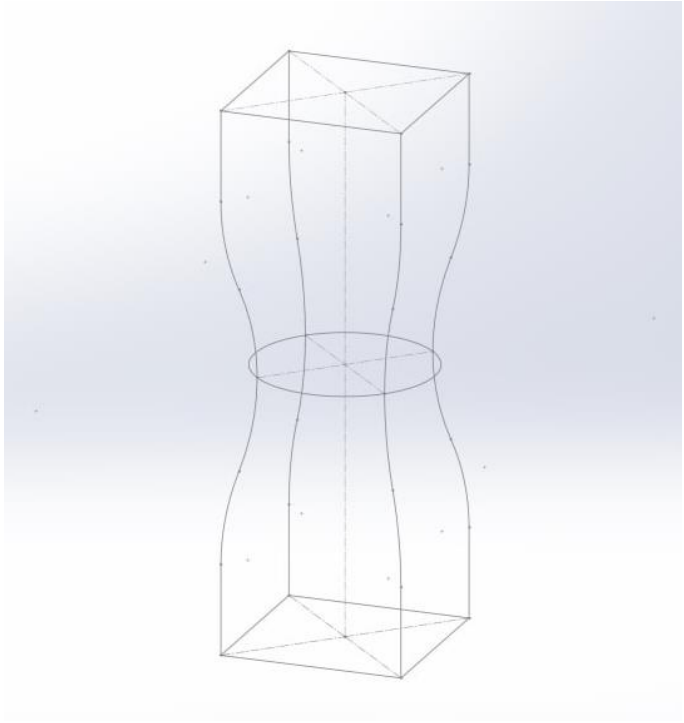


Рисунок 2.5 - Сплайн 3D об’єкт[34]

Створення 3D моделі за допомогою тривимірних кривих, які описують форму об'єкта, як каркас. Модель задає лише лінії, а поверхні між ними з'являються автоматично. Цей спосіб дозволяє створювати моделі плавними відразу, без поступового збільшення числа полігонів у сітці. Крім полігонального, сплайн застосовується і у твердотільному моделюванні. З ним можна створювати форму моделі не лише попередньо в ескізах, а й під час роботи із самою моделлю.

2.6 Аналіз методу Nurbs

На рисунке 2.6 [34] зображено nurbs моделювання.

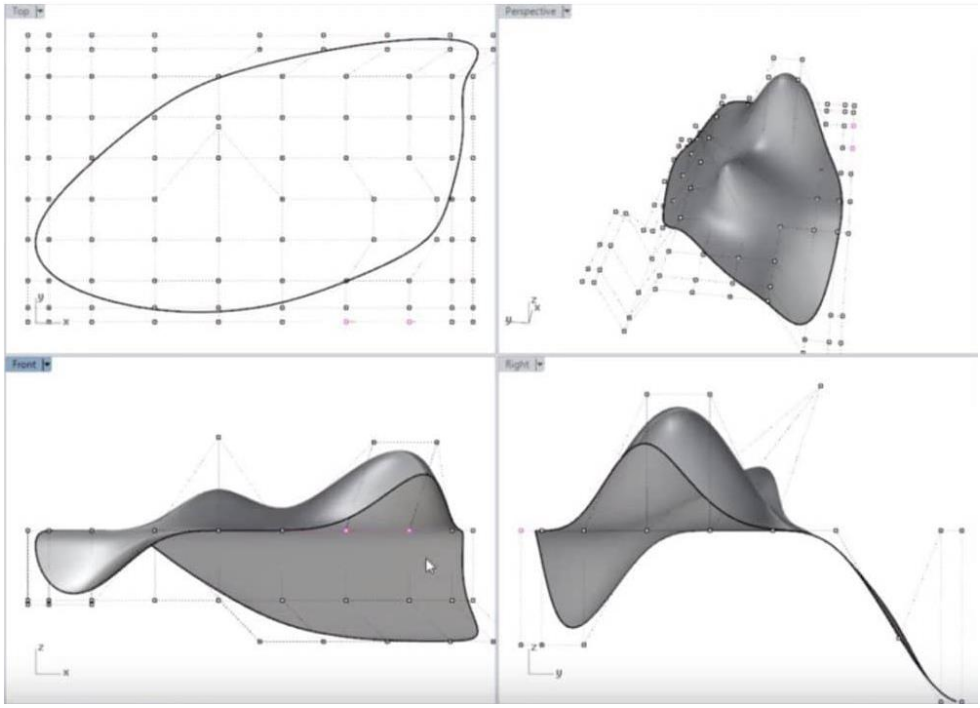


Рисунок 2.6 - Nurbs 3D об’єкт [34]

Моделювання за математичними формулами, цей спосіб з'явився задовго до ПК та 3Д-графіки. Він дозволяє створювати крапки та криві у тривимірній площині. Спосіб вимагає профільної освіти та застосовується в автомобільній промисловості, літако- та суднобудуванні.

2.7 Аналіз методу поверхневого моделювання

На рисунке 2.7 [34] предоставлено приклад поверхневого моделюваняя.

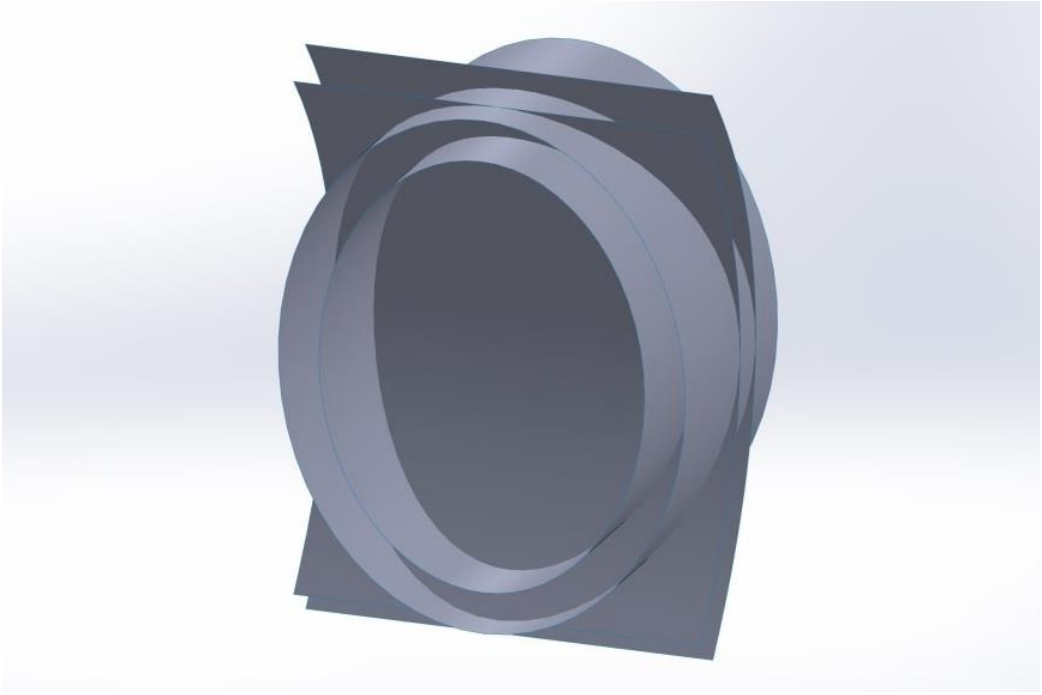


Рисунок 2.7 - 3D об’єкт, створений поверхневим моделюванням [34]

Створення окремих примітивів, які перетинаються між собою та утворюють модель. Після перетину в потрібній точці предмет обрізається шляхом заокруглення або застосування інших інструментів.

2.8 Аналіз методу скульптингу

Скульптурне моделювання або 3D-скульптинг – особливий вид моделювання об'єктів, який зображен на рисунке 2.8. Він значно відрізняється від звичайного полігонального моделінгу.

Рисунок 2.8 - Скульптинг 3D моделі

Основний метод створення 3D-скульптур – це деформація частин об'єкта. Вона створюється за допомогою різних інструментів (так званих кистей скульптингу), які дозволяють змінювати увігнутість або опуклість моделі, видаляти або додавати матеріал, змінювати кути та грані поверхні, а також робити безліч додаткових перетворень. 3D-скульптинг не є заміною тривимірного моделювання. До нього приходять лише для створення органічних форм або будь-яких інших форм, які набагато легше створити з «віртуальної глини», аніж рухати полігони за крапки. На цьому етапі можна поринути в творчість і ліпити, не замислюючись про полігони, яких буде досить багато, тому необхідно додатково обробити модель за допомогою інструментів, що входять до програм для моделювання. Цей процес називається ретопологія, тобто перетворення правильної полігональної сітки.

В цьому розділі було проведено огляд видів полігоновів та полігональних сіток. Полігони існують із трьома, чотирма і п’ятьма вершинами. Правильна полігональна сітка повинна бути плавною та мати потокову організованість полігонів. Також проаналізовано методи полігнонального моделювання такі, як: твердотільне моделювання, полігональне моделювання, hard surface, сплайн моделювання, nurbs моделювання, поверхневе моделювання та скульптинг. Для кваліфікаційної роботи було обрані методи скульптинга та полігонального моделювання. Завдяки скульптингу в програмі ZBrush буде отримано модель персонажа, а полігональним моделюванням в MAYA буде зроблена ретопологія.

3 АНАЛІЗ ЕТАПІВ СТВОРЕННЯ 3D-МОДЕЛІ

3.1 Аналіз етапу «ААА-пайплайн»

Скорочення ААА це:

- A lot of time (багато часу);
- A lot of resources (багато ресурсів);
- A lot of money (багато грошей).

ААА-гра – це гра з великими вкладеннями її розробку і маркетинг. Наприклад, бюджет Grand Theft Auto V оцінюється в 265 000 000 $.

Triple-A проект – це гра високих очікувань. На розробку топової гри у творців вистачило грошей і тепер гравці очікують від неї якості. Звідси випливають високі вимоги до кожної 3D моделі всередині гри.

Пайплайн – це конвеєр розробки всіх 3D моделей.

ААА-Пайплайн - це великий технологічний процес створення та оптимізації моделі, щоб помістити її в гру. Процес починається з блокінгу та закінчується готовою моделлю всередині проекту [35].

Працюючи по пайплайну, ви не просто рухаєш точки, а вирішуєш цілу низку технічних та художніх завдань:

- в якому стилі повинна бути модель;
- скільки полігонів у ній буде;
- чи необхідна карта нерівностей (normal map);
- використовуємо сучасні фізично коректні матеріали (PBR) або робимо плоский колір з картою відблиску (а може і без карти відблиску зовсім);
- яка роздільна здатність текстур на квадратний метр (тексель) і якого дозволу самі текстури;
- з якого ракурсу її бачитимуть найчастіше, модель на передньому чи задньому плані, і чи вона анімуватиметься;

– як і де пекти карти (не забути зробити скоси для запічки!);
– скульптувати чи створювати фактуру у фотошопі/пейнтері;
– правильно розбити усі полігони на трикутники;
– чи потрібна карта прозорості, чи варто розгортати кілька моделей в один атлас і як пакувати текстури;
– як експортувати в двигун;
– зробити спрощені моделі, які завантажуються на відстані (лоди), створити геометрію для прорахунку фізики (коліжн).

пайплайн складається з 5 етапів:

– драфт (форми та силует);
– сітка (хайполі, лоуполі);
– розгортка;
– запікання;
– текстури;

Все, що потрібно знати про етапи ААА-пайплайну:

1) послідовність

Це означає, що доти, доки повністю не закінчен, наприклад, етап блокінгу, починати роботу над low poly, high poly, розгорткою та іншими неможна (рисунок 3.1 [35]).

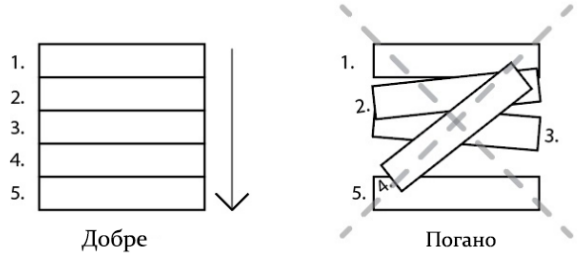


Рисунок 3.1 - Приклад правильної послідовності [35]

2) Варіації пайплайна

Деякі етапи пайплайну можна робити. Пайплайн для гармат, техніки та предметів трохи відрізняється від пайплайну для персонажів чи будівель.

Змінюються програми та технічні вимоги, проте порядок етапів залишається тим самим.

3) Недопрацювання одного етапу найчастіше веде до провалу інших етапів

Робота може посипатися як картковий будиночок, якщо було зроблено неякісно один із етапів пайплайну. Наприклад поганий драфт не врятувати навіть найкрутішими текстурами, крива хайполі загубить відблиск навіть ідеальної лоуполі, а розгортка зроблена без урахування вимог запічки наробить купу потворних швів.

4) Пайплайн універсальний

Можна моделити для ААА або інді-гри. Можна моделити в Maya або Blender – етапи залишаться колишніми та їх порядок збережеться.

Так, частина кнопок зміниться, вимоги відрізнятимуться, та й результат вийде різний, але послідовність не зміниться.

3.2 Аналіз етапу «Референси»

Спочатку потрібно зрозуміти те, що моделюватиметься. Для цього потрібно зібрати референси – фотографії, відео та малюнки цього об'єкта, які тільки можна знайти в інтернеті.

Після референсів приділіть час, щоб проаналізувати зібраний матеріал. Зрозумійте композицію форм, механіку елементів та як вони пов'язані між собою. Переробляти на етапі сітки (чи ще далі) буде набагато складніше [36].

Референси необхідно групувати. Кожна категорія – окрема папка на компьютері або нова група в програмі PureRef.

Референси, як правило, групуються за нас наступними категоріями:

– Концепт - малюнок про те, як виглядає модель. Приклад концептів представлено на рисунке 3.2 [36]. Це її загальний вигляд, який затвердив арт-директор і гейм дизайнер (режисер у кіно). На кожній картинці потрібно бачити силует майбутньої моделі та великі форми, з

яких вона складається. Зазвичай на проектах є концепт дизайнери, які вигадують, як виглядатимуть об'єкти у грі/кіно. Якщо на проекті немає концепту дизайнера — він може візуально розвалюватися (всі моделі будуть у різному стилі).

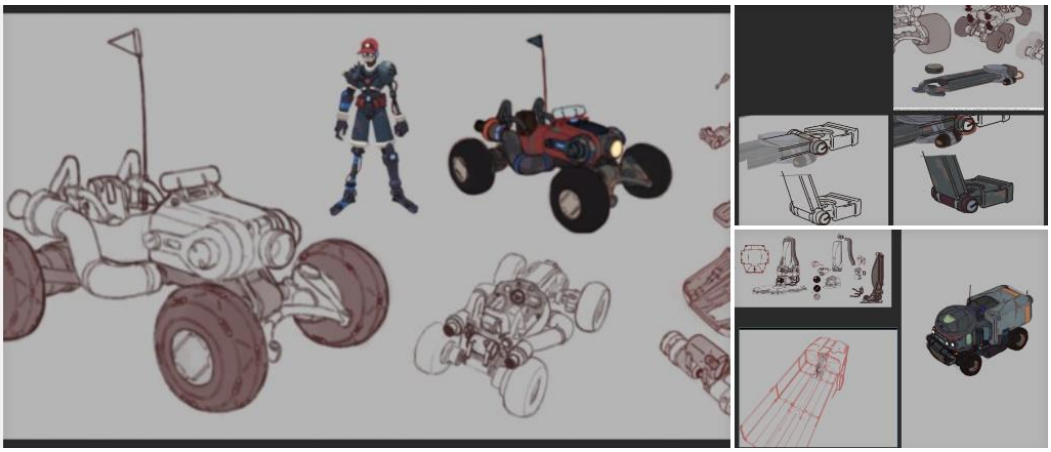


Рисунок 3.2 - Дошка за концептами [36]

– Фотографії — щоб краще зрозуміти силует та складові моделі, як на рисунке 3.3 [36]. Робота художника — це завжди наслідування реального світу, тому в першу чергу варто шукати референси в реальності чи на фотографіях, бо будь-який малюнок — це необ'єктивний референс.

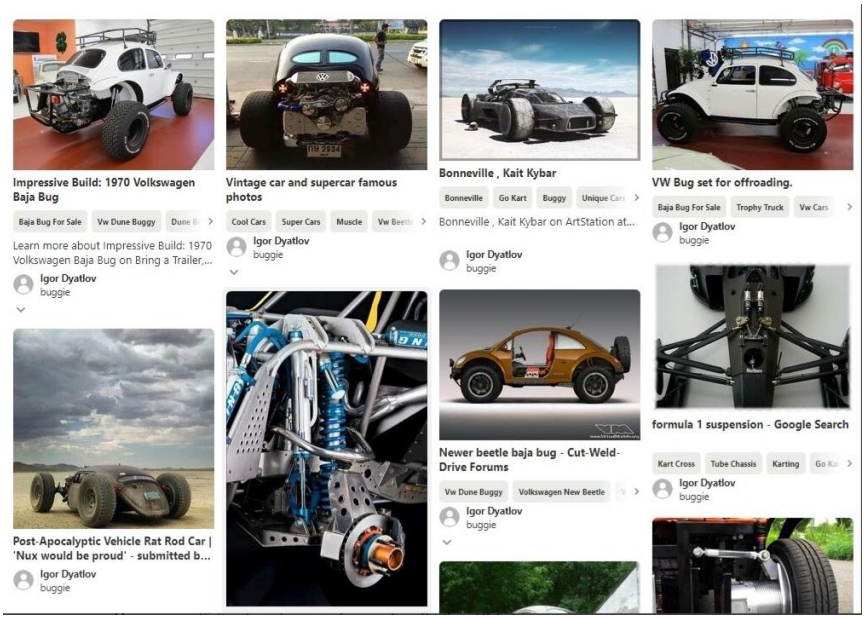


Рисунок 3.3 - Дошка з фотографіями на сайті pinterest [36]

– Роботи інших художників — швидше за все, хтось уже моделював цю річ. А її не тільки моделювали раніше, а й малювали в 2D і знімали в кіно. Можливо, якісь форми вони спрощували, якісь робили виразніше, як на рисунке 3.4 [36]. Досвід десятків інших художників заслуговує на окрему категорію референсів.

