ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтерактивна розвивальна гра на основі технологій Unreal Engine»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр Дячук
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Олександр Дячук
група КНДМ-61

Керівник: Сергій ІЩЕРЯКОВ
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дячуку Олександру Вадимовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтерактивна розвивальна гра на основі технологій Unreal Engine»

керівник кваліфікаційної роботи Сергій ІЩЕРЯКОВ к.т.н., доцент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з питань, що пов'язані з розробкою інноваційних, інтерактивних, розвивальних ігор..

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1. Концепція розробки гри.
   4.2. Прототипування гри.
   4.3. Інші етапи розробки гри.
   4.4 Створення ігрового контенту.
5. Перелік графічного матеріалу: *презентація*

5.1. Правила, закони та приклади основних принципів розробки гри.
5.2. Приклади стилістик ігрових інтерфейсів та 3D моделей.
5.3. Приклади об’єктів та рівнів у Unreal Engine 5.
5.4. Результат створення розвивальної гри.

6. Дата видачі завдання«16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.24	Виконано
2	Визначення базових понять гри	30.09-11.10.24	Виконано
3	Розробка концепції гри	14.10-25.10.24	Виконано
4	Прототипування гри	28.10-08.11.24	Виконано
5	Створення інтерфейсу та 3D моделей	11.11-22.12.24	Виконано
6	Створення проекту у Unreal Engie	25.11-29.11.24	Виконано
7	Оформлення роботи: вступ, висновки, реферат	02.12-06.12.24	Виконано
8	Розробка демонстраційних матеріалів	09.12-13.12.24	Виконано
9	Попередній захист роботи		

Здобувач вищої освіти _____ Олександр ДЯЧУК
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій ІЩЕРЯКОВ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 87 стор.,135 рис., 15 джерел.

Наукове завдання – дослідження принципів розробки інтерактивних, розвивальних ігор для дітей з використанням сучасних технологій, таких як Unreal Engine, Blender, та Adobe Photoshop.

Мета роботи – розробити інтерактивну гру, яка поєднує навчальний та ігровий процеси, сприяючи розвитку критичного мислення, логіки та уяви у дітей.

Об'єкт дослідження – інтерактивна, розвивальна гра.

Предмет дослідження – процес створення та реалізації інтерактивної, розвивальної гри з акцентом на освітню та розвивальну складову.

Методи дослідження – аналіз функціоналу платформ розробки, використання інструментів Unreal Engine, Blender, Adobe Photoshop, а також методів візуального та інтерактивного дизайну.

Короткий зміст роботи: У роботі досліджено основні принципи створення інтерактивних ігор, які сприяють розвитку дітей. Розроблено прототип гри з головоломками, яка стимулює навчання через ігрову взаємодію. Основні етапи роботи включали створення концепції гри, розробку ігрового середовища за допомогою Unreal Engine, моделювання об'єктів у Blender, розробку текстур і графічних елементів у Adobe Photoshop, а також програмування ігрових механік та інтерактивного контенту. Завдяки сучасним технологіям забезпечено високу якість графіки, доступний інтерфейс та інтерактивність. Гра орієнтована на розвиток когнітивних здібностей, моторику та логіку у дітей через вирішення цікавих головоломок у навчально-ігровому середовищі.

Ключові слова: ноди, Blueprint, скелет моделі (ригінг), скульптинг, сітка, ретопологія, текстура, UV розгортка, UI - інтерфейс користувача (User Interface), UX - досвід користувача (User Experience), зручність використання (Usability), візуальна бібліотека, референс, вайрфрейм, скетч, полішинг, мудборд (Moodboard), User Flow (візуальне уявлення про шлях користувача).

ABSTRACT

The text part of the qualification work for obtaining the master's degree: 87 pages, 135 pictures, 15 sources.

The scientific task – research of principles of development of interactive, educational games for children using modern technologies, such as Unreal Engine, Blender, and Adobe Photoshop.

The purpose of the work is to develop an interactive game that combines educational and gaming processes, promoting the development of critical thinking, logic, and imagination in children.

The object of research is an interactive, educational game.

The subject of research is the process of creating and implementing an interactive, educational game with an emphasis on the educational and developmental component.

Research methods – analysis of the functionality of development platforms, use of Unreal Engine, Blender, Adobe Photoshop tools, as well as methods of visual and interactive design.

Summary of the work: The work investigates the basic principles of creating interactive games that promote children's development. A prototype of a puzzle game that stimulates learning through game interaction has been developed. The main stages of the work included creating a game concept, developing a game environment using Unreal Engine, modeling objects in Blender, developing textures and graphic elements in Adobe Photoshop, as well as programming game mechanics and interactive content. Thanks to modern technologies, high-quality graphics, an accessible interface and interactivity are ensured. The game is focused on developing cognitive abilities, motor skills and logic in children through solving interesting puzzles in an educational and gaming environment.

Keywords: nodes, Blueprint, model skeleton (rigging), sculpting, mesh, retopology, texture, UV scan, UI - user interface, UX - user experience, usability, visual library, reference, wireframe, sketch, polishing, moodboard, User Flow (visual representation of the user journey).

ЗМІСТ

ВСТУП

Тема кваліфікаційної роботи "Дослідження інтерактивної, розвивальної гри на основі технологій Unreal Engine" обрана через актуальність вивчення технологічних інновацій у сфері освіти. Сучасні освітні тенденції спрямовані на впровадження інтерактивних підходів, які поєднують розваги з навчанням, сприяючи залученню та розвитку користувачів.

Дослідження зосереджене на створенні гри для дітей, орієнтованої на розвиток логічного мислення та когнітивних навичок через головоломки. Ця гра є прикладом того, як цифрові технології можуть бути інтегровані в освітній процес, стимулюючи дітей до навчання в ігровій формі.

Розробка ігор є складним процесом, який потребує співпраці спеціалістів різних напрямів:

- Гейм-дизайнер: відповідає за створення концепції гри, механік і правил. У нашому випадку це створення головоломок, що розвивають когнітивні здібності.
- Дизайнер інтерфейсу: забезпечує інтуїтивний і зручний користувацький інтерфейс, який сприяє легкому освоєнню гри.
- 3D-художник: створює моделі, текстури та інші графічні ресурси для гри.
- Програміст: займається реалізацією ігрової логіки, написанням коду та інтеграцією всіх компонентів у рушій.
- Тестувальник: забезпечує перевірку якості гри, знаходить помилки та оптимізує її роботу.

Розробка інтерактивної гри включала три ключові фази:

1) Pre-production:
   a) Концепція:
      - Визначено освітню мету гри — розвиток у дітей логічного мислення через головоломки.
      - Створено початковий сценарій і основні механіки гри.
   b) Прототипування:

- Розроблено базові механіки для тестування ідей.
- Побудовано простий прототип для перевірки взаємодії гравців з грою.

c) Вертикальний зріз:

- Створено перший повноцінний ігровий рівень, який включає всі основні функції гри.

2) Production:

a) Виробництво контенту:

- Blender: створення тривимірних моделей персонажів, об'єктів та ігрового середовища.
- Adobe Photoshop: розробка користувацького інтерфейсу (UI/UX), який забезпечує легкість та зручність навігації.
- Unreal Engine 5: створення ігрового середовища, інтеграція контенту та написання коду.

b) Тестування:

- Перевірка функціональності та стабільності гри.
- Збір зворотного зв'язку від тестувальників.

c) Підготовка до релізу:

- Оптимізація продуктивності та усунення виявлених недоліків.

3) Post-production:

a) Оперування:

- Випуск гри та подальша її підтримка.
- Моніторинг відгуків гравців і внесення змін для покращення ігрового досвіду.