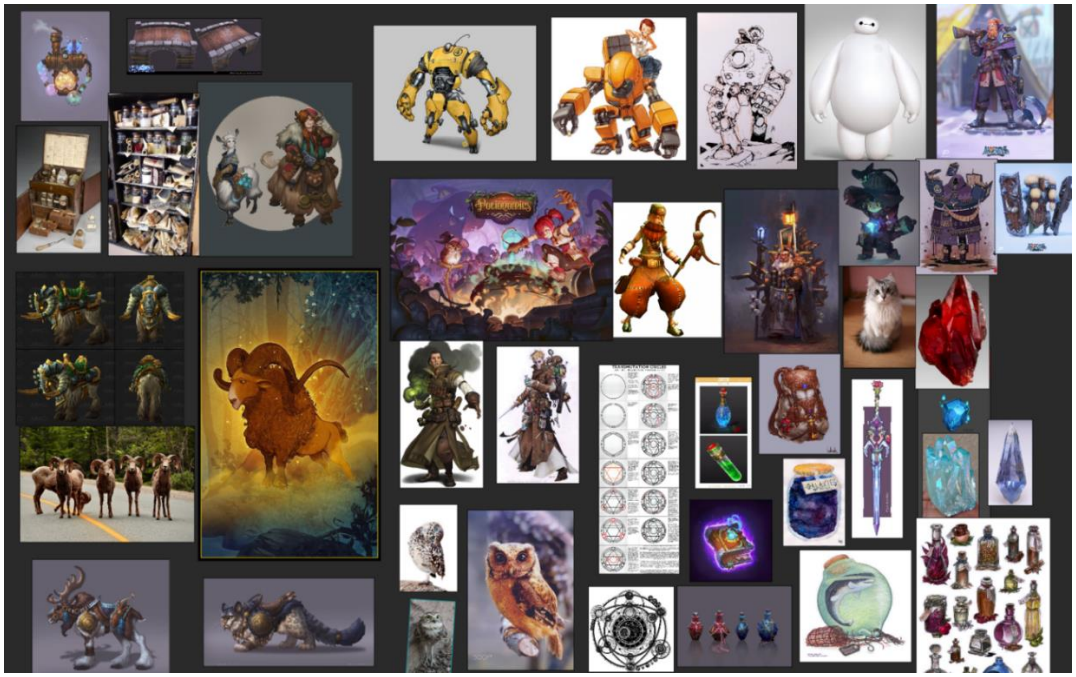


Рисунок 3.4 - Дошка референсів інших художників [36]

– Деталі — щоб краще розуміти, як функціонує об'єкт і як деталі кріпляться один одному, потрібно шукати референси деталізації. Часто це референси з різних об'єктів. На рисунке 3.5 [36] приклад дошки з рефами деталів.

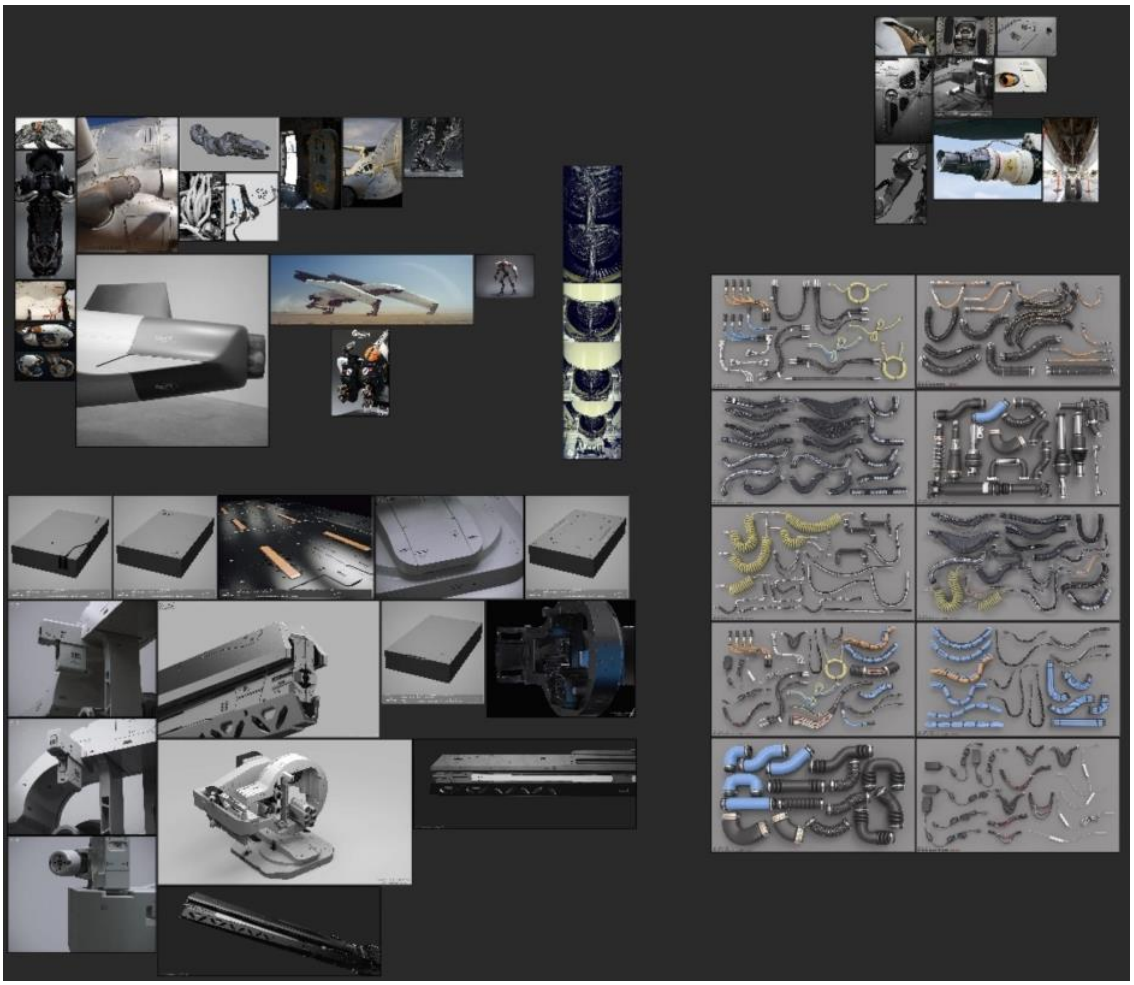


Рисунок 3.5 - Дошка з деталями об’єктів [36]

– Матеріали, текстури – важливо знайти хороші фотографії та арти з якісними текстурами та фактурами, як на рисунке 3.6 [36]. З голови хороший матеріал не зробити, тому дуже важливо запасатися рефами текстурок.

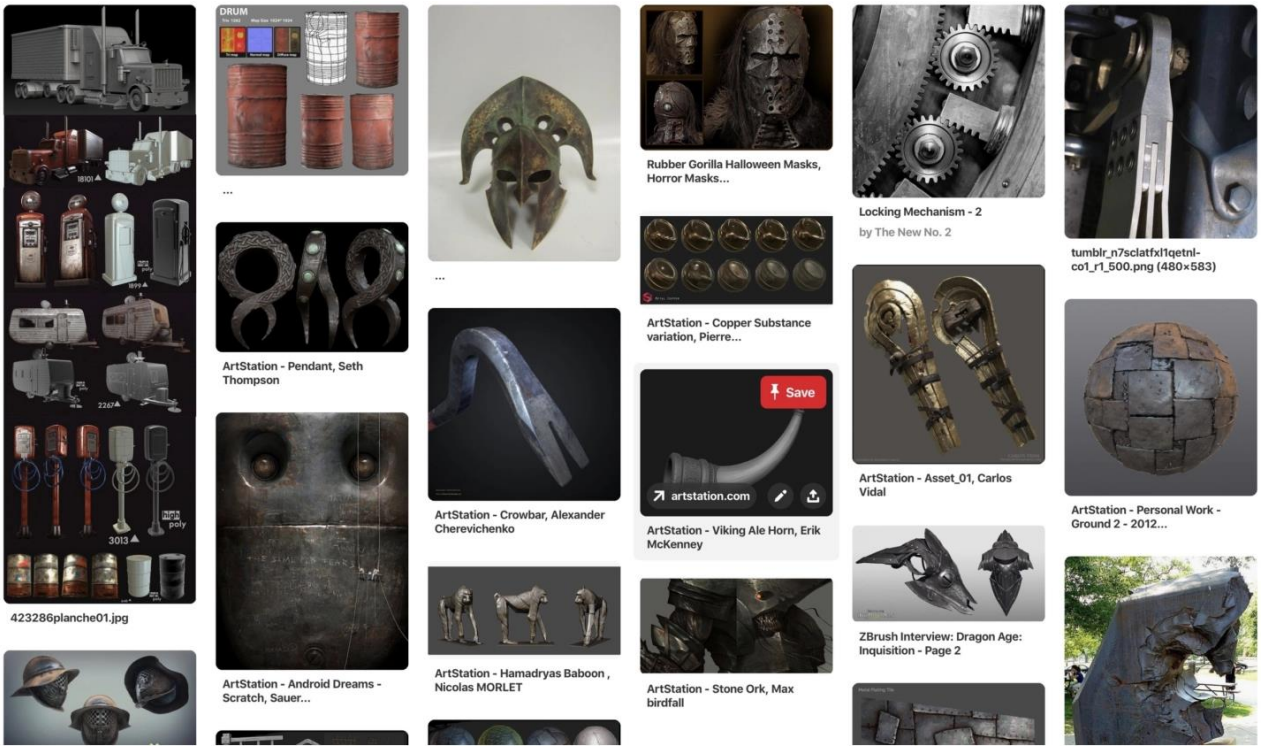


Рисунок 3.6 - Дошка з матеріалами [36]

– Ушкодження, потертості та інше – у різних об'єктів вони дуже відрізняються, це добре видно на рисунке 3.7 [36].

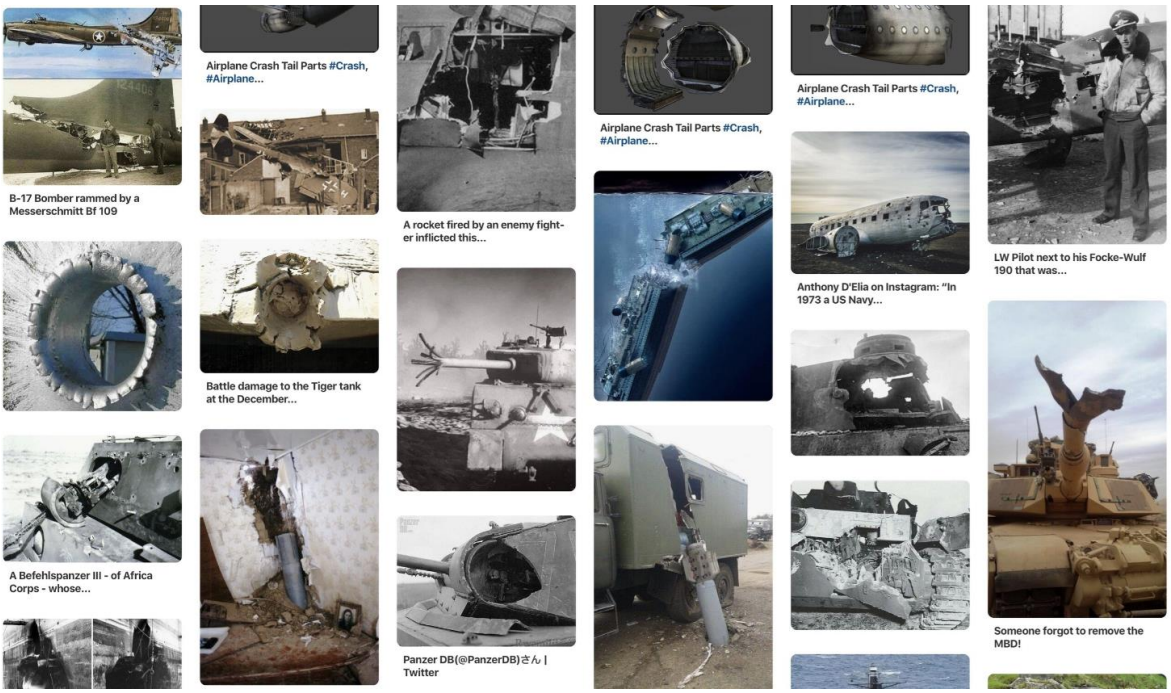


Рисунок 3.7 - Дошка з пошкодженнями [36]

Ушкодження - це складні форми, за якою видно внутрішню частину та пристрій об'єкта, і без референсів гарні пошкодження зробити неможливо.

3.3 Аналіз етапу «Драфт»

Кожна модель починається з драфта. А будь-який драфт починається з референсів. [37]. Будь-який драфт починається з блокінгу - начерк моделі з примітивів, який передає суть об'єкта. На етапі блокінгу немає жодної дрібної деталі, лише великі та середні форми. Все зроблено простими боксами, сферами та циліндрами, як на рисунке 3.8 [37].

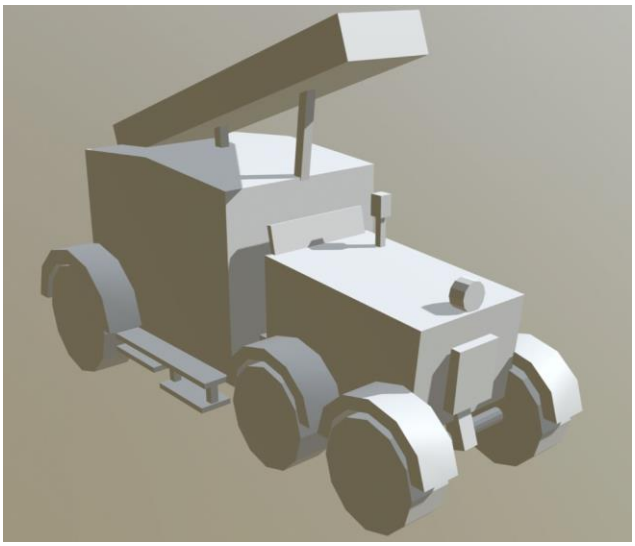


Рисунок 3.8 - Блокінг авто [37]

Блокінг - це робота над силуетом, правильними пропорціями моделі і робота над великими формами.

У середньому на блокінг йде до 40 хвилин.

Наступний етап драфту – це «деталізація». На цьому етапі дуже важливо розібратися з механікою моделі, щоб глядач міг повірити у її функціональність. Варто опрацювати переходи між геометрією та продумати важливі смислові деталі. Цей етап робить геометрію та силует цікавішим.

На цьому етапі треба розподілити кольори за моделлю, щоб краще розуміти її фінальний вигляд рис. 3.9 [37].

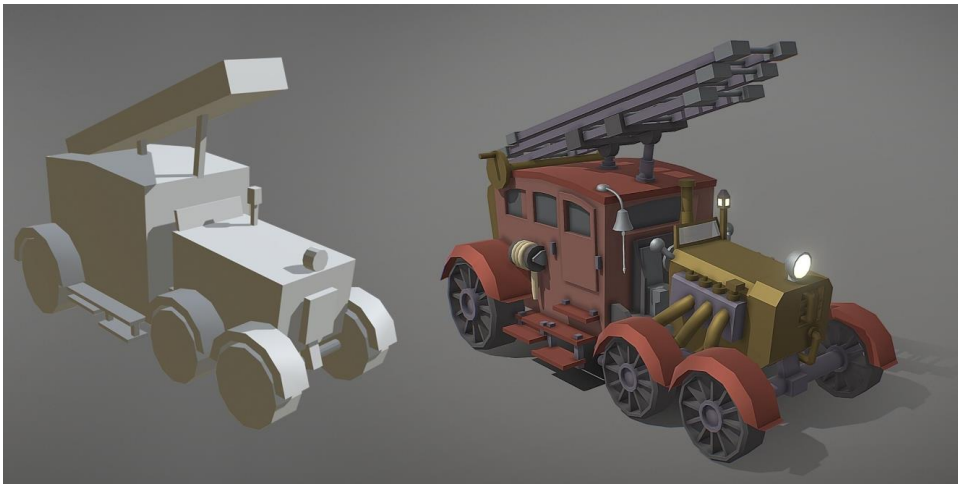


Рисунок 3.9 - Ліворуч блокінг; праворуч детальний драфт [37]

Саме на етапі драфта вирішується, як виглядатиме і читається модель. Якщо драфт вийшов невиразним, перероби його, поки не пізно, бо косяки на цьому етапі нічим не врятувати.

Форми, які зроблені на драфті, дуже складно змінюватиме протягом подальшої роботи над моделлю, тому драфт потрібно робити з усією відповідальністю перед фінальною моделлю. Це «артовий» етап роботи (крім текстур). Решта роботи – це технічні етапи.

3.4 Аналіз етапу «Сітка»

Є три види сітки: лоу-, мід-і хайполі

- LowPoly - спрощена та оптимізована модель для гри з мінімальною кількістю полігонів.
- HighPoly – максимально деталізована модель, яка потрібна, щоб перенести всю деталізацію на LowPoly через Normal Map
- MidPoly - компроміс між нескінченно деталізованими хайполі та оптимізованими лоуполі. Використовується здебільшого для кіно.

Лоуполі (Lowpoly) - це максимально легка 3D модель, в якій кожна площина, грань і вершина мають функціональне завдання: впливають на силует, правлять відблиск, вирішують завдання розгорнення і так далі.

При цьому полігони, яких не видно, видаляються, вся геометрія максимально оптимізується, циліндри будуються із збереженням довжини ребра та інше. На лоуполі багато нюансів, що впливають на розгортку та запічку.

Це та сама модель, яка буде у грі (рисунок 3.10 [37]). Саме її ми будемо розгортати, пекти і текстурити. Завдання лоуполі - знайти ідеальний баланс між виразністю та легкістю моделі. На реальних проектах часто видають бюджет на полігони. Наприклад, майже всі танки для World of Tanks містять менше 50 тисяч трикутників (але бувають винятки).

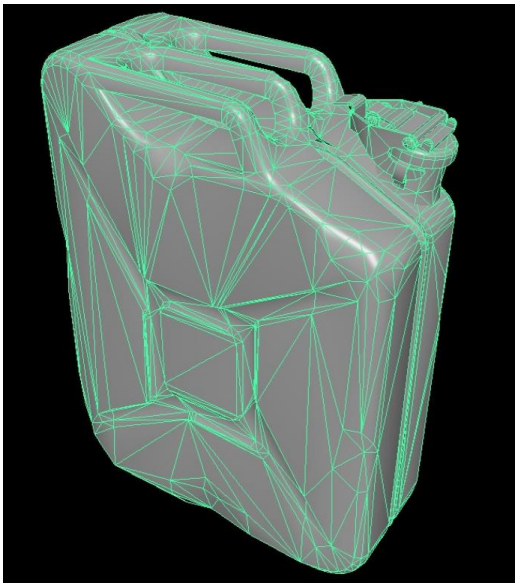


Рисунок 3.10 - Оптимізована лоуполі модель [37]

Лоуполі для різних ігор різняться. На мобільних пристроях лоуполі дуже легка, 2к – 10к трикутників. На деяких проектах ААА люблять вшивати всю геометрію у величезні цілісні форми, а на інших кожну деталь відбивають окремою геометрією. Є проекти, де на головного персонажа виходить 50-60к трикутників, а бувають проекти по 250к трикутників на персонажа.

У сучасній грі модель на передньому плані може бути більш деталізованою, ніж у кіно на середньому та задньому плані — кіно виглядає реалістичніше через більш просунуте світло і набагато більші текстури. Кадр фільму може вважатися на супер комп'ютері кілька годин, а відео гра повинна видавати цілих 60 кадрів на секунду - тому навіть більш детальні моделі в іграх виглядають не так соковито, як модельки в кіно (і багато залежить від гри).

Лоуполі дуже залежить від технології. На мобільній грі на важливому об'єкті 5-7к полігонів. У старих іграх важливі об'єкти були у 10-15-20к полігонів. Зараз ігри легко тягнуть моделі по 50-150 до полігонів. Бувають і вище, по 200-300 до полігонів на величезні моделі, наприклад, кораблі/танки/будинки. На лоуполі бувають різні правила. Одні моделі робляться цілісними шматками геометрії, інші розбиваються на деталі.

Ліміт на лоуполі вираховується технічним директором. Припустимо, технічний директор вирахував обмеження на локацію в 10 млн. полігонів і 200 матеріалів. Звідси вираховується. За 150-200 моделей. У середньому від 10 до 100 до полігонів. Це обмеження взялося з продуктивності двигуна і заліза, під яке робиться гра. І, звичайно, без досвідченого технічного директора такі ліміти не вирахувати [38].

Хайполі (Highpoly) – це дуже деталізована модель. На ній немає обмежень за кількістю полігонів (головне обмеження – щоб модель взагалі можна було відкрити на твоєму комп'ютері).

Перше завдання на хайполі – зробити круглі фаски, фактури та деталі, які не потрапили на лоуполі. Хайполі – це етап, коли можна все. Декілька технологій хайполі: SubDivide, Crease edge і скульпт. Хайполі потрібна лише для запікання деталей та фасок на лоуполі.

Хайполі не обмежена кількістю полігонів. Головне обмеження – щоб файл відкрився на комп'ютері.

Хайполі роблять без особливих обмежень по сітці, в ній воліють квадрати (ними простіше будувати полігональні лупи - про це трохи пізніше),

а від незграбних позбавляються на згладжуванні. У ігровий двигун таку сітку не засунеш, зате вона може бути скільки завгодно деталізованою і гладкою.

3 способи зробити хайполі

1) Сабдив (subD) – це створення хайполі через підтримки (рисунок 3.11 [38]). Необхідно створити форми, накласти підтримки, вони правильно округляються, виходить максимальна точність.

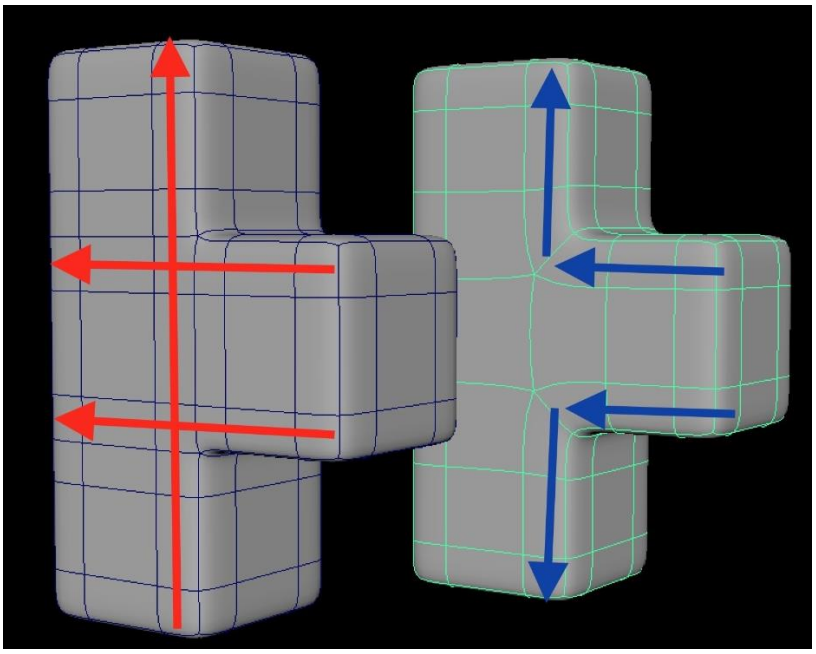


Рисунок 3.11 - Приклади моделювання під сабдів [38]

Це стара та якісна технологія, завдання якої робити круглі фаски та гладкі форми. Найважливіше на сабдіві — зрозуміти, як будувати лупи та кільця полігонів, щоб вони наголошували на твоїй формі.

2) Скульпт - необхідно, якщо модель має м'які форми, органіка, неточні, нерівні форми. Це як пластилін, лише у 3д (рисунок 3.12).

Рисунок 3.12 - Приклад скульптингу персонажа

Скульпт робить сітку настільки щільною, що мне її як пластилін у спеціальних програмах (zBrush, mudbox, 3d coat). Головна особливість скульпту – можна працювати з десятками та сотнями мільйонів полігонів.

3) Геометрія CAD. Є програма fusion360 (рисунок 3.13 [38]) . Її зробили для інженерів, щоб вони проектували та збирали реальні мости, машини, роботів та інші штуки для реального світу. Але програма виявилася настільки зручною, що в ній почали скористатися концепт художники. Щоб робити роботів та залізо.

Рисунок 3.13 - Приклад моделі в Fusion360 [38]

Ще Sketchup дуже гарний у кад моделюванні. Від звичайного моделювання CAD відрізняється тим, що кожна поверхня в ньому задається формулою, тому вона може бути як завгодно гладкою і плавною, і її легко редагувати на будь-якому етапі. Але у відеоіграх такі моделі не використовують, тому найчастіше CAD моделі можна побачити в концепт арті під малювання.

Мідполі (Midpoly)- компроміс між нескінченно деталізованими хайполі та оптимізованими лоуполі (рисунок 3.14 [38]).

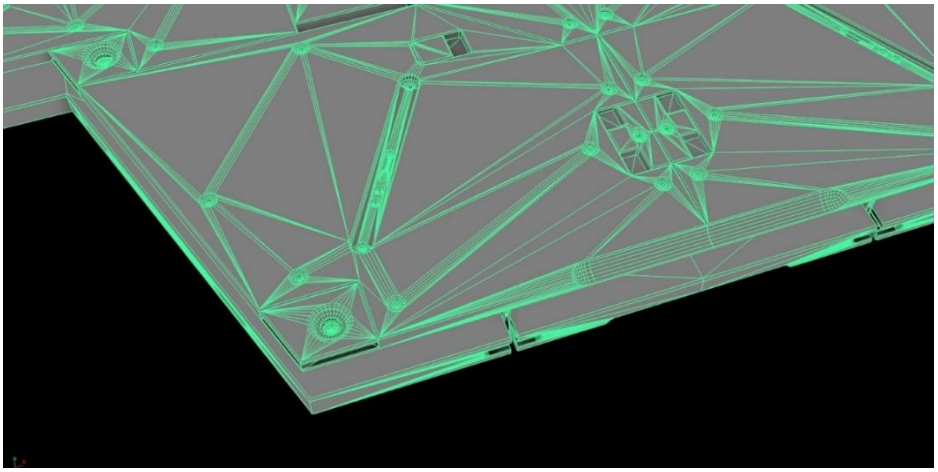


Рисунок 3.14 - Приклад midpoly моделі [38]

З цією сіткою роблять супер детальні та цікаві моделі, які кльово виглядають у кадрі, добре шейдять, але в той же час вони досить оптимізовані та легкі, для зручності текстурування та роботи з ними.

3.5 Аналіз етапу «Ретопологія»

Ретопологія – це процес створення лоуполі сітки на основі хайполі моделі.

Після скульптингу є хайполі модель, але в ній завелика кількість полігонів і дуже хаотична та щільна сітка (рисунок 3.15 [38]).

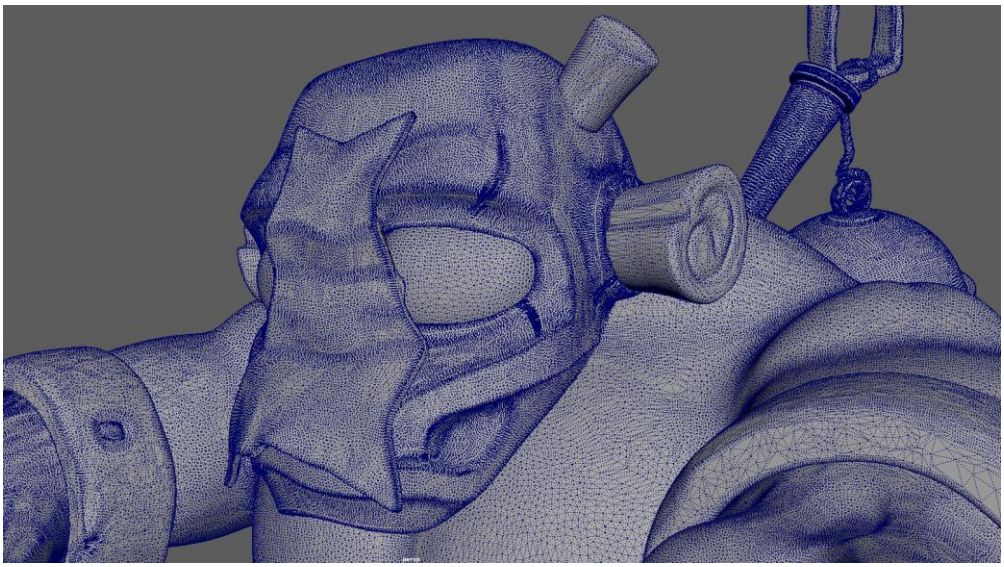


Рисунок 3.15 - Сітка після скульптингу

Майже у всіх випадках після скульптингу буде така неякісна сітка.

За допомогою інструментарію в 3д-софті необхідно створити правильну топологію (рисунок 3.16).

Рисунок 3.16 - Приклад правильної ретопологіїї

Необхідно слідкувати за щільністю сітки, вона повинна бути рівномірною. Занадто витягнуті та криві полігони, різного розміру – погана практика. Ігрові движки не дуже добре їх сприймають. Полігони повинні повторювати форму вихідної геометрії.

По можливості потрібно уникати гострих кутів, але у випадках коли не виходить цього зробити, зазвичай використовують хард ежі, або додають додатковий бевел.

Складки на одязі краще робити трикутниками, щоб краще повторювати форму, як це зображено на рисунку 3.17 [38].

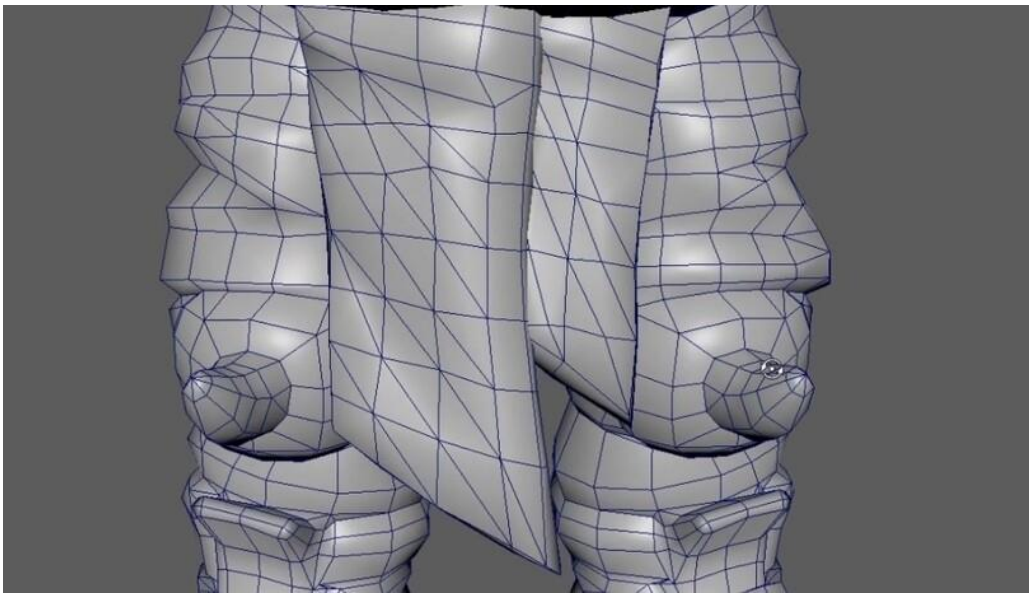


Рисунок 3.17 - Приклад складок [38]

Пальці на руках будуть анімуватися тому необхідно підсилити сітку у місцях згину кісток, як це представлено на 3.18.

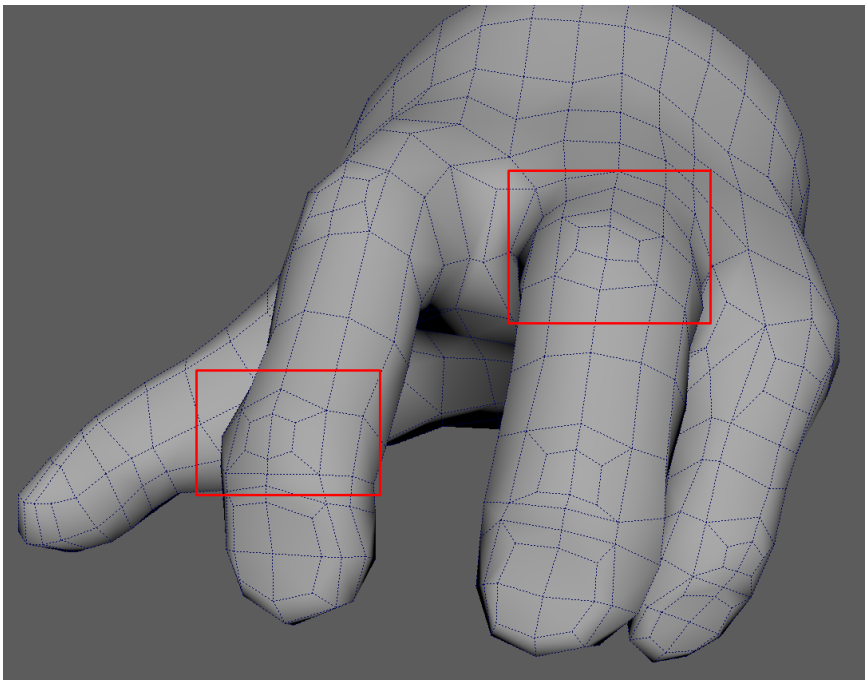


Рисунок 3.18 - Приклад реалізації сітки на кістках рук

З такою сіткою при анімації не будет потягів на текстурі.

Полігони повинні повторювати форму м'яза і утворювати лінію, як на рисунку 3.19 [38].

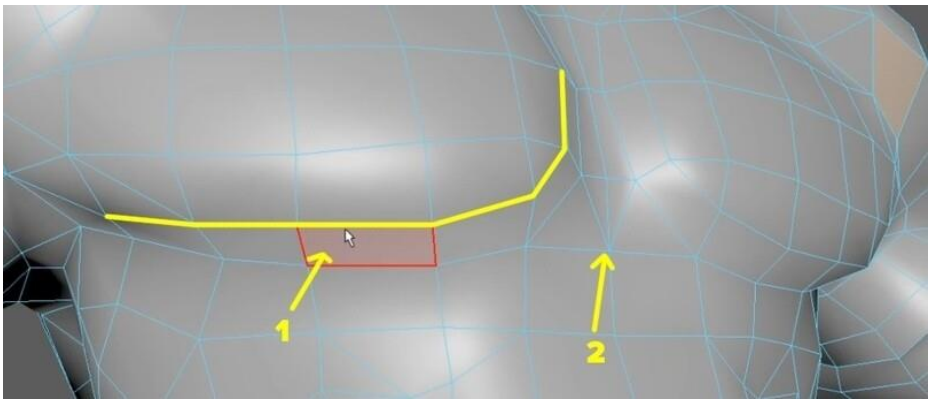


Рисунок 3.19 - Приклад реалізації м’яз [38]

Якщо підчеркнути форму м’язі відповідною сіткою, то вони будь краща читатися та анімуватися.

Ретопологія лиця – це велика тема. Складність у тому, щоб створити сітку, яка надалі реалістично анімуватиметься, як на рисунку 3.20 [38].

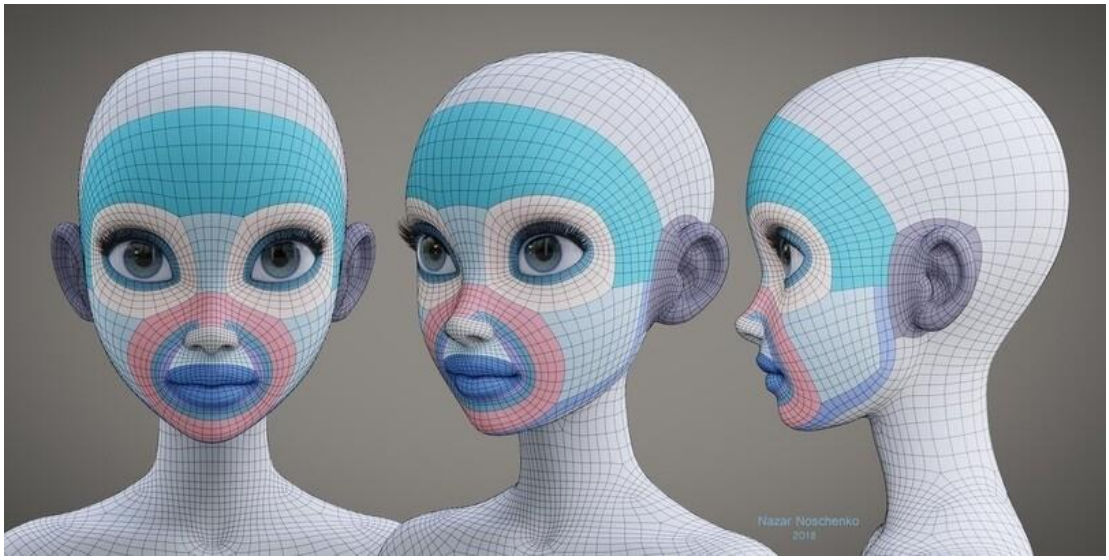


Рисунок 3.20 - Приклад правильної ретопології лиця [38]

3.6 Аналіз етапу «UV-розгортка»

3D модель, за визначенням, виготовлена з об'ємних форм. На будь-який об'єкт у редакторі можна покласти матеріал, вибрати його колір та налаштувати відблиск. Але покласти текстури на 3D об'єкти не можна доти,

доки зроблено розгортка. Програма просто не знає, як накладати плоску текстуру на геометрію.