Унікальність роботи полягає у впровадженні ігрових технологій для вирішення освітніх завдань. Поєднання навчання та ігрового процесу сприяє кращому сприйняттю інформації дітьми. Технологічні інновації, такі як Unreal Engine 5, дозволяють створювати інтерактивні середовища, які є захоплюючими та ефективними для навчання.

1 КОНЦЕПЦІЯ

1.1 Перший етап розробки

На першому етапі команда на чолі з гейм-дизайнером збирається разом та проводить брейнштorм.

Розробники вигадують гру, спочатку позначають жанр. Він стане головним вектором у розробці (рис.1.1).

Рисунок 1.1 – Основні пункти концепції гри

Вибирається сеттінг гри. Це коли і де відбуватимуться дії (рис.1.2).

Обговорюється художній напрямок гри, її стиль графіки (рис.1.3).

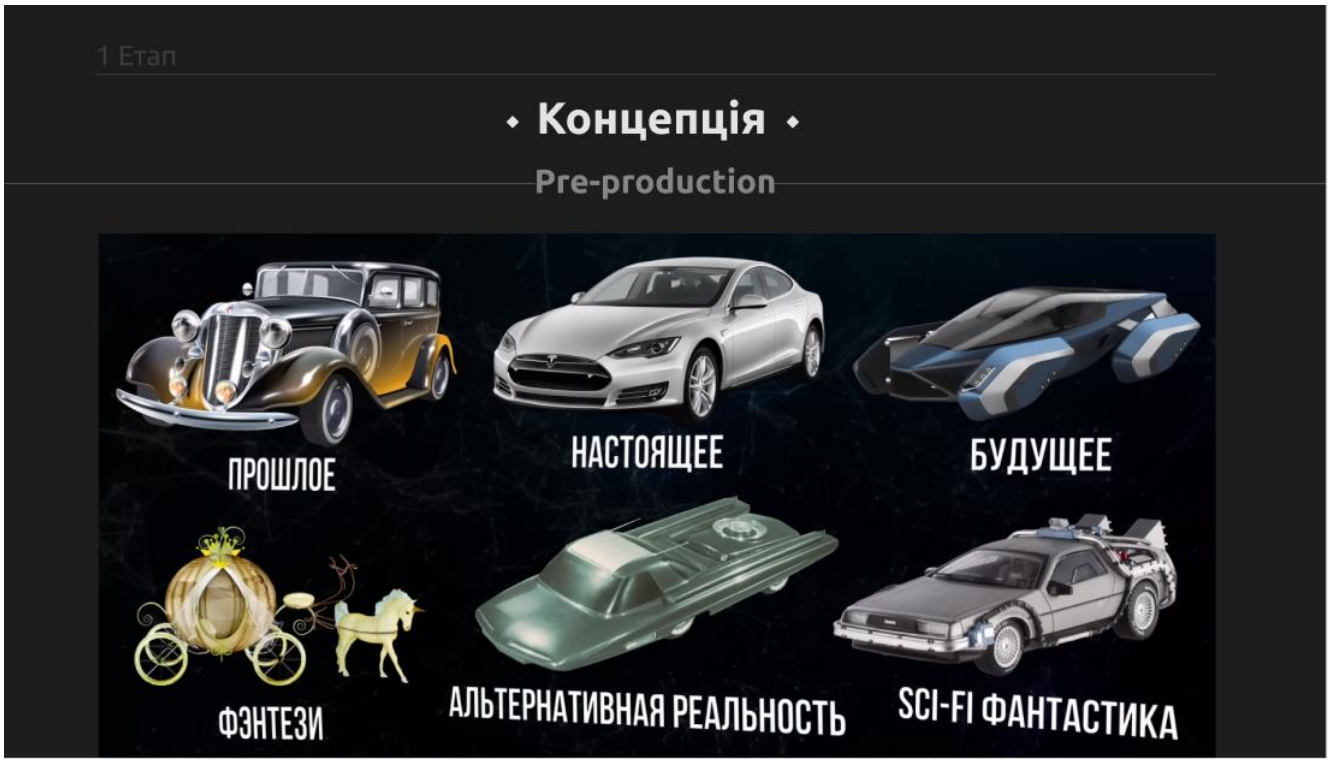


Рисунок 1.2 – Приклади сеттінгів гри

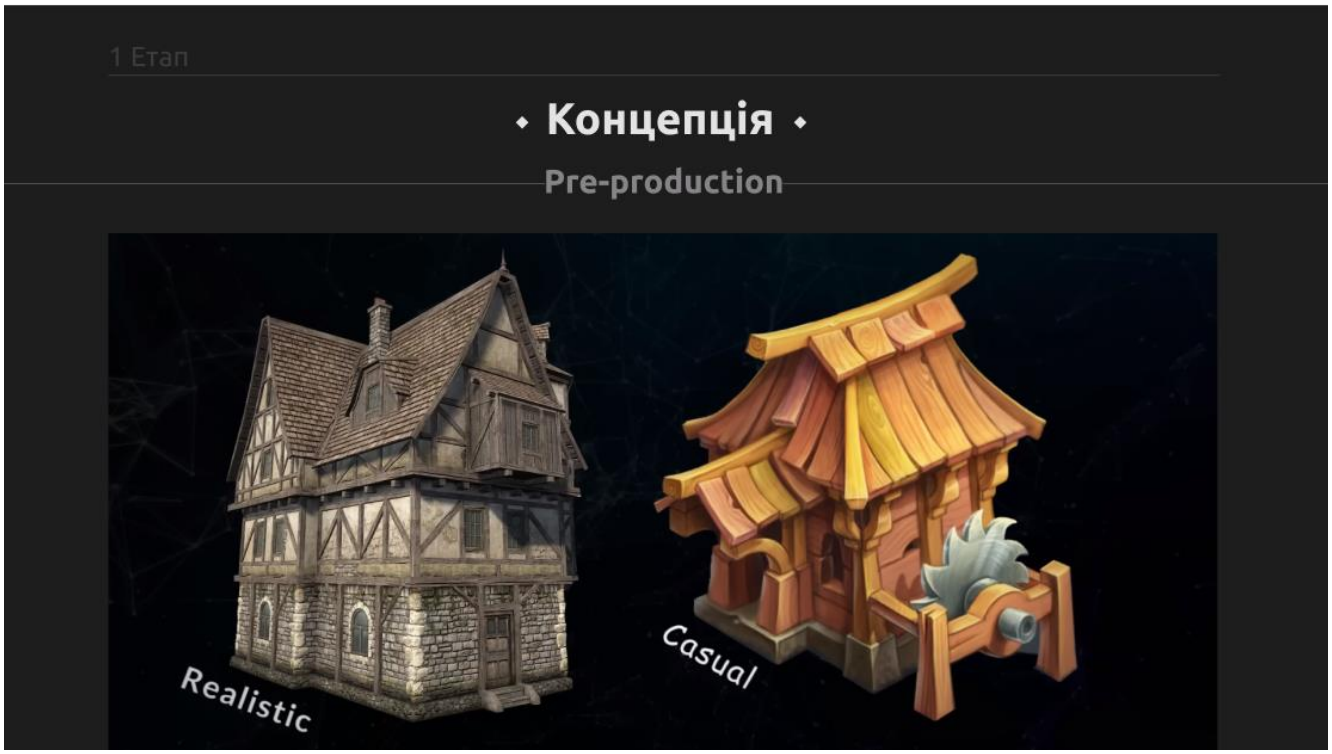


Рисунок 1.3 – Приклади стилів графіки гри

Усі фінальні думки заносяться до трьох кінцевих документів

Перший - "Concept Document" це короткий та ємний опис ідеї гри (рис.1.4):