В орігамі з плоского аркуша паперу виходить об'ємний об'єкт [39]. Розгортка робить те саме, але навпаки — з об'ємного робить плоске, я на рисунку 3.21 [39].

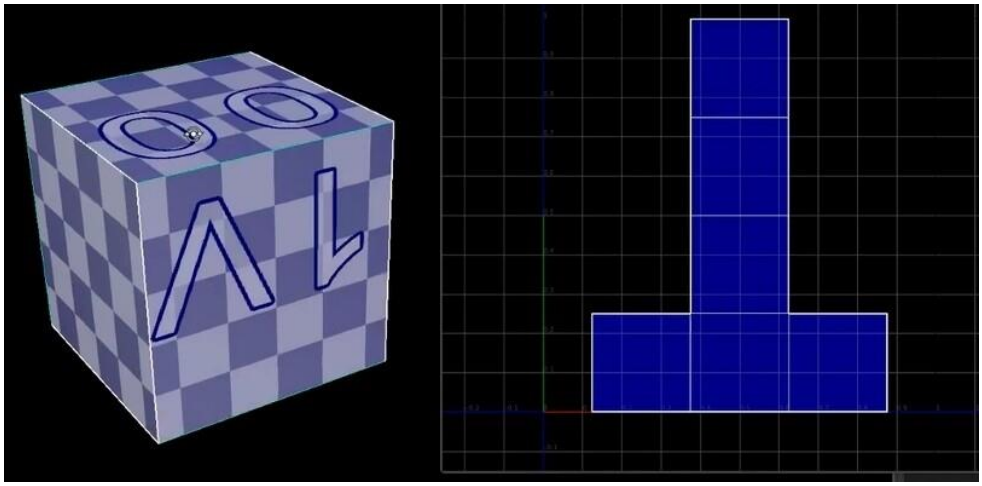


Рисунок 3.21 - UV розгортка куба [39]

Завдання на розгортці - нарізати модель на площині з найменшою кількістю швів, переконатися, що текстури не тягнуться, острівці не перетинаються (щоправда, є виняток у вигляді роботи з атласами), елементи, що повторюються, лежать один на одному, і щільність пікселів на текстурі відповідає вимогам проекту [40].

Якщо на модель дбає карта нерівностей - то на всіх жорстких ребрах робиться розріз на розгортці.

З чого складається розгортка.

Як видно на рисунку 3.22 [40], у куба є 8 вертексів и 6 квадратних полігонів.

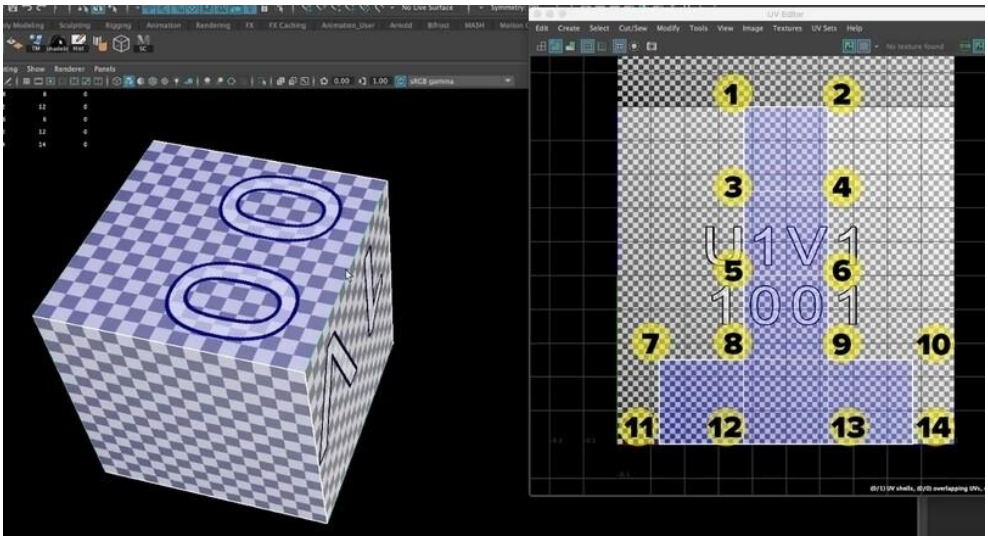


Рисунок 3.22 - Відмінності куба від її розгортки [40]

Тепер дивимося на розгортку і бачимо, що на розгортці також 6 квадратів. Але кількість верткесів на UV більше, ніж на самій моделі – на розгортці їх цілих 14. Щоб зрозуміти, звідки вони з’явились, необхідно виділити на кубі один верткс, як на рисунку 3.23 [40].

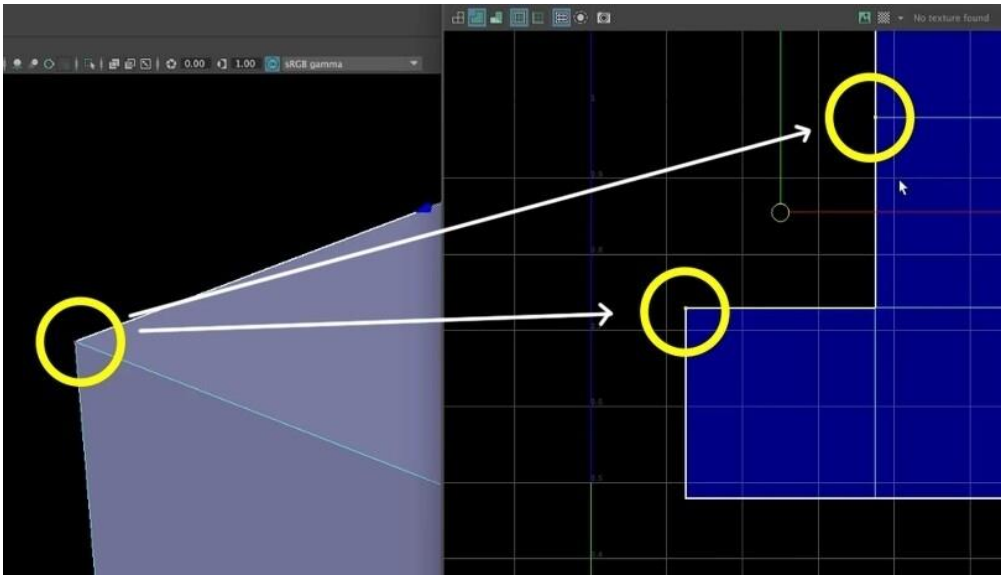


Рисунок 3.23 - Виділений вертекс куба [40]

Один вертекс на моделі може мати декілька вертексів на UV.

Насправді, на перетині полігонів відбувається таке: у кожного вертексу, крім положення у просторі та нормалі, є ще один параметр – становище цього вертексу на UV.

Кожен полігон має своє місце на UV як плоский об'єкт, щоб програма знала, як класти текстуру на геометрію.

Розглянемо два існуючих вида мапінгу:

1) Зробити унікальний мапінг;
2) Зробити тайловий мапінг.

Перед тим як детальніше розбиратись з кожнім з типів мапінгу, звернемо увагу на таке: на UV просторі багато квадратів, як на рисунке 3.24 [40]. Насправді вони нескінченні.

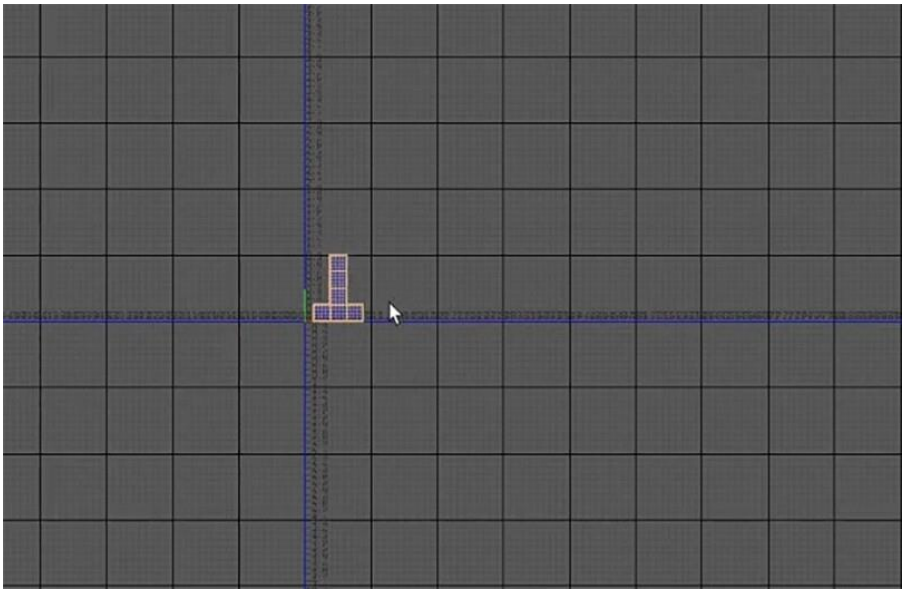


Рисунок 3.24 - UV простір [40]

Нескінченний тайловий простір. Квадрати будь-коли закінчаться, кожен із них повторює текстуру з першого квадрата, а вмістити розгортку треба у квадрат 0-1.

Але нас особливо цікавить саме перший квадрат на UV, який знаходиться від 0 до 1. Саме на нього накладається текстура, а на решті квадратів текстура просто повторюється.

Якщо розгортати модель на UV так, що всі об'єкти знаходяться в першому квадраті і не виходять за межі, це називається унікальним мапінгом.

3.6.1 Використання унікального мапінгу

Один матеріал має лише одну єдину текстуру. І вона повністю розтягується на перший UV квадрат. Тепер уявімо, що необхідно зробити мапінг вантажівки. У нього є текстура з деревом для кузова та текстура з фарбованим металом для кабіни (рисунок 3.25 [40]).

Рисунок 3.25 - Модель вантажівки для мапінгу [40]

В цілому, можно зробити 2 матеріала. Один під дерево, інший під фарбований метал. І вже мапити об'єкти на ці два матеріали. Але є проблема. Кожен новий матеріал - це зайвий виклик малювання (draw call), який навантажує ігровий двигун. А модель – вантажівка для гри. Нам дуже важливо використовувати мінімальну кількість ресурсів.

Тому замість того, щоб плодити купу зайвих матеріалів, ми робимо всього один матеріал. Спочатку ми розгортаємо всі об'єкти на UV, потім кидаємо у фотошоп або Substance Painter та фарбуємо ці елементи окремо.

На виході отримуємо одну квадратну текстуру, створену спеціально для нашої моделі (рисунок 3.26 [40]).

Рисунок 3.26 - Приклад унікального мапінгу [40]

Це називається унікальним мапінгом. Це технологія, яка полягає в наступному:

- Ми мапимо всі об'єкти на UV тільки до першого його квадрата. Жоден із шеллів не виходить за його межі, і шелли не перетинаються (крім оверлапів, про які поговоримо сьогодні трохи пізніше).

- Об'єкти пофарбовані по-різному (кузов дерев'яний, кабіна із фарбованого металу). Причому це не повторювані, а унікальні текстури.

- Але на виході ми маємо одну єдину текстуру (не рахуючи карти нормалі, рафнесу та металнесу) і один матеріал, тим самим економимо ресурси ігрового двигуна.

3.6.2 Використання тайлового мапінгу

Квадрати на UV нескінченні. Це означає, що текстура, яку кладеться в перший квадрат, буде повторюватися нескінченну кількість разів. Це нескінченне повторення текстур називається тайлом.

Тайловий мапінг дуже часто використовується для розгортання будівель, предметів оточення та ландшафтів.

Припустимо, що потрібно зробити розгортку цього будинку (рисунок 3.27 [40]):

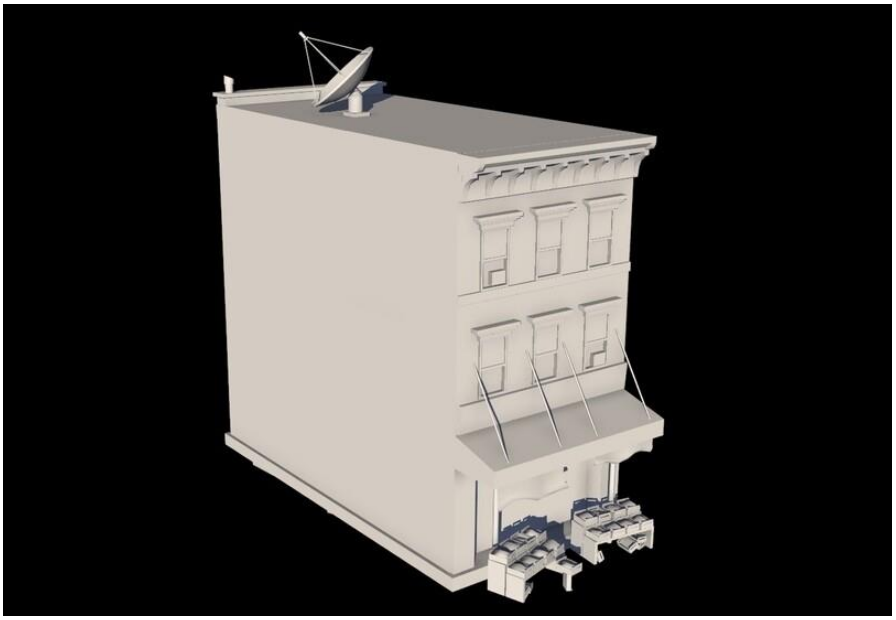


Рисунок 3.27 - Будинок для мапінгу [40]

Спочатку створюємо матеріали з безшовними текстурами. Потім кладемо їх на геометрію за допомогою авторозгортки (box maping).

І так робимо для кожного із об'єктів. Для стіни будинку один матеріал з цегляною кладкою. Для прилавка інший матеріал з текстурою дерева. І так далі.

Навіщо це потрібно? Припустимо, що є безшовна текстура з цеглою. Якщо розгортка всіх стін буде в першому квадраті, то цегла буде занадто великою. Оскільки текстура з першого квадрата нескінченно повторюється, а текстура не має швів — ми просто змінюємо розмір шеллів на UV і підбираємо відповідний для нас розмір (рисунок 3.28 [40]).

Рисунок 3.28 - Будинок після розгортки та текстур [40]

В автомапінгу не так важливо як виглядає UV розгортка. На відміну від унікального мапінгу, немає сенсу вміщати всі шелли всередині першого квадрата та економити простір на UV. Головне, щоби текстури виглядали акуратно і без потягів рисунок.

3.7 «Аналіз етапу Bake»

Запікання (Bake) – четвертий етап AAA-пайплайну. У минулому була зроблена розгортка, а зараз вона буде використана для створення Normal Map, AO та Color ID [41].

Погляньмо на low poly модель (рисунок 3.29 [41]).

Рисунок 3.29 – Lowpoly [41]

У ній мінімум полігонів та чиста сітка. Ця модель завантажуватиметься в ігровий двигун.

Є ще high poly модель з високою деталізацією та скульпт (рис. 3.30 [41]).

Рисунок 3.30 - Highpoly та Sculpt [41]

High poly модели под сабдив и скульпт.

Технологія запічки дозволяє перенести всі деталі з high poly та скульпта на low poly-модель.

У результаті, в ігровому двигуні буде low poly модель з мінімальною кількістю полігонів, але виглядатиме вона так, ніби на ній є всі ці деталі (рисунок 3.31 [41]):

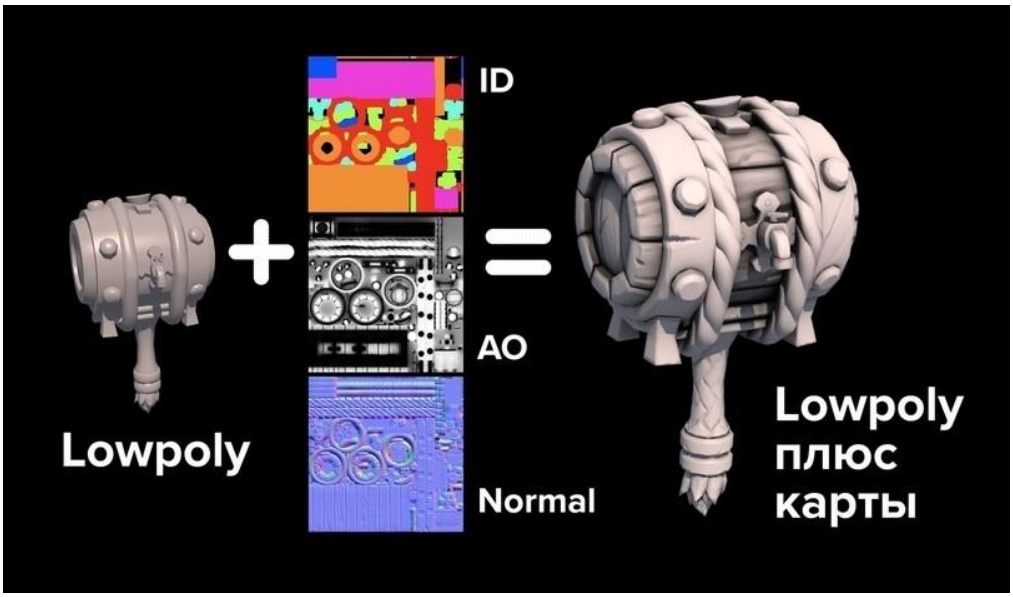


Рисунок 3.31 - Lowpoly з деталізацію Highpoly [41]

ID Map ніяк не впливає на вигляд моделі, на відміну від нормалу та AT, але вона стане в нагоді при текстуруванні, тому цю карту теж пектимемо.

Беремо low poly модель і додаємо на неї запечені карти нормалу та AT. Ці карти дурять поведінку ігрового світла. Модель починає блікувати так, начебто на ній є всі ці фаски, вирізи та інші деталі.

3.7.1 Використання Normal map

Normal Map перекладається як "карта нормалей". Вона змінює напрямок відблиску на геометрії (рисунок 3.32 [42]).

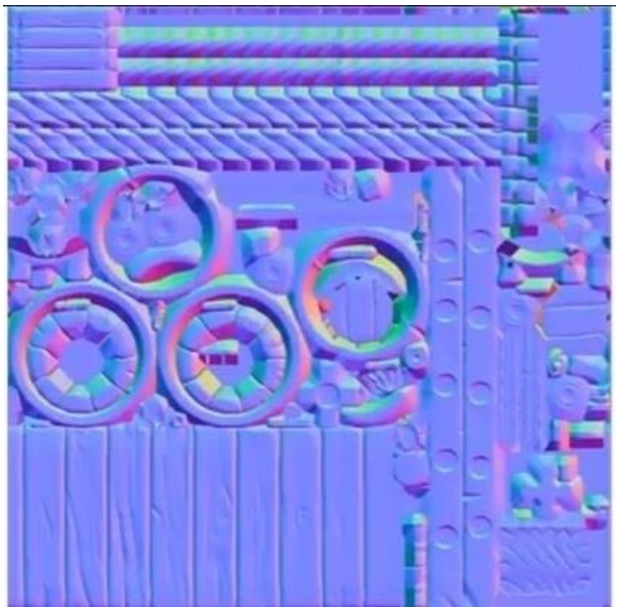


Рисунок 3.32 - Normal map [41]

Карта створює віртуальні вертекс нормалі в кожній точці low poly моделі і спотворює поведінку світла. Якщо накласти запечений нормал на модель, вона виглядатиме майже як high poly [42].