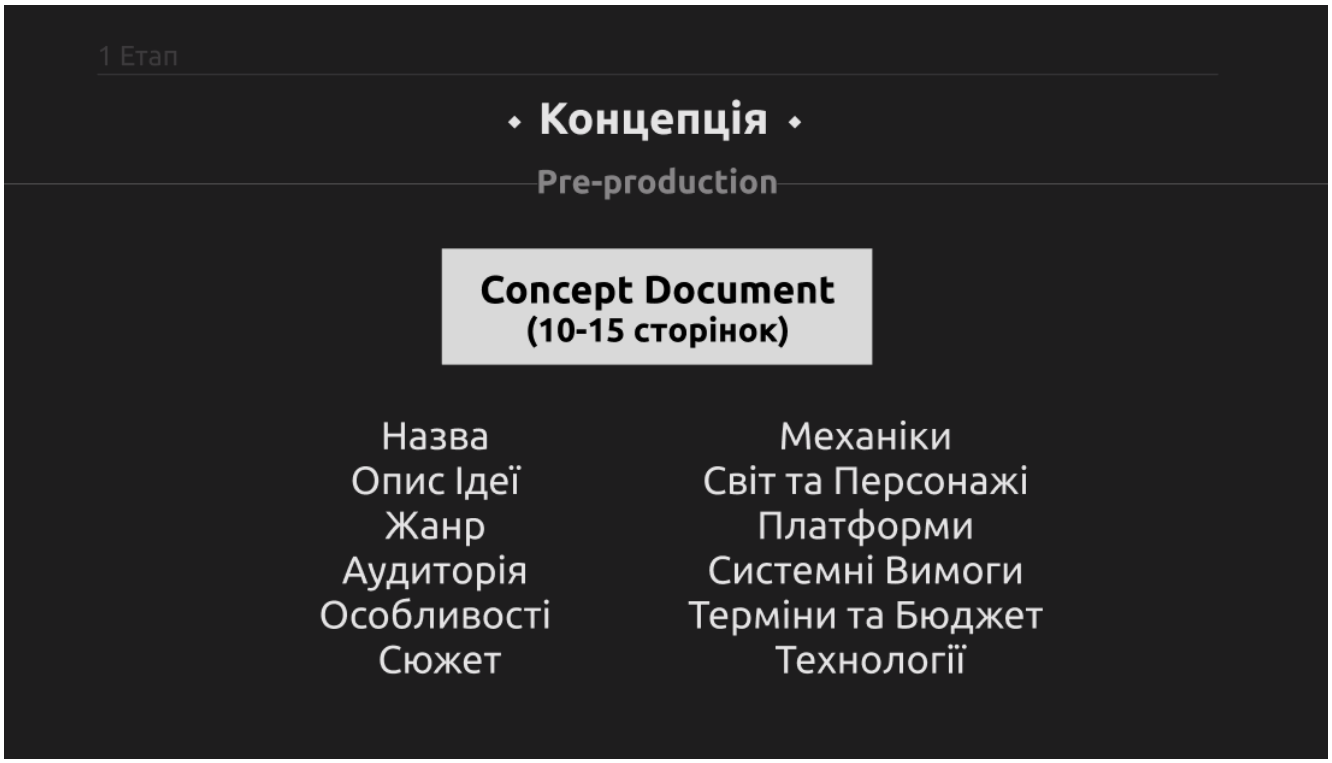


Рисунок 1.4 – Основні пункти Concept Document

Другий – "Vision Document".

У цьому документі вже описуються не сам ігровий процес, а те, що хочеться отримати в ситозі, вся гра як кінцевий бізнес продукт (рис.1.5)

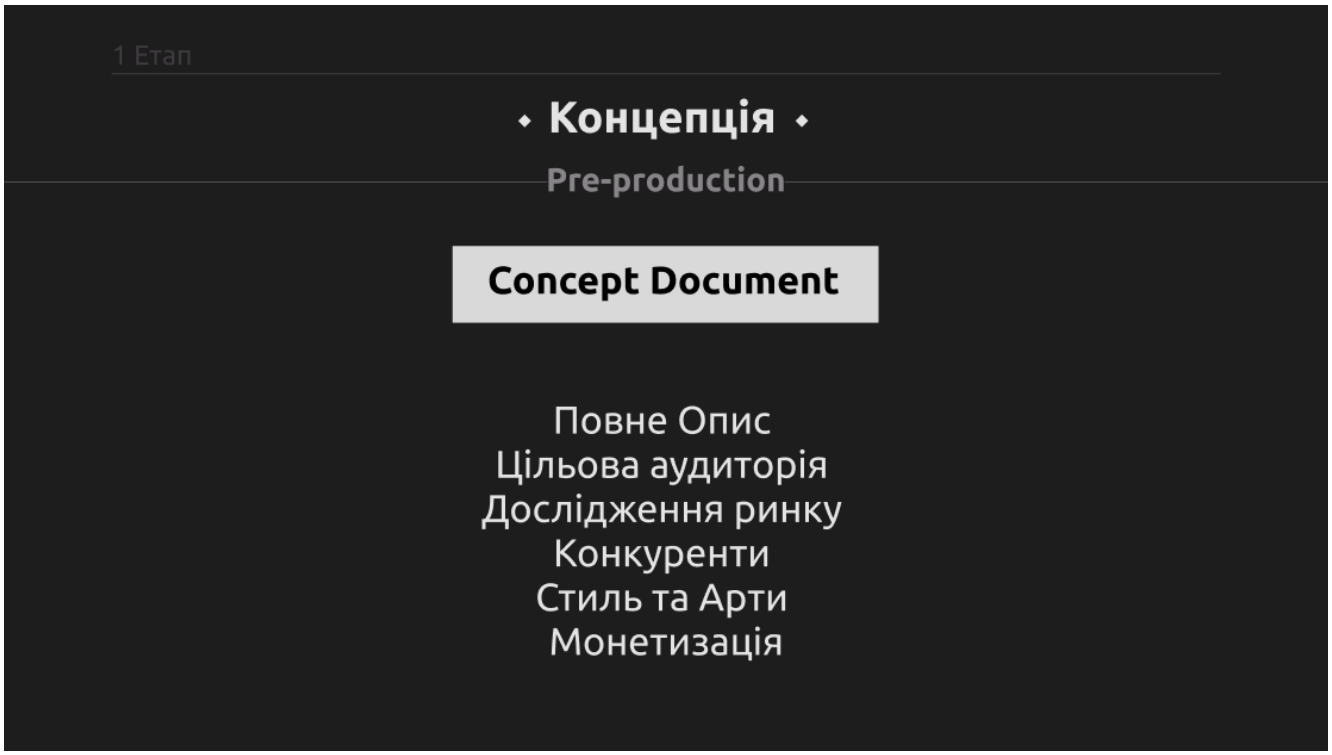


Рисунок 1.5 – Основні пункти Vision Document

Третій документ "Feature List" - список особливостей, фіч, на які деталі буде більша акцентованість, на графіку, унікальні механіки або щось ще (рис.1.6).

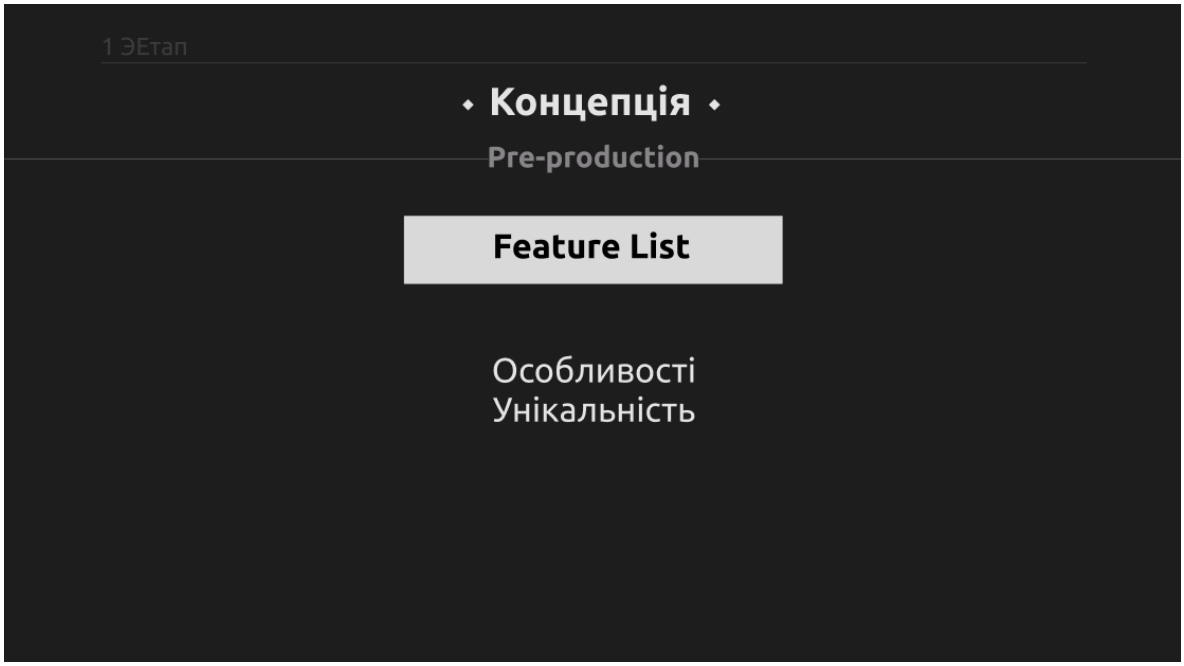


Рисунок 1.6 – Основні пункти Feature List

Також сюди входить unix select points – унікальні фічі гри, які виділятимуть її на тлі конкурентів, залучатимуть гравців та збільшуватимуть продажі.

Ці три документи взаємопов'язані між собою та формує загальний дизайн документ "Design Document" (Діздок) (рис.1.7).

Рисунок 1.7 – Основні пункти Design Document

Також у Діздок може входити опис штучного інтелекту, інтерфейсів, звукового оточення та рівнів. Якогось сертифікованого стандарту зі складання диздока не існує, все залежить від проекту та студії. Дизайн документа складають гейм-дизайнери і в ньому прописуються технічні завдання.

Є ще "Pitch List" це стисліша версія Концепт документа зі списку Фіч (рис.1.8).

Рисунок 1.8 – Основні пункти Pitch List

Відразу на ранній стадії вибирається рушій (рис.1.9).

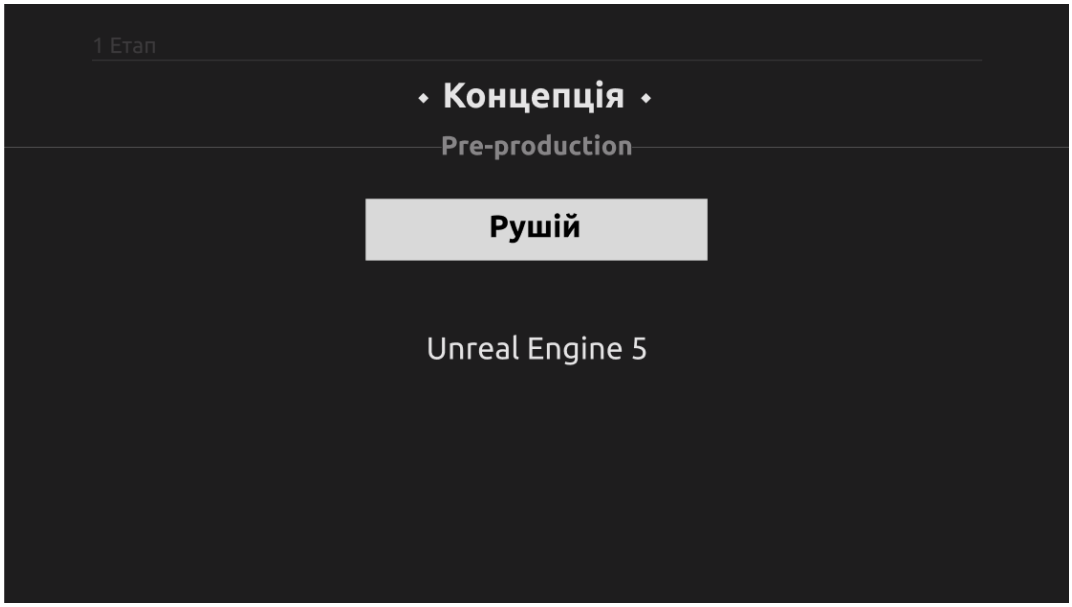


Рисунок 1.9 – Рушій проекту

Також збираються референси, фотографії з реального життя, ігор чи кіно і ці концепти репрезентують продюсерам і далі з цього всього, концепт художником стане ясно, що від них вимагається і ось сформувавши достатньо інформації, концепт художники приступатиме до створення ескізів та ілюстрацій (рис.1.10 - рис.1.12).