Нові полігони не створюються, це лише ілюзія форми (рисунок 3.33 [41]):

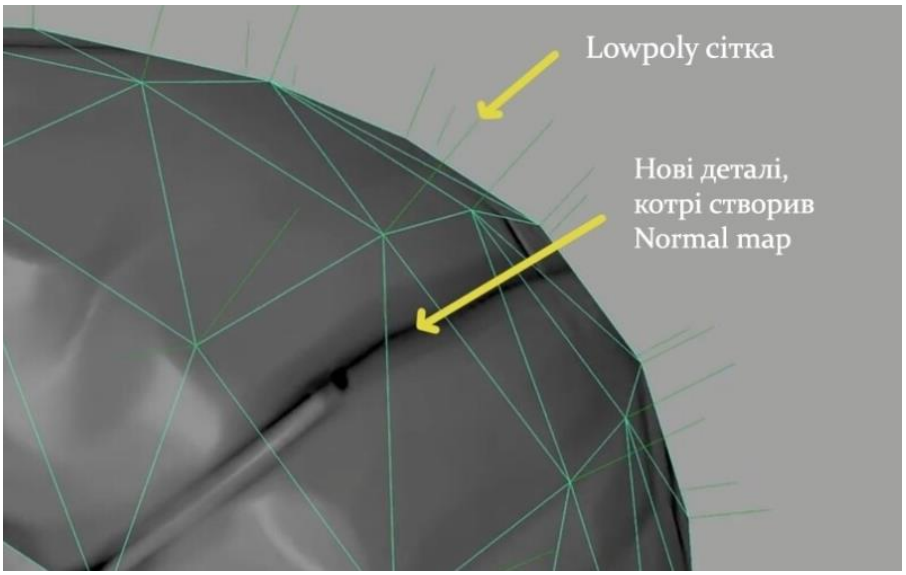


Рисунок 3.33 - Деталі з карти нормалі [41]

Всі деталі та затінення на цій low poly сітці – ілюзія.

Щоб створити нормал, потрібно взяти розгортку, яку ми робили раніше, взяти high poly та скульпт, завантажити у програму для запічки та натиснути кнопку «bake».

Під час запікання карт нормалей могуть появитись чорні полосі на нормалі (рисунок 3.34 [41]) – це нормально.

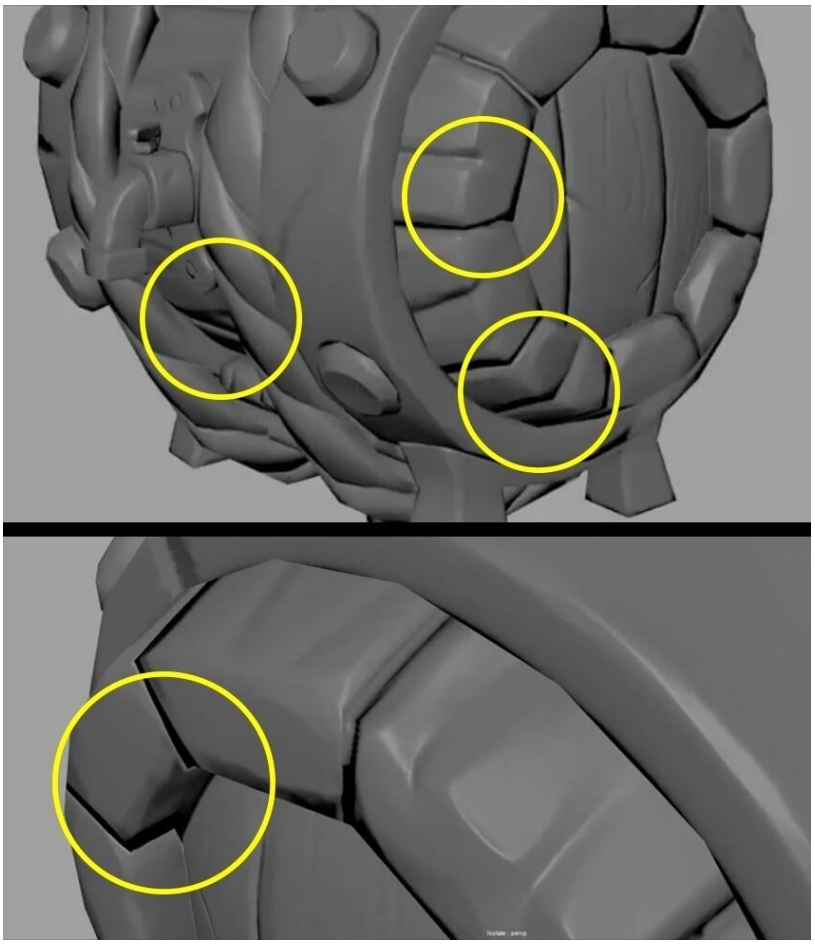


Рисунок 3.34 - Чорні полоси на нормалі [41]

З чого складається карта нормалей.

Normal map створює ілюзію обсягу завдяки трьома картами в каналах текстури: червоний, зелений, синій.

Щоб перемикатися між каналами, відкриємо файл із нормалом у Photoshop і зайдемо у вкладку Channels (рисунок 3.35 [41]).

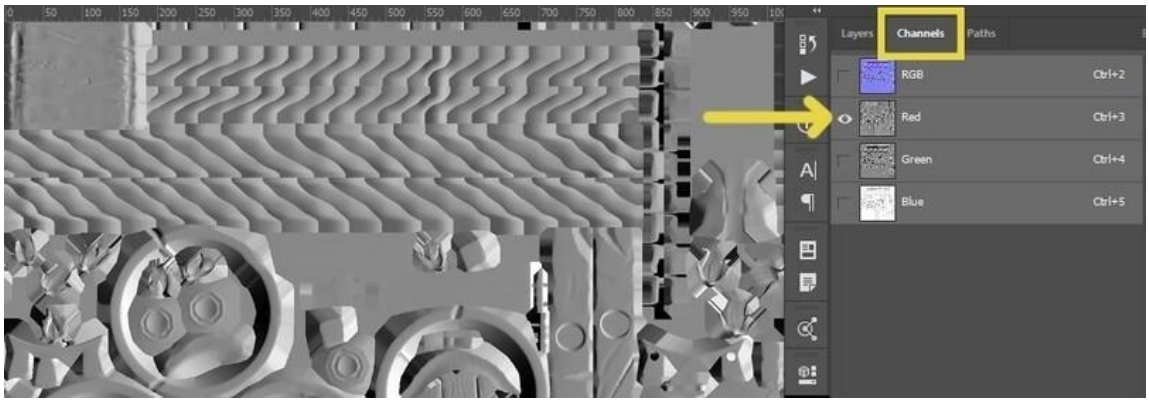


Рисунок 3.35 - Нормал карта у фотошопі [41]

Червоний канал показує спотворення вертекс нормалей по горизонталі рисунок (рисунок 3.36 [41]).

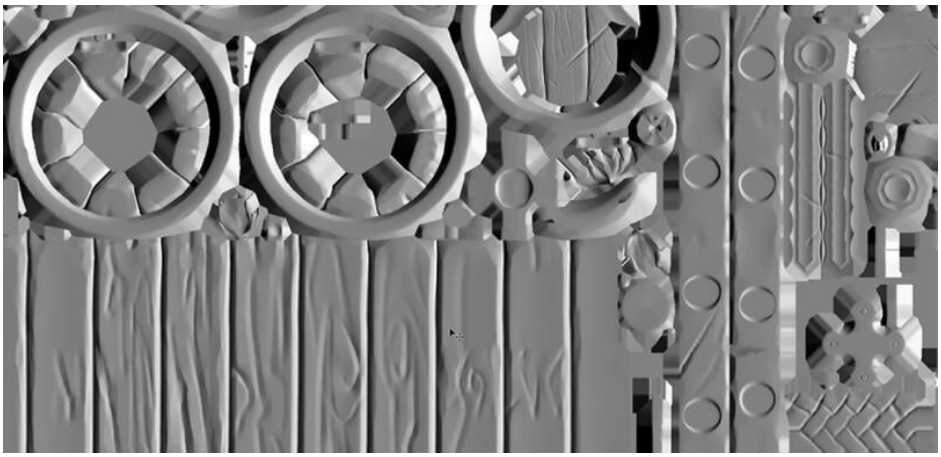


Рисунок 3.36 - Червоний канал нормала [41]

Чим світліша пляма на червоному каналі — тим сильніша віртуальна поверхня нахилена вправо, а чим темніша — тим більше «поверхня» блищить вліво.

Зелений канал працює так само, але спотворює шейдинг по вертикалі (рисунок 3.37 [41]).

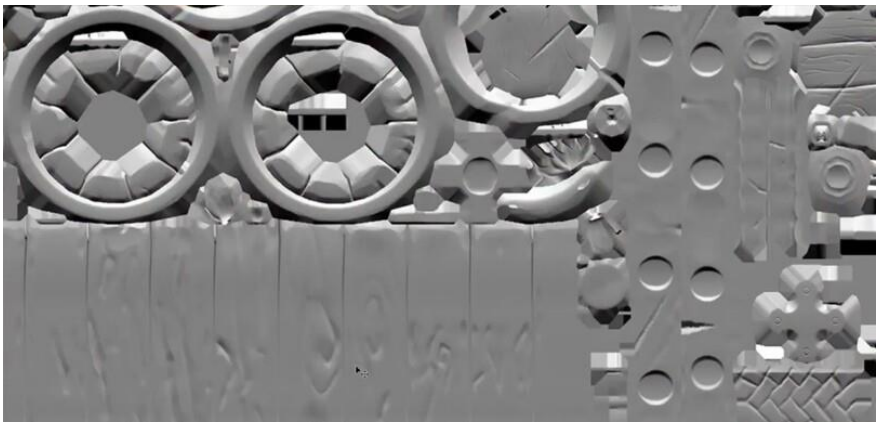


Рисунок 3.37 - Зелений канал нормалу [41]

Світлий-відблиск повертається вгору, а темний - вниз.

Синій канал імітує заглиблення в об'єкті. В іграх він практично не використовується (рисунок 3.38 [41]).

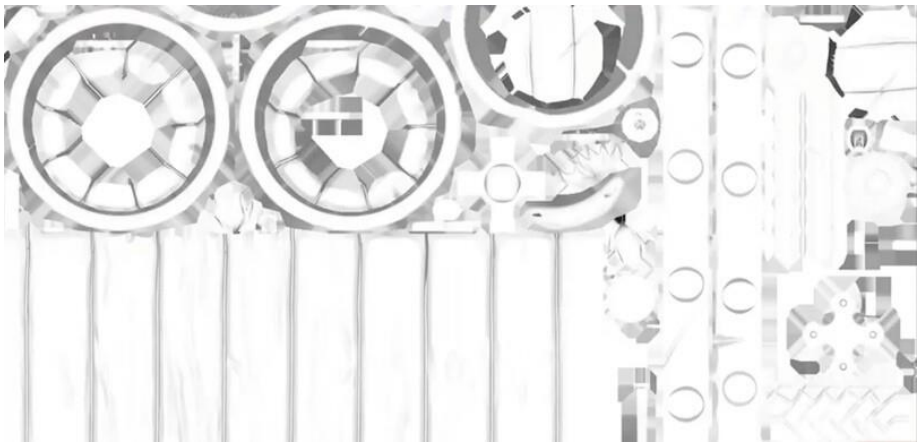


Рисунок 3.38 - Синій канал нормалу [41]

За рахунок зміни синього кольору на нормалі утворюються спотворення на плоскій поверхні.

Якщо на синьому каналі є артефакти, можна просто вимкнути цей канал, тобто відключити синій колір у нормал карту у фотошопі. Або просто їх замазати артефакти пензликом.

3.7.2 Використання Ambient occlusion map

Карта Ambient occlusion (AO) — це карта затенення (рисунок 3.39 [41]).

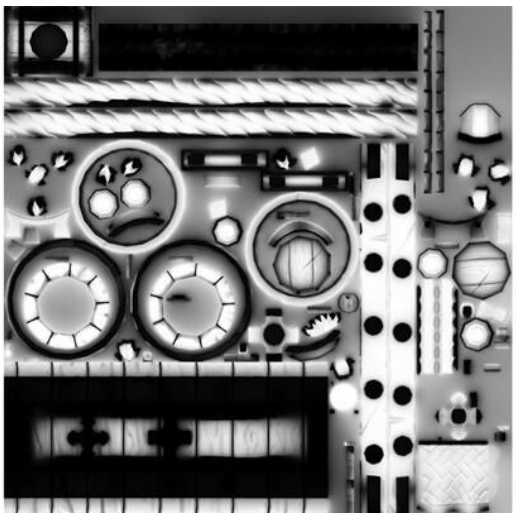


Рисунок 3.39 - Карта затемнення [41]

Вона показує найглибші тені в об'єкті — в основному це тені в поглибленнях і на пересіченнях. Ця карта використовується в ігрових рухах для створення коректного освітлення, а також вона дуже корисна в текстуруванні.

Відмінний приклад того, що складно зробити без AO — це грязь (рисунок 3.40 [41]).

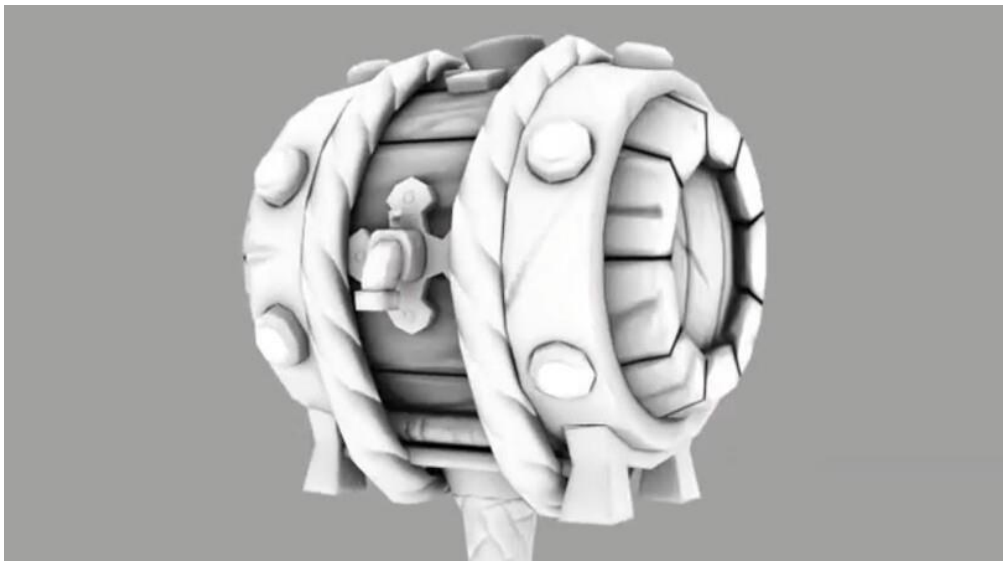


Рисунок 3.40 - Приклад затемнення на моделі [41]

Вона забувається в самих глибоких місцях моделі, — карта AO допоможе легко сгенерувати маску углиблений, щоб забити їх грязью.

3.7.3 Використання Color ID map

Наша модель складається з різних матеріалів: метал, камінь, дерево і тканини.

Під час текстурування нам потрібно швидко виділити геометрию, яка буде, наприклад, металом або деревом.

Щоб це можна було зробити одним клацанням миші всередині Substance Painter, нам потрібно запекти ідентифікатор кольору карти (рисунок 3.41 [41]).

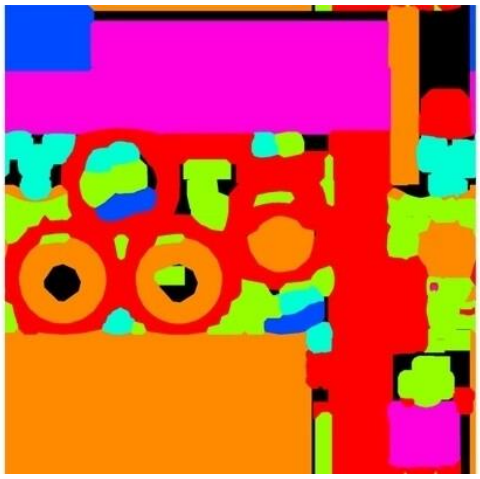


Рисунок 3.41 - Карта ідентифікатора кольору [41]

Вона дозволяє швидко виділити більші куски. А в програмі для текстурування цієї групи об'єктів можна легко затекстурувати.

Для того, щоб створити ідентифікатор кольору карти, нам потрібно пофарбувати модель high poly в різні кольори. Один колір — це одна група об'єктів, як правило розбита по матеріалам. Наприклад, все дерево на Color ID оранжеве, вся шкіра — рожева, ржавий метал — синій, шерсть — голуба, а новий і чистий метал — зелений.

Це не колір майбутньої моделі — тут вони можуть бути будь-якими. Главное, чтобы они были контрастными.

3.8 Аналіз етапу «Текстурування»

3.8.1 Використання маски

Маска в текстуруванні це зображення, яке складається з 1-го каналу і необхідне регулювання інших каналів і візуалізації параметрів фактур (рисунок 3.42 [43]).

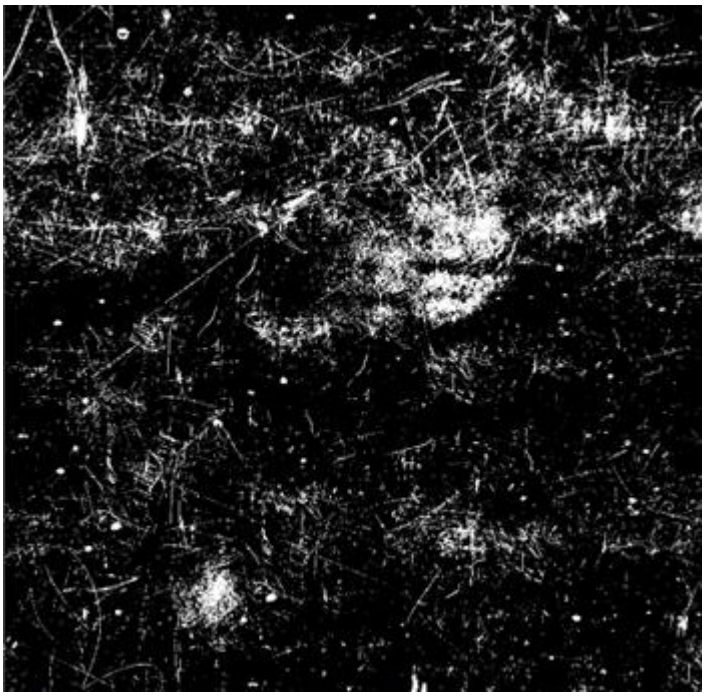


Рисунок 3.42 - Приклад маски [43]

Тобто маска використовується для вирішення 2-х завдань:

- Визначення ступеня прозорості.
- Визначення інтенсивності чогось у пікселі (про це пізніше, але обов'язково).