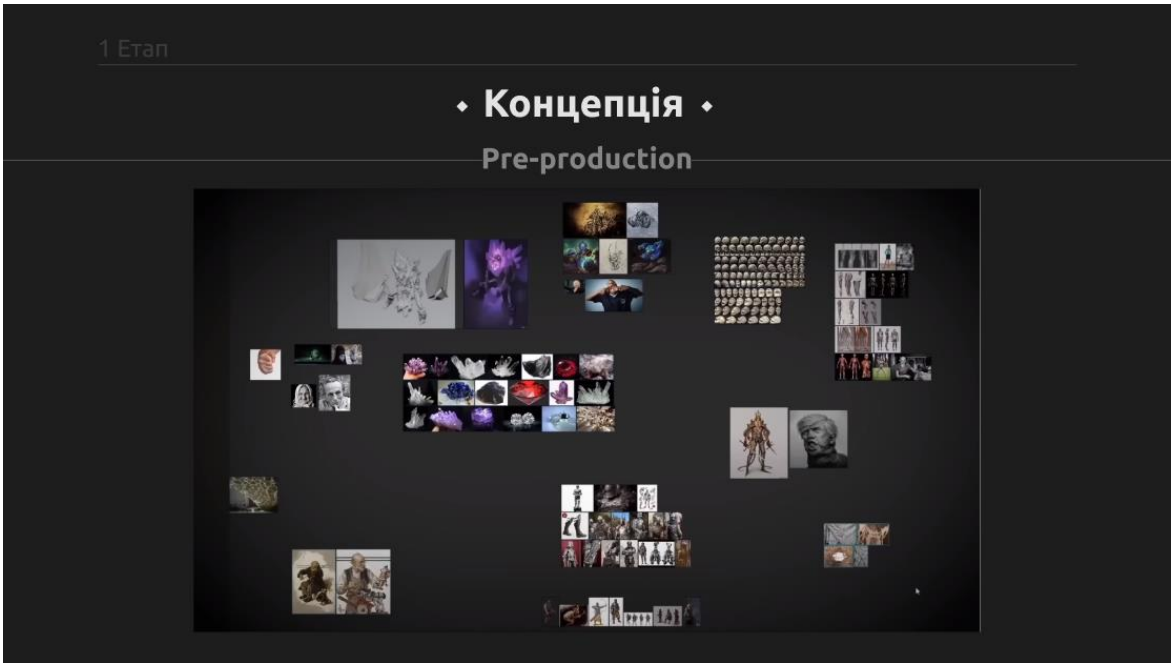


Рисунок 1.10 – Приклади рефесенців для створення ілюстрацій №1

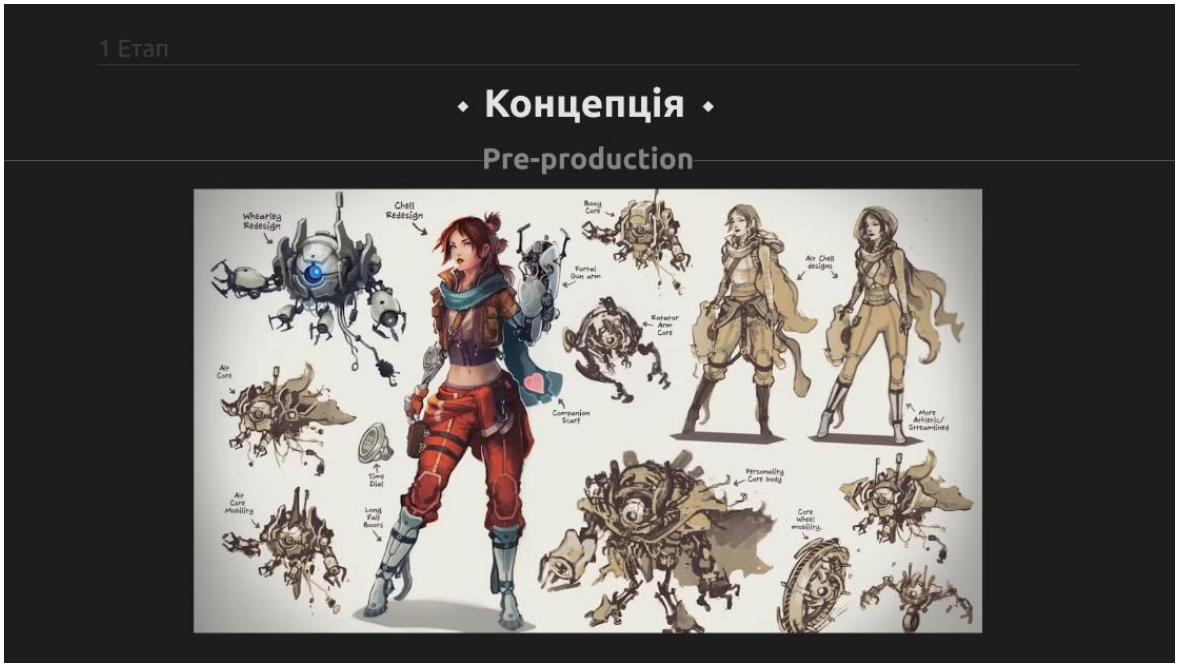


Рисунок 1.11 – Приклади рефесенців для створення ілюстрацій №2

Рисунок 1.12 – Приклади рефесенців для створення ілюстрацій №3

2 ПРОТОТИПУВАННЯ

2.1 Початок розробки

І ось починається розробка, звіряючись з раніше створеними документами, ТЗ та концептами, розробники приступають до прототипування гри.

Художники продовжують працювати над концептами і потім поступово починають створювати ігрові об'єкти – це персонажі, транспорт, рослинність, будівля, дрібні проб об'єкти та спецефекти.

Тим часом левелл-дизайнери починають виконувати свою роботу, створюють рівні (рис.2.1).

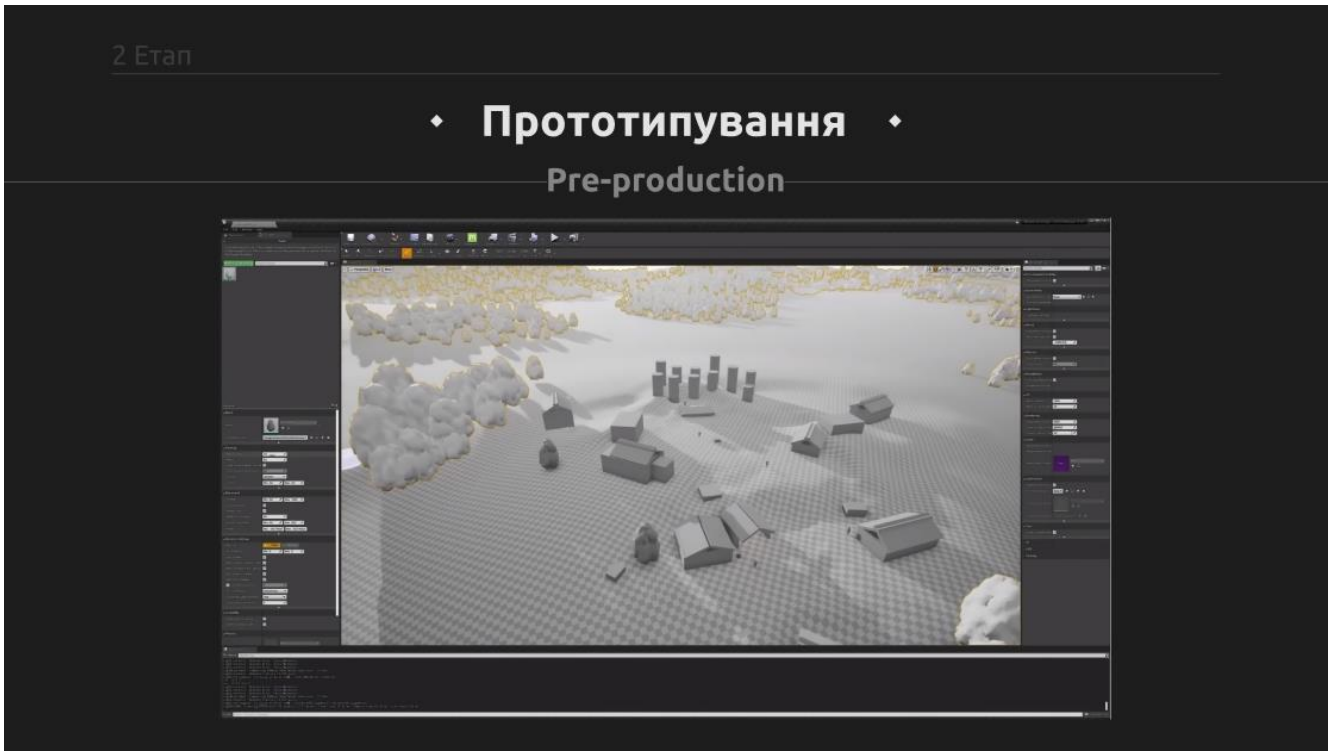


Рисунок 2.1 – Приклад рівня гри створеного левелл-дизайнером

Концепт-арти передаються 3d художникам, і вони починають свою роботу (рис.2.2, рис 2.3).