Насправді ділити на два завдання роботу масок не обов'язково — і там і там використовується ступінь інтенсивності, просто в першому випадку для прозорості, а в другому для цілей регулювання зовнішнього вигляду текстури [43].

Шари – це заливка на всю площу проекту, яка має купу каналів для контролю стану зображення (рисунок 3.43 [43]).

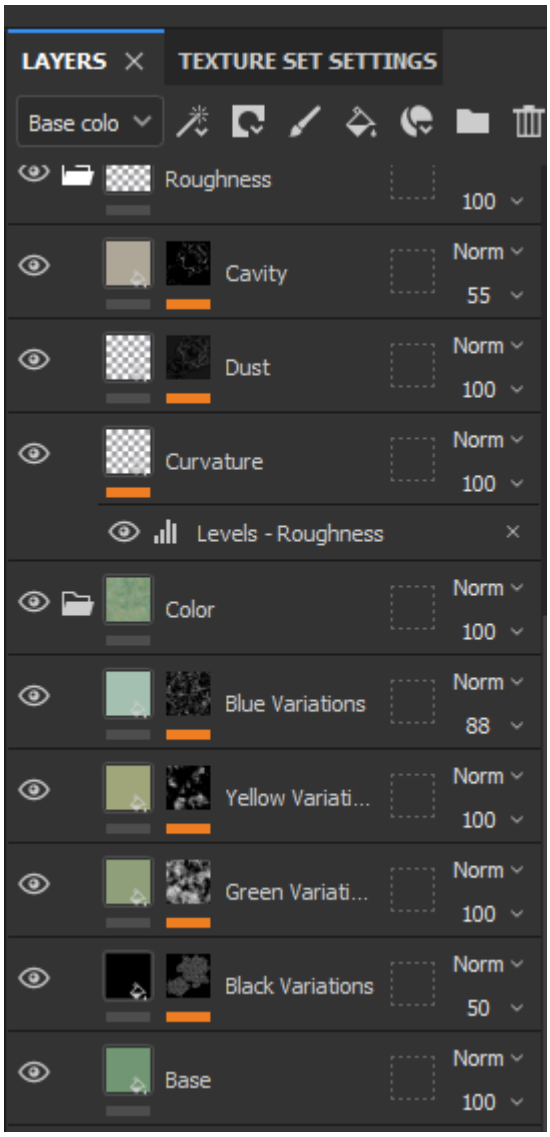


Рисунок 3.43 - Шари в програмі Substance Painter [43]

Кількість каналів для одного пікселя може бути дуже велика. При стандартних налаштуваннях Substance Painter враховує вже в першому шарі 9 каналів - 3 на колір, і 3 для різних ефектів, 3 на карти нормалей. Саме зображення на картинці у вашому браузері містить лише 3 канали + 1 додатковий, якщо є прозорість (рисунок 3.44 [43]).

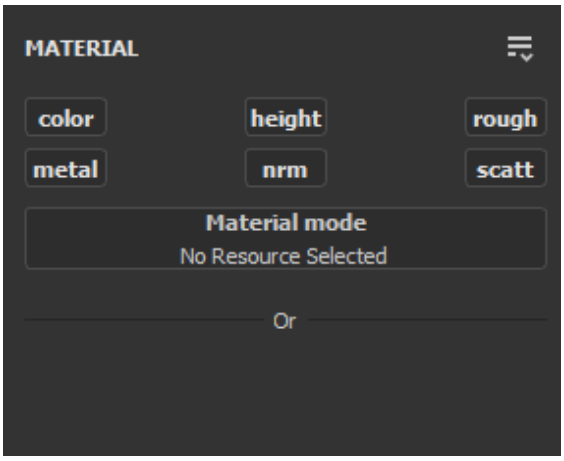


Рисунок 3.44 - Приклад каналів для шарів [43]

Причому, шари можуть один на одного накладатися, і програма вже робить розрахунки на 1 піксель значно більші, ніж для 9 каналів. Тобто, у разі змішування 2 шарів в 1-му пікселі, ПЗ необхідно прорахувати 9 каналів одного шару, потім врахувати прозорість, яка була вказана в каналі маски та прорахувати загальну інтенсивність пікселя з урахуванням маски. Після чого необхідно прорахувати другий шар, та його 9 каналів. Потім накласти зверху напівпрозорий шар. Разом 18 каналів шарів та 1 канал змішування. Звичайно, коли ви вивантажуватимете це у зображення (текстури), то ніяких 19 каналів не буде - Substance Painter створить 3 зображення (текстури):

- Зображення з кольором (3 канали на піксель).
- Зображення з каналами для параметрів PBR.
- Зображення (текстура) визначення нормалей.

Тобто Substance Painter вже зробить всі розрахунки так, щоб можна було отриманий кінцевий результат просто зберегти в зображенні.

3.8.2 Використання текстури

Текстура за фактом - це не зображення у звичному для нас розумінні.

Текстура - це набір параметрів для того, щоб ігрові движки або програми для роботи з текстурами розуміли, як потрібно обробити піксель [44].

Текстура може містити у собі 1 канал. Наприклад, для прозорості. Або може містити в собі відразу 3 канали для різних цілей (наприклад, для параметрів роботи металіка, шорсткості та затінення).

Різновидів застосування текстур дуже багато, але вони зводяться до одного — до роботи з 1-м чи кількома каналами.

Наприклад, текстура Cavity (тріщини) зазвичай описує грані (ребра) об'єкта на розгортці, виділяючи пікселі максимальним білим по всій довжині ребер. Але за фактом це просто текстура, канал якої спрямований на відображення інтенсивності пікселів у потрібних точках. Таку текстуру можна використовувати, наприклад, для масок бруду — тобто по ній ми можемо відображати бруд на об'єкті.

Застосування масок за межами програм для текстурування.

Маски потрібні не тільки для програм типу Substance Painter. У ігрових двигунах текстурування дуже схоже на роботу Substance Painter - ви можете просто встромити готові текстури з Painter, або можете почати налаштовувати шейдери (матеріали) за власним бажанням, комбінуючи текстури через маски різними способами. І якщо Painter дає готовий результат, то шейдер, що базується на масках, робить розрахунки в реальному часі.

3.8.3 Використання PBR-текстурування

Розшифровується ця абревіатура так: Physically Based Rendering. Що в перекладі означає «Заснований на фізиці рендер».

Як ми всі знаємо, ми не могли б бачити жодного об'єкта, якби об'єкти не відображали світло, яке на них падає. Все це працює дуже просто – промінь світла падає на стіл, відбивається і потрапляє вам на сітківку ока.

Залежно від поверхні, від її стану (чи відбиває воно, як дзеркало, чи має шорсткість, чи є краплі бруду/води/кави на поверхні) вам в око приходить вже видозмінений промінь (скажімо так). Причому промінь — це не зовсім правильно, бо вже відомо, що світло — це ще й хвиля. Але це вже дуже глибоко для нас, і тут я можу помилятися [45].

А суть залишається незмінною — те, як ми бачимо об'єкти, вишиковується з того, як світло відбивається від поверхні. І стандарт PBR описує те, які параметри повинні враховуватися, щоб світло виглядало максимально кінематографічно.

Виходячи з того, як працює текстура і для чого вона потрібна, можна дуже швидко пояснити, що таке PBR.

PBR - це заснований на фізиці рендер текстур. Тобто це набір параметрів, керуючи якими ми можемо змусити текстуру виглядати кінематографічно.

Для цього використовуються параметри, які відповідають за те, як світло має відображатися на пікселі.

Ось список цих параметрів:

- Metallic. Відповідає за представлення пікселя як метал. Те, наскільки легко світло відбивається від пікселя.
- Roughness. Відповідає за ступінь шорсткості пікселя. Те, наскільки світло розсіюється, стикаючись з пікселем.
- Ambient Occlusion (застаріле). Відповідає за затінення пікселів. Той параметр, який стає застарілим і незабаром, ним не користуватимуться взагалі.
- Normal Map. Відповідає за те, як світло може спотворюватися на пікселі, створюючи уявну несправжню глибину чи вигин.
- Albedo (Color) — колір, який має відображатися піксель.

Існує ще багато різних параметрів, які можна чи не потрібно враховувати, щоб отримати потрібний результат. Все залежить від того, якого ефекту ви досягаєте.

Наприклад, якщо програмі встромити в канал металіка піксель з єдиним каналом, інтенсивність якого дорівнюватиме 1 (255), як на рисунку 3.45 [45].

Рисунок 3.45 – Плейн, у якого інтенсивність металику дорівнює 1 [45]

То цей піксель максимально відображатиме все навколо, як чистий метал.

На рисунку 3.46 [45] в каналі Roughness налаштовуємо всі пікселі в 1 (255).

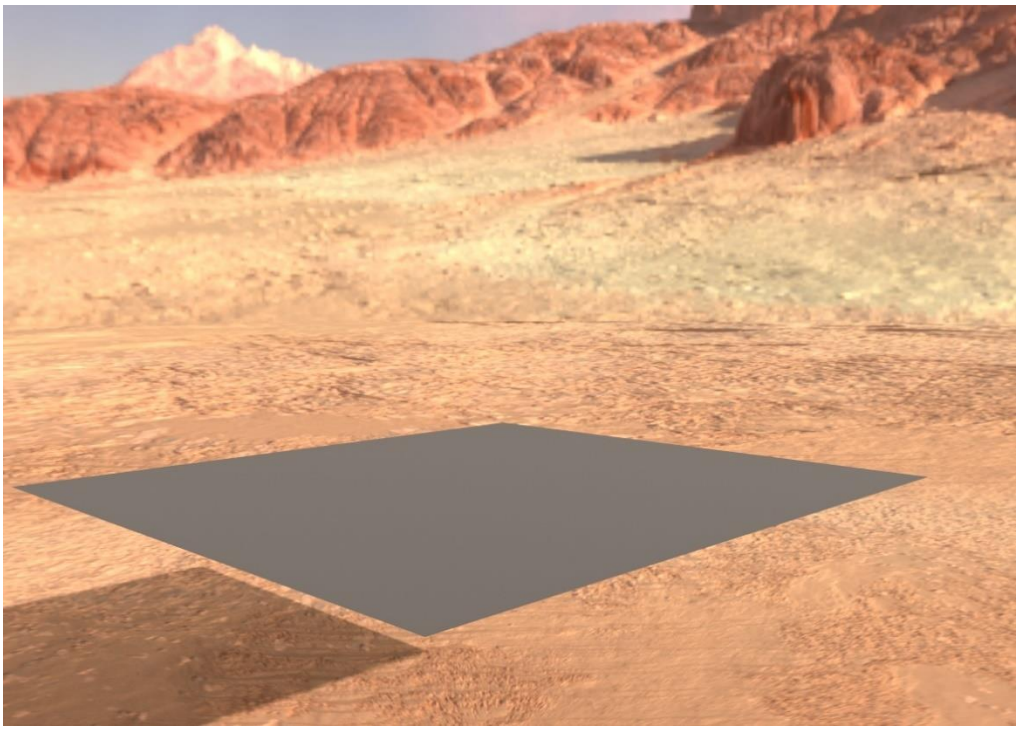


Рисунок 3.46 – Плейн, у якого інтенсивність рафнесу дорівнює 1 [45]

Ми бачимо, що тепер піксель перестав відбивати все навколо і почав розсіювати світло настільки сильно, що став сірим.

Дуже важливо розуміти, що не існує окремих текстур, що спеціалізуються на чомусь одному. Всі текстури — це канали, пікселі яких мають інтенсивність або значення, вказуючи які, ви повідомляєте тому чи іншому параметру реагувати так чи інакше.

В цьому розділі було проведено огляд пайплайну створення 3D-моделей в ігровій індустрії. Пайплайн складається з драфту, пошуку силуету, створення сітки highpoly та lowpoly, розгортки, запікання карт та текстурування. Було проаналізовано, с чого краще починати роботу, а саме з пошуку референсів та аналізу форм моделі, яку необхідно моделювати.

Розглянуті принцип драфту моделей, що в першу чергу краще моделити з простих великих форм, щоб в точності зробити правильний силует моделі, а вже потім робити більшу деталізацію. Описані типи полігональних сіток моделей, а саме: lowpoly - сітка з низьною деталізацією; ,

midpoly – сітка за середнім рівнем деталізацією; highpoly – дуже деталізована сітка.

Проаналізовані правила ретопології highpoly моделі. Отримана після ретопології, lowpoly модель повинна мати низькодеталізовану рівномірну полігональну сітку, переважно с чотирикутників, але дозволяється робити трикутники в місцях, де вони будуть менш всього видні та не будуть ломати форму, а навпаки будуть її підкреслювати без нарощування зайвих полігонів.

Розглянуто для чого необхідна UV-розгортка та як її правильно робити. Проаналізовані типи UV-розгортки: унікальний мапінг – це одна текстура, яка намальована спеціально під створену розгортку; тайловий мапінг – це текстура, яка буде повторюватися нескінченну кількість разів. Також описано процес запікання карт нормілів, яка необхідна для передачі рельєфа з highpoly на lowpoly, та ambient occlusion, яка необхыдна для передачі на моделі тіней.

Проаналізовано процес створення текстур, для цього використовується програма Substance painter, в якій налаштовуються різноманітні параметри PBR текстур, основні з них це: висота, шорсткість, металічність та колір.

4 РОЗРОБКА 3D-МОДЕЛІ ІГРОВОГО ПЕРСОНАЖУ

Для розробки 3D-моделі ігрового персонажу слідуємо за пайплайном з 3 розділу, в першу чергу мы обираемо референс для майбутнього персонажа [46].

Для цієї роботи був обран референс на рисунке 4.1 [46].

Рисунок 4.1 - Референс персонажу [46]

На данному референсі у першу чергу виділимо основні форми, щоб спростити роботу (рисунок 4.2).

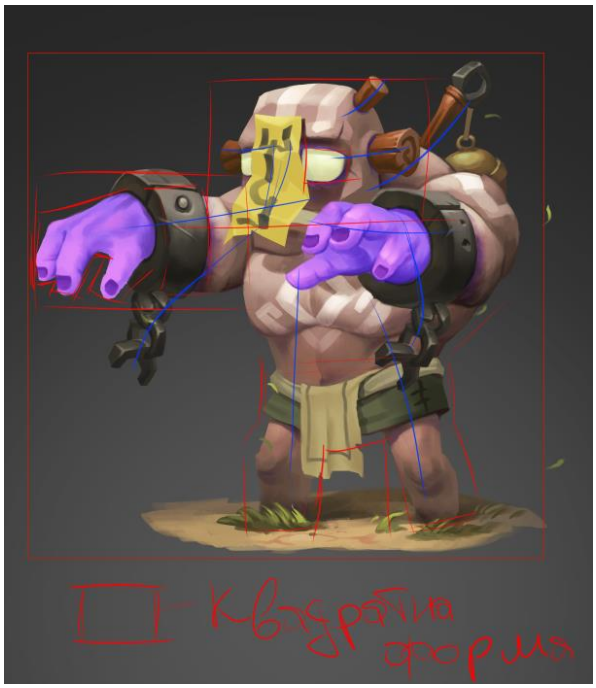


Рисунок 4.2 – Референс з виділиними формами

Як ми бачимо, персонаж має достатнь прямокутні форми. Згідно з психологією прямокутник - це стабільність, сила, залізна воля та завзятість [47]. Багато класичних силачів з мультфільмів і коміксів виглядають, як бетонний блок з ногами.

Після аналізу референса, відкриваємо програму ZBrush та починаємо робити драфт великими формами, як на рисунке 4.3.

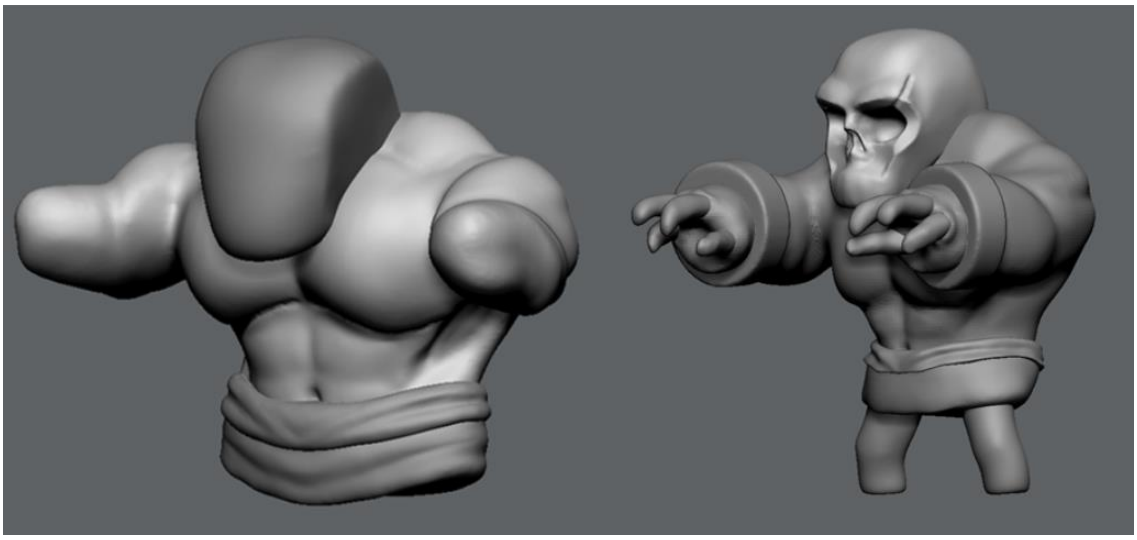


Рисунок 4.3 – Початок етапу блокінгу

В процесі блокінгу йде пошук необхідних форм, багато правок, щоб на етапі деталізації неприходилось правити основні форми. Для більшої зручності та швидкості деякі деталі були зроблі в Мауа полігональним моделінгом, ці елемнти показано на рисунке 4.4.

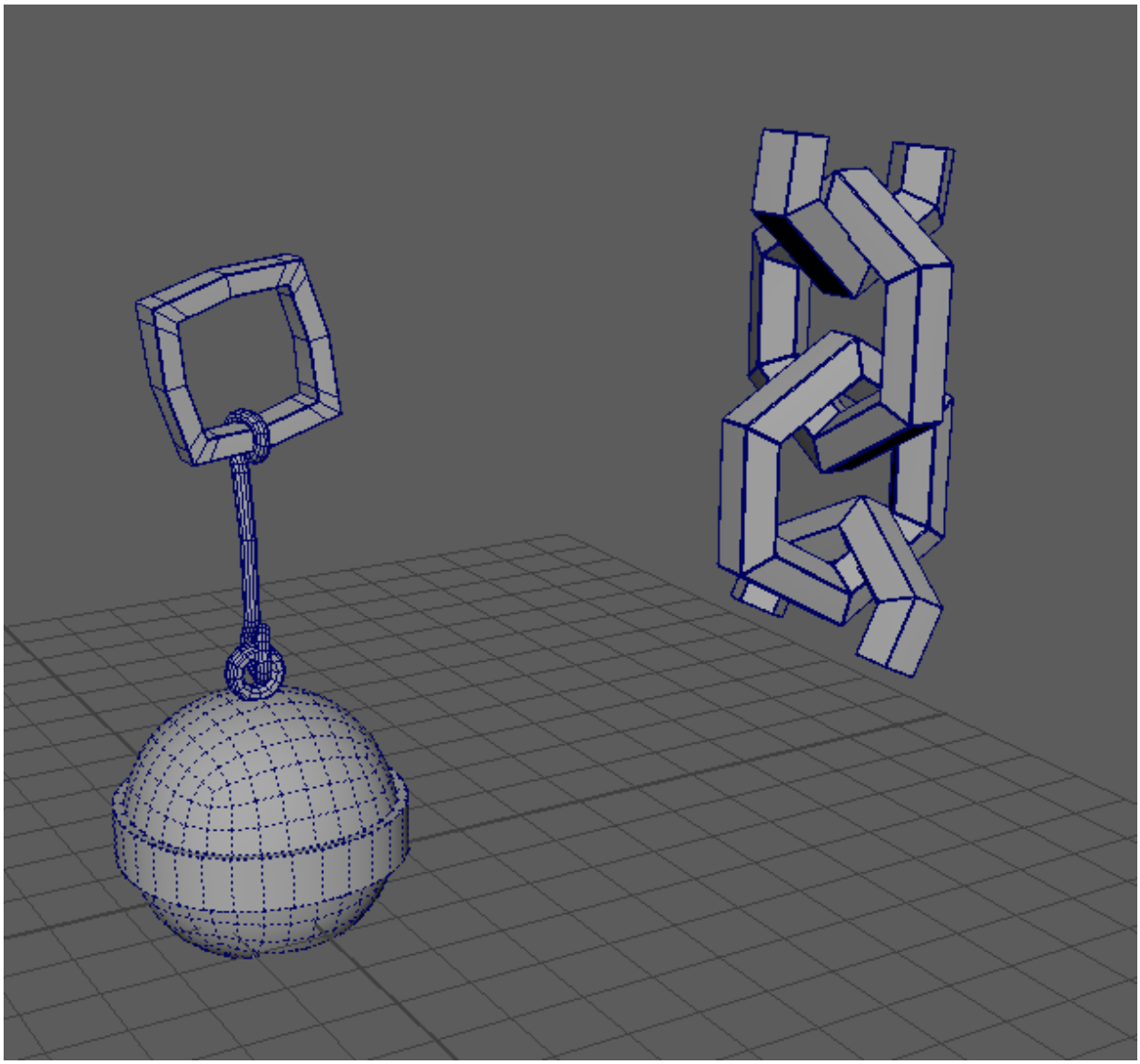


Рисунок 4.4 – Елементи, зроблені в Мауа

Ці елементи добавлені до фінального делізованого драфту на рисунке 4.5, в якому намічені усі основні форми.

Рисунок 4.5 – Фінальний деталізований драфт

Як видно на цьому фінальному драфті персонаж виглядає досить детально, усі форми та об’єкти знаходяться на своїх місцях.

Наступним кроком починаємо деталізацію. Це достатньо довгий этап, необхідно проробити середні та маленькі форми, на загальний силует це не вплине, але персонаж стане цікавіше.

Фінальний результат деталізації зображено на рисунке 4.6.

Рисунок 4.6 – Деталізований персонаж

В процесі деталізації персонажа було прийнято рішення проробити обличчя, яке на референсі прикрите папіром з символом.

Далі по пайплайну йде ретополія і UV-розгортка тому деталізованого персонажа ми експортуемо до софту Maya (рисунок 4.7).

Рисунок 4.7 – Перносаж у вікні програми Maya

Розглянемо наступні важливі проблеми:

1) Перша проблема завеликий poly count, це видно на рисунке 4.8, одим мільйон трикутників це забагато.

Рисунок 4.8 - Кількість трикутників на моделі

2) Друга важлива проблема це неравномірна сітка з трикутників (рисунок 4.9).

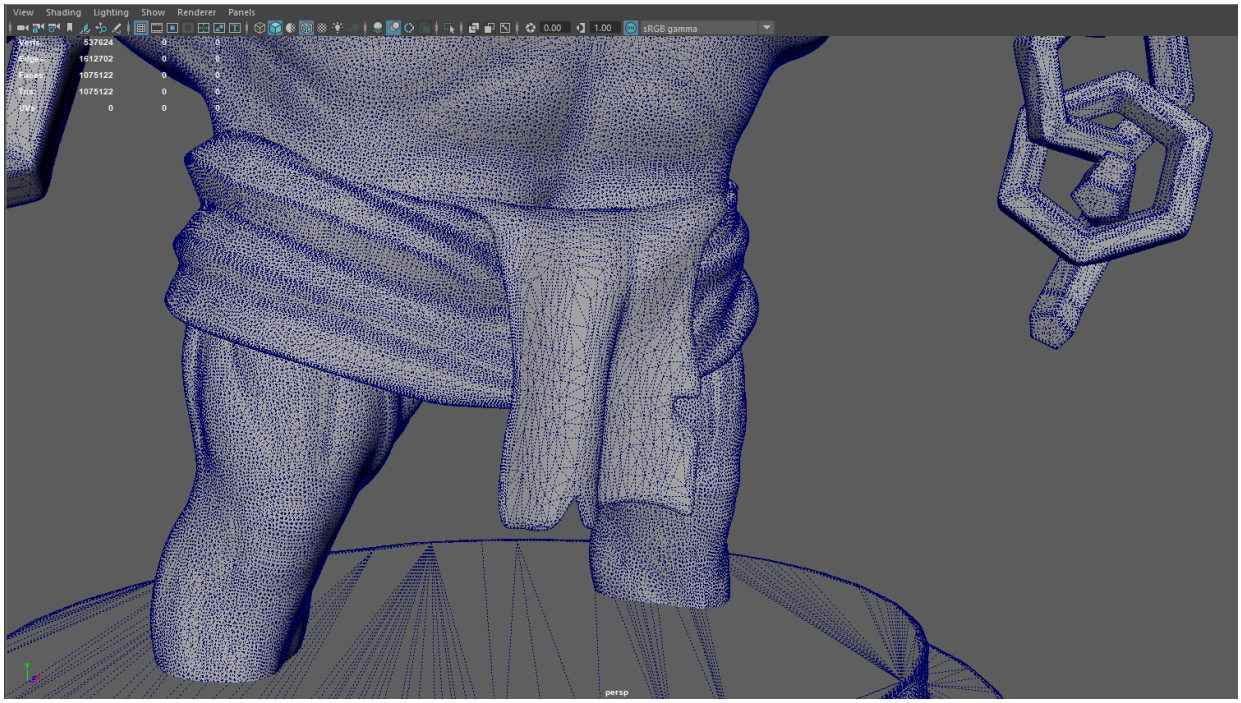


Рисунок 4.9 – Сітка на моделі

Тому необхідно зробити ретопологію, щоб зменшити кількість полігонів, зробити сітку більш рівномірною та отримати lowpoly модель (рисунок 4.10).

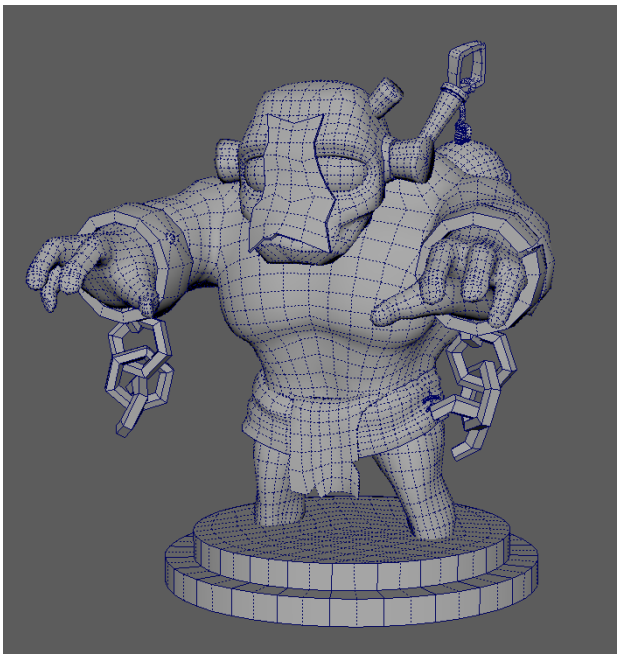


Рисунок 4.10 – Lowpoly модель

В результаті ми отримали модель з рівномірною сіткою всього на 16000 трикутників.

Далі необхідно зробити розгортку(рисунок 4.11). На лоуполі моделі це легка задача, бо вся модель скаладеється з еджлупів.

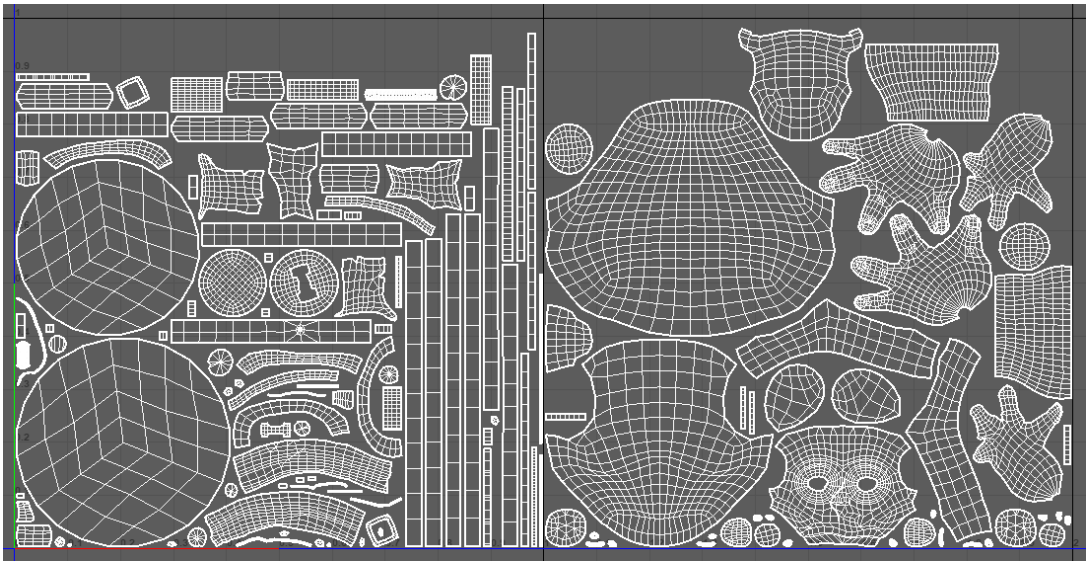


Рисунок 4.11 – UV-розгортка персонажа

Ось так виглядає розгортка цієї моделі. Щоб на этапі текстурування не втратити в якості, розгортка була розділина на два юдіми.

Після того як було зроблена ретопологія та UV-розгортка, наступним этапом необхідно зробити bake, щоб перенести деталі з хайполі моделі на лоуполі.

Для шкидкого та зручного бейку необхідно усі елементи моделі правильно іменувати. Лоуполі повинно бути з припискою _low, а хайполі з припискою _high, як на рисунке 4.12.

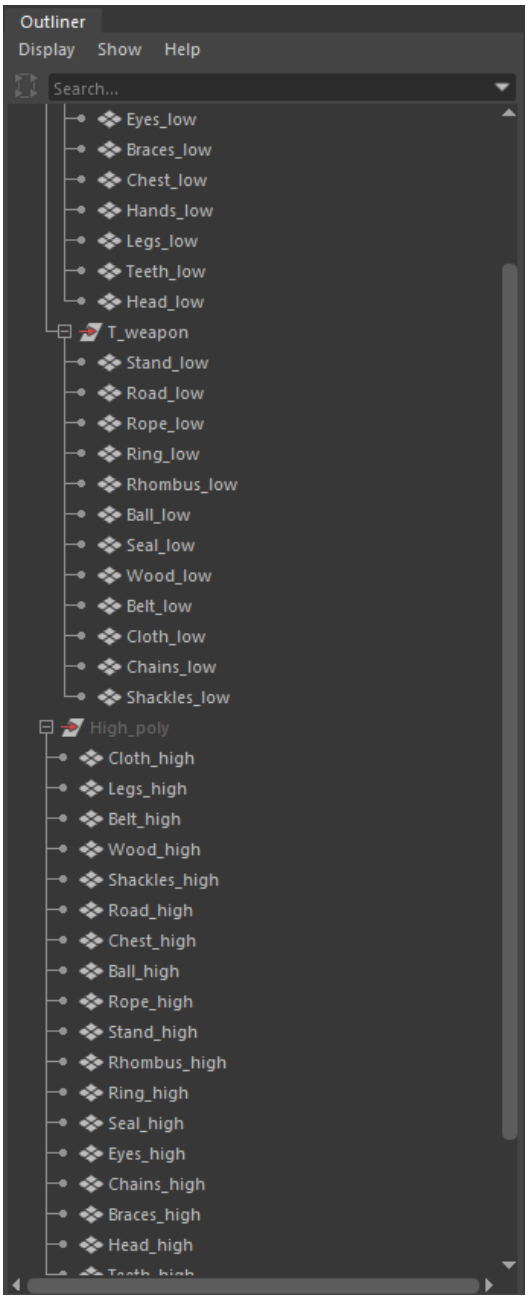


Рисунок 4.12 – Outliner з іменованими мешами

Наступним кроком після іменування необхідно зробити триангуляцію лоуполі моделі на рисунке 4.13.

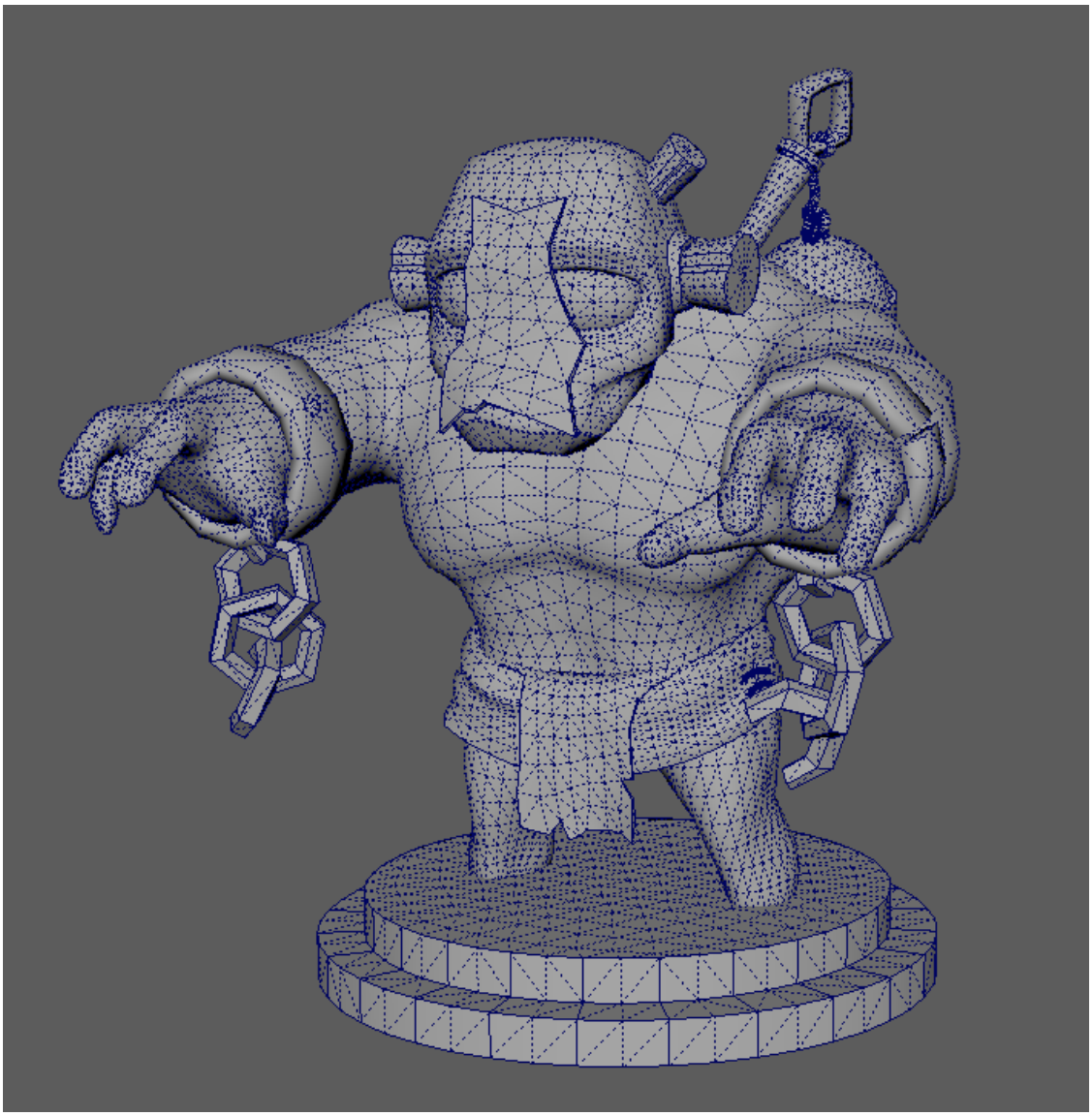


Рисунок 4.13 – Триангульована лоуполі

Триангуляцію робити необов’язково, але для кращого результату та появи меньшої кількость артефакті бажано триангулювати.

Після цих швидких маніпуляцій експортуемо лоуполі і хайполі в одному файлі до програми Marmoset Toolbag (рисунок 4.14).

Рисунок 4.14 – Вікно програми Marmoset Toolbag

Завдяки правильному наіменуванні усі меші моделі згрупувалися у папки по назвам. Все що тепер необхідно, це обрати папку куди буде збережен бейк, увімкнути необхідні карти, задати розмір текстур (в данному випадку 4К) і натиснути кнопку “Bake”. Після запікання вся деталізація з хайполі перейде на лоуполі у вигляді карти нормалі та ambient occlusion, як на рисунке 4.15.