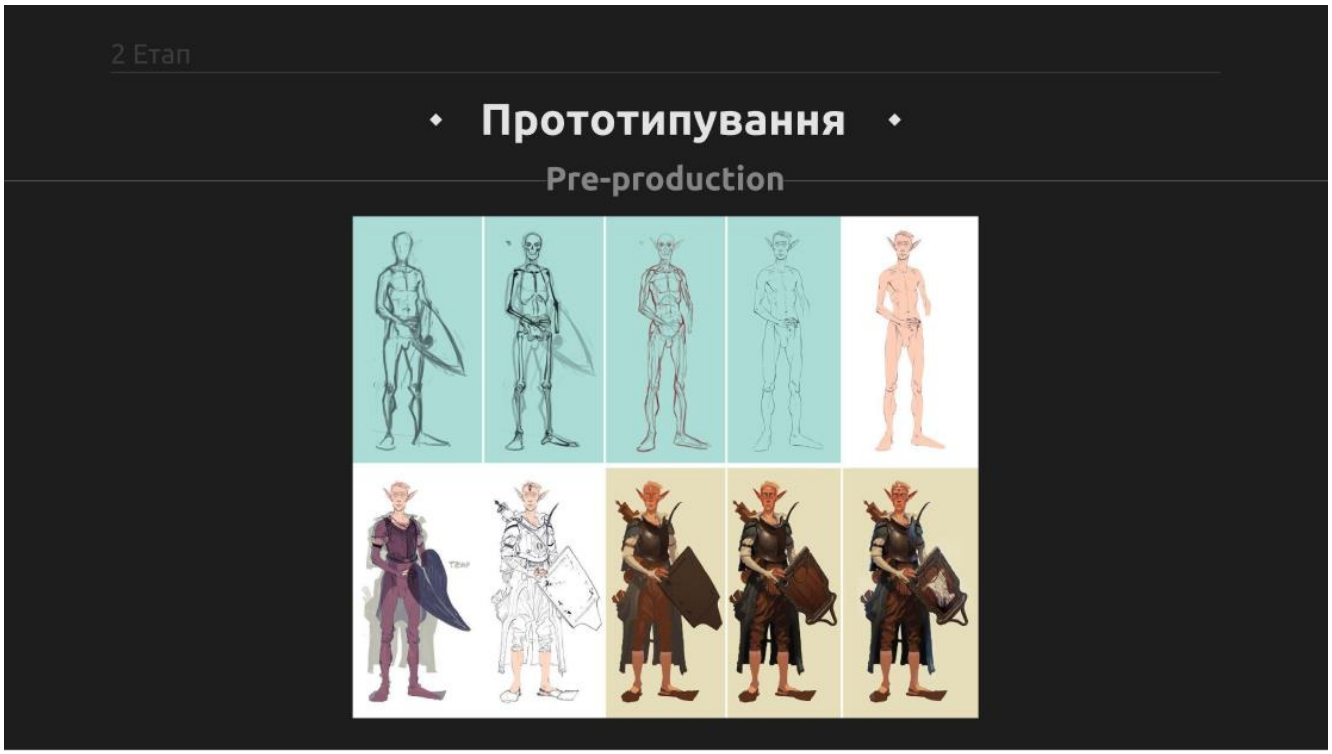


Рисунок 2.2 – Приклад концепт-артів

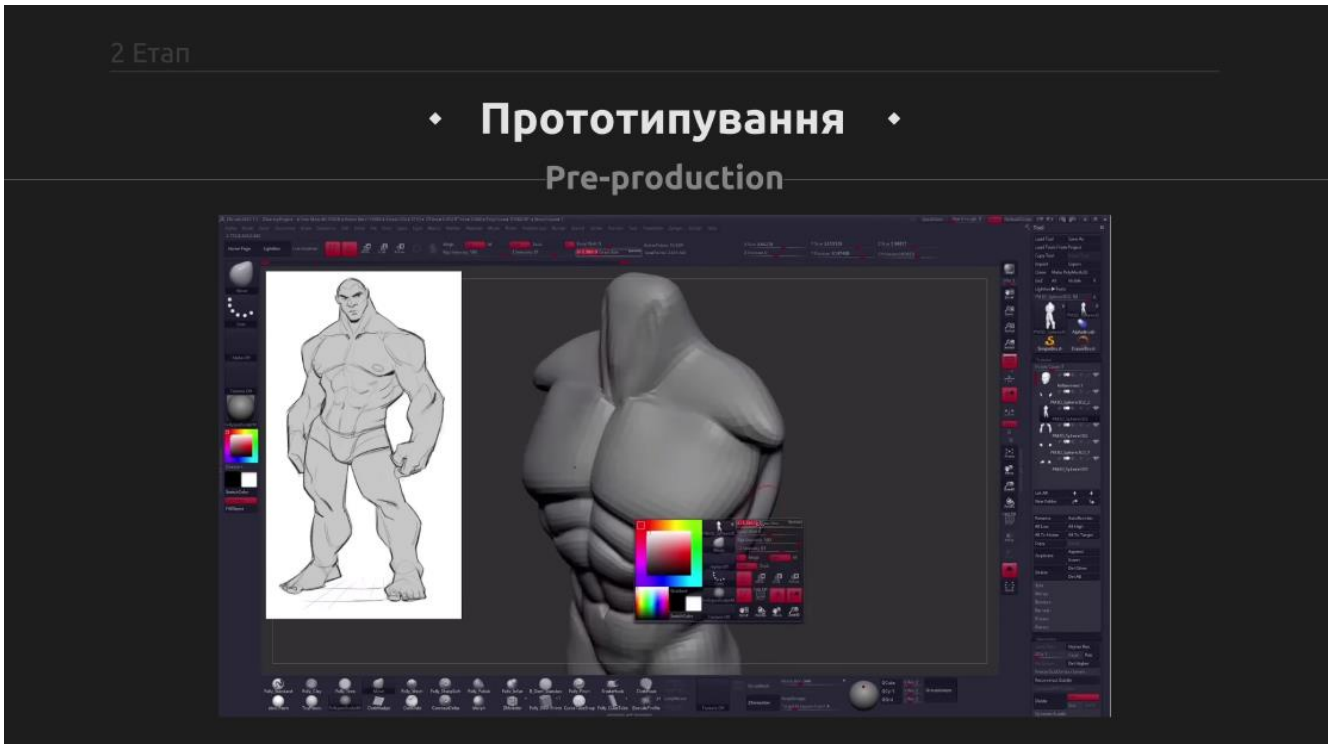


Рисунок 2.3 – Створення 3д моделы по концепт-арту

3d моделювання можна розділити на 2 основні напрямки (рис.2.4):

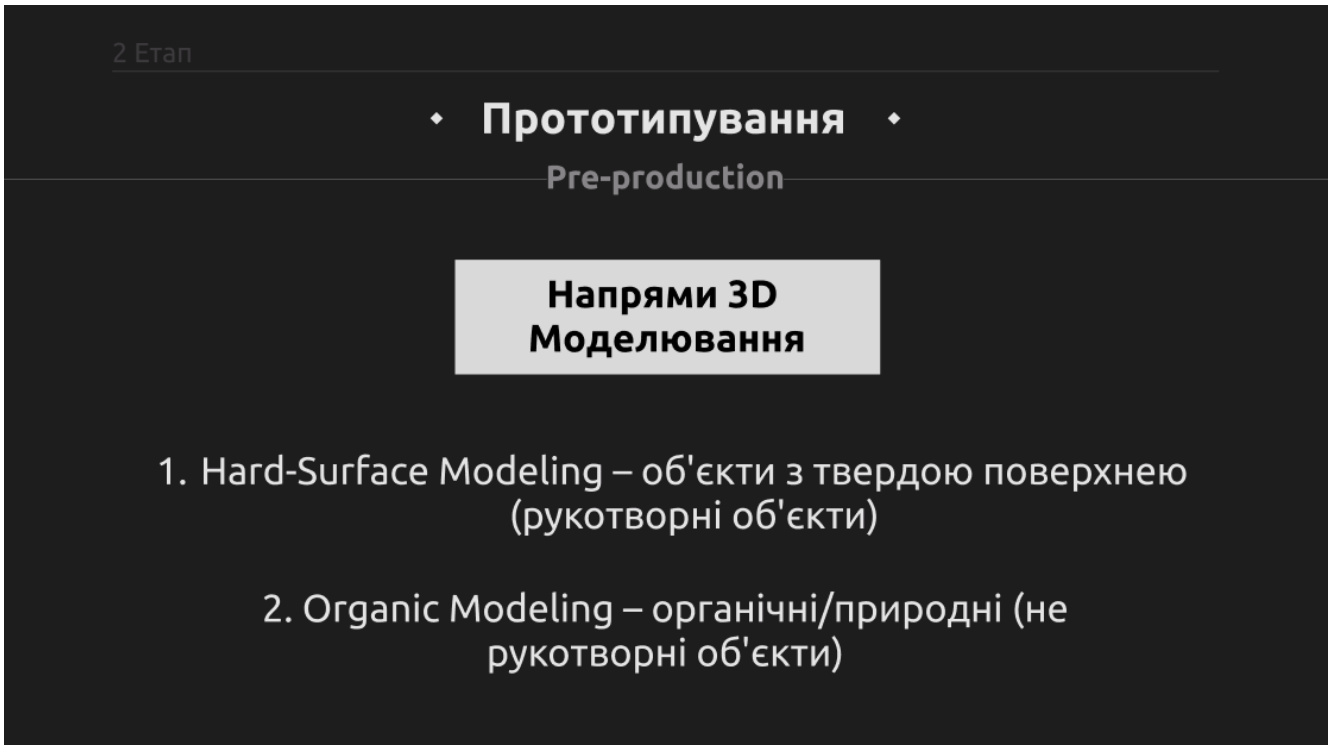


Рисунок 2.4 – Напрямки 3д моделювання

Перше це Hard-Surface Modeling. Тобто об'єкти з твердою поверхнею, зазвичай це рукотворні об'єкти, дрібні пропси, об'єкти для наповнення світу - бочки, пляшки, коробки, цеглини та інший мотлох (рис.2.5).

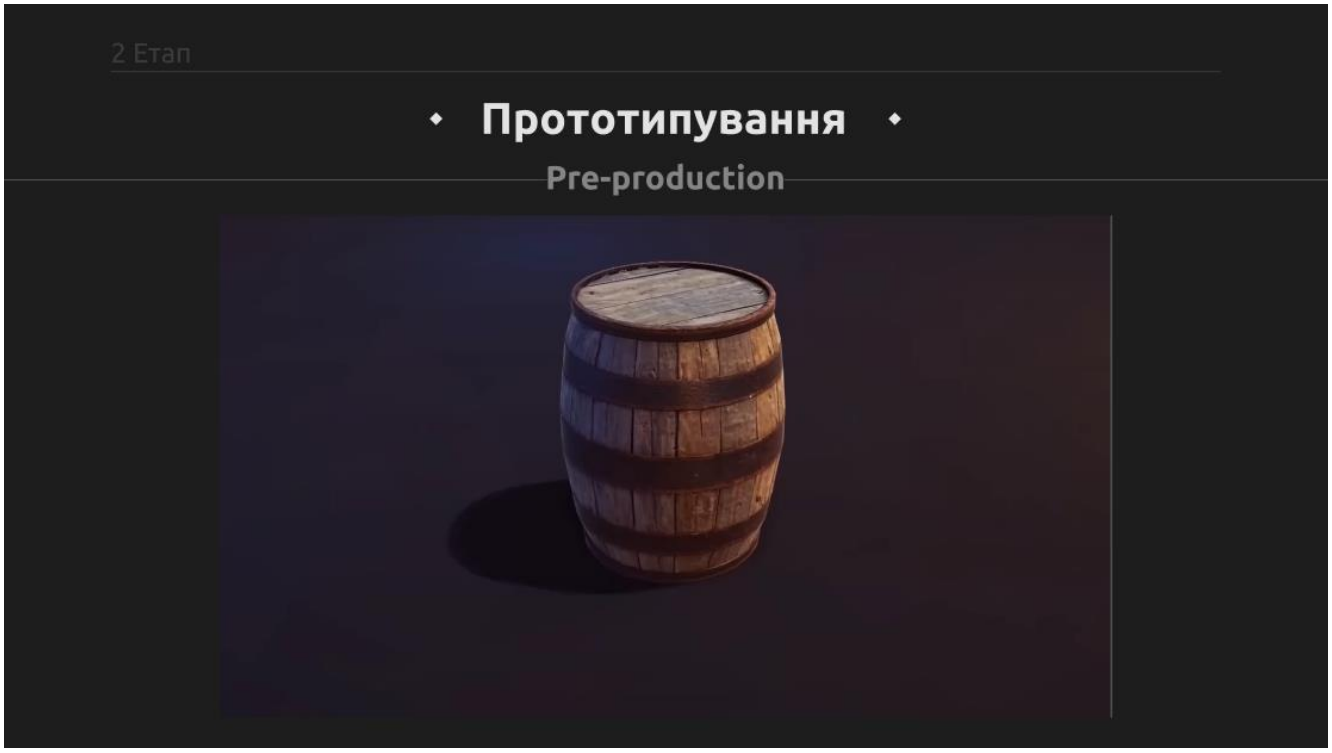


Рисунок 2.5 – Приклад моделей дрібних пропсів

До Hard-Surface також можна віднести будівлі, їх можуть збирати з окремих модулів як конструктор і якщо це дуже складні об'єкти, то вдаються до процедурного моделювання (рис.2.6).

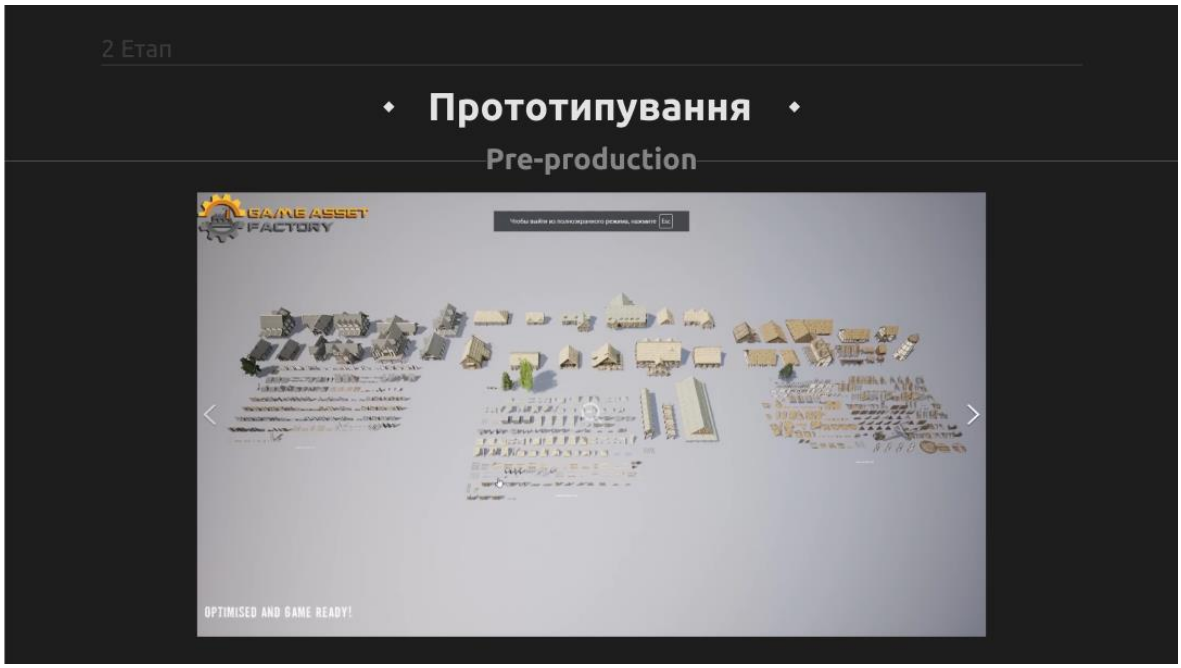


Рисунок 2.6 – Приклад моделей будівель

Використовують програми на кшталт Гудіні, де задається що, скільки і де генерувати (рис.2.7).