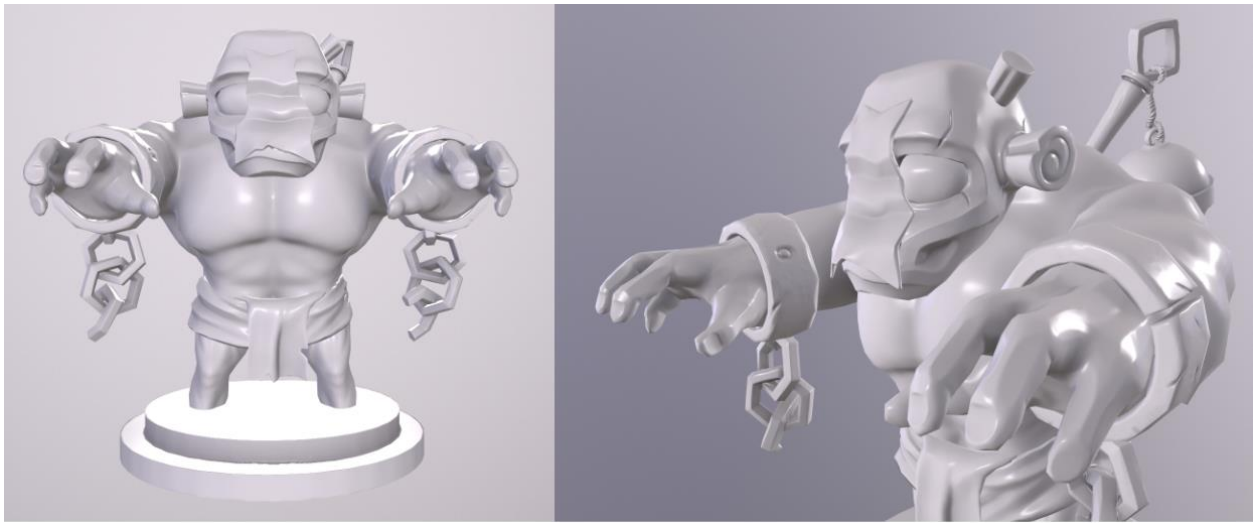


Рисунок 4.15 – Лоу полі після бейку

Як видно на цьому рисунке, у лоуполі з’явилась деталізація з хайполі та контактні тіні.

І останнім етапом отримані мапи з бейку та лоуполі модель необхідно експортувати до програми Substance Painter, де буде проходити текстурування моделі (рисунок 4.16).

Рисунок 4.16 – Модель у вікні Substance Painter

З допомогою різноманітних масок, текстур та хендпеінту робимо матеріали для персонажа. Результат текстурування на рисунке 4.17.

Рисунок 4.17 – Фінальна модель персонажа

Після тестурування модель надходить до ігрового двигуна, наприклад, Unity або Unreal Engine.

В цьому розділі була розроблена 3D-модель персонажу за пайплайном, який описан в 3 розділі, а саме: був підібран референс, визначено основні форми силуета, зроблена блокінг та детальний драфт моделі, після модель була продеталізована та заретоплена, зроблена UV-розгортка та запечені карти нормалів, і контактного затемнення, останнім кроком модель була затекстурована.

ВИСНОВОК

В роботі були розглянуті основні принципи та методи створення 3D моделей, а саме: для створення тривімірного об’єкту необхідно створити модель цього об’єкта у просторових координатах X, Y, Z, тобто описати кожну точку на поверхні цього – вказати його координати; складові моделювання сітки такі, як вершина, ребро, грань та полігон. Також було проаналізовано предствалення полігональних сіток: вершинне уявленя сіток, де об’єкт описується як безліч вершин, уявлення списку граней, де об’єкт представляється як безліч граней та безліч вершин, «крилате» уявлення, явно представляться верниши, грані та ребра сітки. Ще були проаналізовані методи для оптимізації сіток 3D моделей, щоб GPU швидшке та менш трудомістко рендерив полігони, і було виявлено, що на швидкість рендеру впливає не тількі кількість полігонів, а ще й форма цих полігонів, розположення різних об’єктів відносно один до одного та потрапляння певних полігонів у камеру рендера.

Також був проведен огляд видів полігоновів та полігональних сіток. Полігони існують із трьома, чотирма і п’ятьма вершина. Правильна полігональна сітка повинна бути плавною та мати потокову організованість полігонів. Також проаналізовано методи полігнонального моделювання такі, як: твердотільне моделювання, полігональне моделювання, hard surface, сплайн моделювання, nurbs моделювання, поверхневе моделювання та скульптинг. Для кваліфікаційної роботи було обрані методи скульптинга та полігонального моделювання. Завдяки скульптингу в програмі ZBrush було отримано модель персонажа, а полігональним моделюванням в MAYA була зроблена ретопологія.

Ще в роботі був проведен огляд пайплайну створення 3D-моделей в ігровій індустрії. Пайплайн складається з драфту, пошуку силуету, створення сітки highpoly та lowpoly, розгортки, запікання карт та текстурування. Було

проаналізовано, с чого краще починати роботу, а саме з пошуку референсів та аналізу форм моделі, яку необхідно моделювати.

Розглянут принцип драфту моделей, що в першу чергу краще моделити з простих великих форм, щоб в точності зробити правильний силует моделі, а вже потім робити більшу деталізацію. Описані типи полігональних сіток моделей, а саме: lowpoly - сітка з низьною деталізацією; , midpoly – сітка за середнім рівнем деталізацією; highpoly – дуже деталізована сітка.

Проаналізовані правила ретопології highpoly моделі. Отримана після ретопології, lowpoly модель повинна мати низькодеталізовану рівномірну полігональну сітку, переважно с чотирикутників, але дозволяється робити трикутники в місцях, де вони будуть менш всього видні та не будуть ломати форму, а навпаки будуть її підкреслювати без нарощування зайвих полігонів.

Розглянуто для чого необхідна UV-розгортка та як її правильно робити. Проаналізовані типи UV-розгортки: унікальний мапінг – це одна текстура, яка намальована спеціально під створену розгортку; тайловий мапінг – це текстура, яка буде повторюватися нескінченну кількість разів. Також описано процес запікання карт нормілів, яка необхідна для передачі рельєфа з highpoly на lowpoly, та ambient occlusion, яка необхыдна для передачі на моделі тіней.

Описан процес створення текстур, для цього використовується програма Substance painter, в якій налаштовуються різноманітні параметри PBR текстур, основні з них це: висота, шорсткість, металічність та колір.

Та наприкінці була розроблена 3D-модель персонажу за пайплайном, а саме: був підібран референс, визначено основні форми силуета, зроблена блокінг та детальний драфт моделі, після модель була продеталізована та заретоплена, зроблена UV-розгортка та запечені карти нормалів, і контактного затемнення, останнім кроком модель була затекстурована.

Обрані для роботи методи моделювання показали себе ефективними у поставленій задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. MQL5 Як створити 3D-графику на DirectX в MetaTrader 5. [Електронний ресурс] URL: https://www.mql5.com/articles/7708 (14.10.22)
2. Полігональна сітка. [Електронний ресурс] URL: http://surl.li/dydor (15.10. 22)
3. Colin Smith, On Vertex-Vertex Meshes and Their Use in Geometric and Biological Modeling. [Електронний ресурс] URL: http://algorithmicbotany.org/papers/smithco.dis2006.pdf (15.10. 22)
4. Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference (15.10.22)
5. Оптимізація 3D моделей як спосіб зменшення навантаження на графічний процесор. [Електронний ресурс] URL: https://openarchive.nure.ua/bitstream/document/16776/1/KovalevaDei.pdf (16.10. 22)
6. Ігрова індустрія зростає за рахунок мобільних ігор. [Електронний ресурс] URL: http://surl.li/eanlm (16.10.22)
7. Видалити те, що приховано: оптимізація 3D сцен у мобільній грі. Поради працівників Plarium. [Електронний ресурс] URL: https://habr.com/company/plarium/blog/348494/ (16.10.22)
8. Оптимізація 3D моделей для ігрової сцени. [Електронний ресурс] URL: https://habr.com/company/plarium/blog/484792/ (17.10.22)
9. ХАБР Для оптимізації 3D-моделей недостатньо вважати полігони. [Електронний ресурс] URL: https://habr.com/post/433186/ (16.10.22)
10. Хабр Параметри вершин. [Електронний ресурс] URL: http://surl.li/dymxh (16.11.22)
11. Хабр Використання викликів малювання. [Електронний ресурс] URL: https://cutt.ly/T13iTXm (17.11.22)

12. I. Y. A. Corpus, L.Lindner, O.Sergiyenko. "Transimpedance Amplifier for Laser Scanning System Range Extension," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1421-1426, doi: 10.1109/ISIE45063.2020.9152487. URL: https://ieeexplore.ieee.org/abstract/document/9152487 (16.10. 22)
13. Ivanov, M., Sergiyenko, O., Mercorelli, P., Hernandez, W.c, Rodriguez Quinonez, J.C.d, Katashov V., Kolendovska, M., Iryna, T. Effective informational entropy reduction in multi-robot systems based on real-time TVS. IEEE International Symposium on Industrial Electronics, 2019-June,8781209, c. 1162-1167. (16.10. 22)
14. Jonathan J. Sanchez-Castro ; Julio C. Rodríguez-Quiñonez ; Luis R. Ramírez-Hernández ; Guillermo Galaviz ; Daniel Hernández-Balbuena ; Gabriel Trujillo-Hernández ; Wendy Flores-Fuentes ; Paolo Mercorelli ; Wilmar Hernández-Perdomo ; Oleg Sergiyenko ; Félix Fernando González-Navarro. "A Lean Convolutional Neural Network for Vehicle Classification," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1365-1369, doi: 10.1109/ISIE45063.2020.9152274. URL: https://ieeexplore.ieee.org/abstract/document/9152274 (16.10. 22)
15. Lindner, L., Sergiyenko, O., Rivas-López, M., (...), Gurko, A., Kartashov, V.M. Machine vision system for UAV navigation; IEEE, 2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles and International Transportation Electrification Conference, ESARS-ITEC, 2016; pp.1–6. DOI: 10.1109/ESARS-ITEC.2016.7841356. (16.10. 22)
16. M. Ivanov, O. Sergiyenko, V. Tyrsa, P. Mercorelli, V. Kartashov, W. Hernandez, S. Sheiko, M. Kolendovska. Individual scans fusion in virtual knowledge base for navigation of mobile robotic group with 3D TVS // Proceedings of 44th Annual Conference of IEEE Industrial Electronics

Society (IECON).. -2018. – Washington DC, USA. -S. 3187-3192. . ISBN 978-1-5090-6683-4/18/. (16.10. 22)

17. Murrieta-Rico, F.N., Petranovskii, V., Galvan, D.H., Sergiyenko, O., Yocupicio-Gaxiola, R.I., De Dios Sanchez-Lopez, J. Phase effect in frequency measurements of a quartz crystal using the pulse coincidence principle. 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 185-190, 9152255, DOI: 10.1109/ISIE45063.2020.9152255 (16.10. 22)
18. Oleksandr Sotnikov, Vladimir Kartashov, Oleksandr Tymochko, Oleg Sergiyenko, Vera Tyrsa, Paolo Mercorelli, Wendy Flores-Fuentes. Methods for Ensuring the Accuracy of Radiometric and Optoelectronic Navigation Systems of Flying Robots in a Developed Infrastructure. Chapter 16// Machine Vision and Navigation; Springer, Cham. pp.537–578. Editors: Sergiyenko, Oleg, Flores-Fuentes, Wendy, Mercorelli, Paolo. DOI: 10.1007/978-3-030-22587-2_16. (16.10. 22)
19. Optical detection of unmanned air vehicles on a video stream in a real-time/Kartashov, V., Oleynikov, V., Zubkov, O., Sheiko, S.// 2019 International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019 - Proceedings, 2019, 9165362/ (16.10. 22)
20. Principles Of Construction And Assessment Of Technical Characteristics Of Multi-Frequency Atmospheric Sodar In The Humidity Measurement Mode / Kartashov, V.M., Sidorov, G.I., Sheiko, S.A., Kolendovskaya, M.M., Sergienko, O.Yu. // Telecommunications And Radio Engineering (English Translation Of Elektrosvyaz And Radiotekhnika), 2020, ISSN Print: 0040-2508, ISSN Online: 1943-6009, DOI: 10.1615/TelecomRadEng.v79.i4.50, p. 323-333/ (16.10. 22)
21. Research Of The Uncertainty Of Measurement Frequencies And Definitions Of The Frequency Signal In The Waveguide With Respect To Power / Semenets, V.Zakharov, I. Serhiienko, M., Kartashov, V.M, , Kolendovska,

M., Hernandez, W., Hipolito, J.I.N., , Tyrsa, V.// 45th Annual Conference of the IEEE Industrial Electronics Society, IECON 2019; Lisbon Congress CenterLisbon; Portugal; 14 October 2019 до 17 October 2019; CFP19IEC-ART; Код 155980, Volume 2019-October, October 2019, № 8927203, Pages 4674-4679 (16.10. 22)

22. Spatial-Temporal Processing Of Acoustic Signals Of Unmanned Aerial Vehicles /Kartashov V.M., Oleinikov V.N., Zubkov O.V., Sheiko S.A., Kolendovska M.M.// Telecommunications And Radio Engineering (English Translation Of Elektrosvyaz And Radiotekhnika), 2020, ISSN Print: 0040-2508, ISSN Online: 1943-6009, DOI: 10.1615/Telecomradeng.v79.i9.40, p. 769-780 (16.10. 22)
23. Stereoscopic Vision Systems In Machine Vision, Models, And Applications (Book Chapter)/ Ramírez-Hernández, L.R., Rodríguez-Quiñonez, J.C., Castro-Toscano, M.J., Kolendovska, M., Murrieta-Rico, F.N.// Machine Vision And Navigation, 2019 Machine Vision and Navigation30 September 2019, Pages 241-265 (16.10. 22)
24. StrelkovaT., KartashovV., Lytyuga A., Strelkov A. Theoretical Methods of Images Processing in Optoelectronic Systems. Chapter 16. // Biometrics: Concepts, Methodologies, Tools, and Applications; Oleg Sergiyenko and Julio C. Rodriguez-Quiñonez. (341p.), IGI Global, 2017; pp. 361-381. DOI: 10.4018/978-1-5225-0983-7.ch016. (16.10. 22)
25. StrelkovaT., KartashovV., Lytyuga A.,StrelkovA. Theoretical Methods of Images Processing in Optoelectronic Systems. Chapter 6// Developing and Applying Optoelectronics in Machine Vision; Oleg Sergiyenko and Julio C. Rodriguez-Quiñonez. (341p.) – USA, Herhey, IGI Global, 2016; pp.180-205. (16.10. 22)
26. Sytnik O., KartashovV. Methods and Algorithms for Technical Visionin Radar Introscopy. Chapter 13// Optoelectronics in Machine Vision-Based Theories and Applications. IGI Global, 2019; pp. 373-391. (16.10. 22)

27. The Use of Factorization and Multimode Parametric Spectra in Estimating Frequency and Spectral Parameters of Signal/Semenets, V., Kartashov, V., Sergiyenko, O., ...Rodriguez-Quinonez, J.C., Flores-Fuentes, W.//IEEE International Symposium on Industrial Electronics, 2020, 2020-June, p. 215-219 (16.10. 22)
28. Unda, O.F., Hernandez, W., Vargas, O., Mendez, A., Sergiyenko, O., Tyrsa, V. Construction of a robotic platform of differential type for first-year students of electronic engineering, 2020 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, SPEEDAM 2020, 24-26 de junio de 2020, Sorrento, Italia, pp. 538-543, 9161870, DOI: 10.1109/SPEEDAM48782.2020.9161870 (16.10. 22)
29. Основні засади створення 3D-моделей. Поняття та методи оптимізації у тривимірній графіці / А. А. Сивожелезова. [Електронний ресурс] URL: https://cutt.ly/k0fXo2s (29.09.22)
30. Джамбруно М. Тривимірна (3D) графіка та анімація / М. Джамбруно. - М.: Вільямс, 2003. - 640 с. (29.09.22)
31. Топологія, ретопологія, міш, сітка / Збірник статей 3Dyuriki. [Електронний ресурс] URL: http://3Dyuriki.com/2015/03/07/topologiya-retopologiya-mesh-setka-3D-slovarspravochnik (29.09.22)
32. Методи оптимізації високополігональних 3D-моделей. [Електронний ресурс] URL: https://cutt.ly/P0fX29p (29.09.22)
33. Ципцин С. Розуміючи MAYA / С. Ципцин. – К.: Самвиздай, 2012. – 700 с. (29.09.22)
34. QVARTA Детально про 3D-моделювання. [Електронний ресурс] URL: https://qvarta.com/blog/podrobno-o-3D-modelirovanii (30.09.22)
35. DTF Як робляться моделі для AAA-ігор - повний гайд по AAA-пайплайну. [Електронний ресурс] URL: http://surl.li/eanlt (18.10.22)
36. DTF AAA-Пайплайн. Все про драфти. [Електронний ресурс] URL: http://surl.li/eanlv (20.10.2022)

37. DTF Про сітку. Lowpoly, Highpoly і вертекс нормалі. [Електронний ресурс]
URL: https://cutt.ly/l0fBWyZ (21.10.22)
38. DTF Ліміт полігонів на лоуполі. [Електронний ресурс]
URL: https://cutt.us/B8YDU (21.11.22)
39. DTF UV Розгортка. [Електронний ресурс]
URL: http://surl.li/eanme (23.10.22)
40. DTF Розгортка. [Електронний ресурс] URL: http://surl.li/dymxj (20.11.22)
41. DTF Все про Bake. [Електронний ресурс] URL: http://surl.li/eanmh (25.10.22)
42. ХАБР Моделі, нормалі и розгортка. [Електронний ресурс]
URL: https://habr.com/post/458988/ (22.11.22)
43. ХАБР Текстурування, або що потрібно знати, щоб стати Художником на поверхні. Піксель. [Електронний ресурс]
URL: https://habr.com/post/458342/ (28.10.22)
44. ХАБР Маски та текстури. [Електронний ресурс]
URL: https://habr.com/post/458470/ (29.10.22)
45. ХАБР PBR и матеріали. [Електронний ресурс]
URL: https://habr.com/post/458696/ (29.10.22)
46. Референс. [Електронний ресурс]
URL: https://www.artstation.com/artwork/3BBm2 (02.11.22)
47. DTF Дизайн персонажа. Від простого до хитромудрого. [Електронний ресурс] URL: http://surl.li/dtbii (01.11.22)