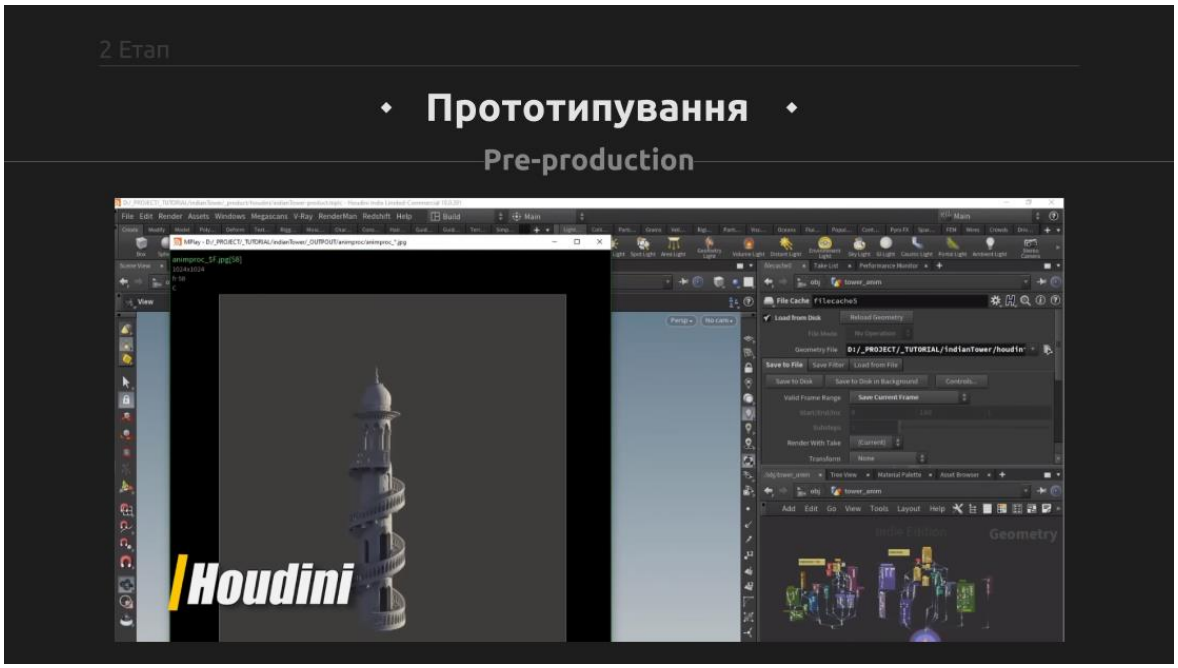


Рисунок 2.7 – Програма Houdini

Також до цієї категорії входить транспорт, складається він з 3d моделі, до якої приєднаний скелет, так само як і до персонажів, він завантажується в рушій і вже там з нього роблять автомобіль, використовуючи різний інструментарій, що дозволяє налаштувати управління, фізику, підвіску і т.д (рис.2.8).

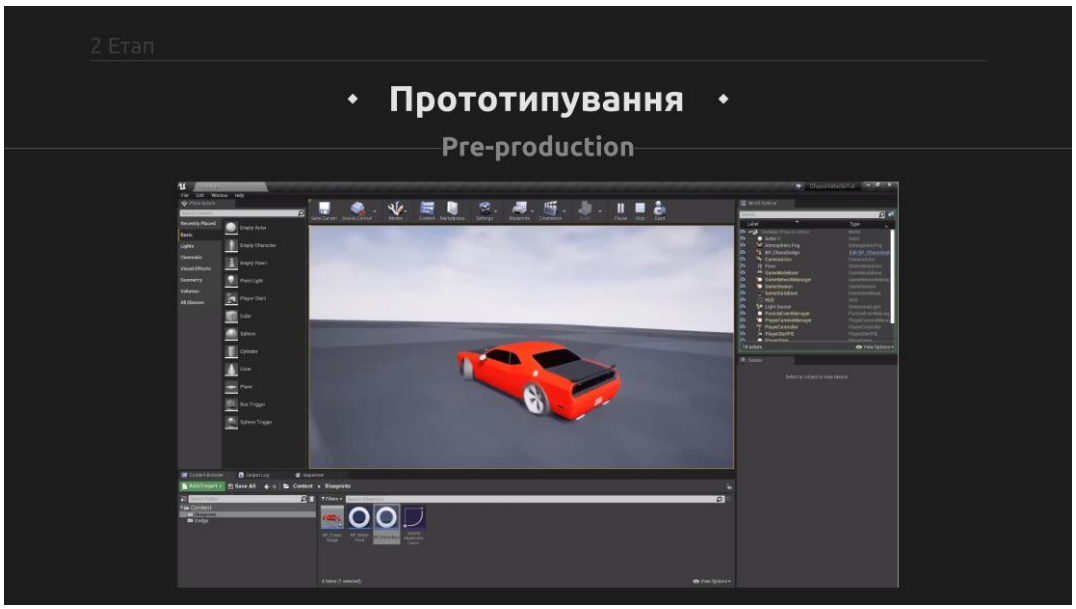


Рисунок 2.8 – Приклад моделі транспорту

Пошкодження на автомобілі бувають статичними (заздалегідь створеними) або адаптивними, тут частини кузова не замінюються іншими, пом'ятими, а деформується по полігонах, реагуючи на колізію іншого об'єкта, яскравий представник такої технології BeamNG.drive (рис.2.9).

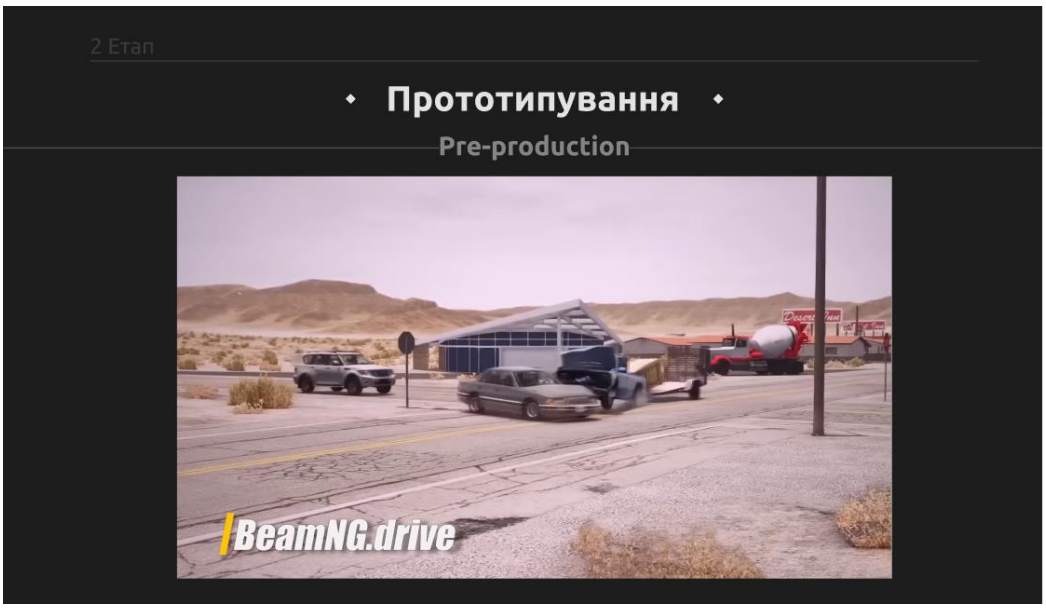


Рисунок 2.9 – Приклад пошкоджень за допомогою технології BeamNG.drive

Дерева можуть окремо генеруватися в такій програмі SpeedTree (рис.2.10).

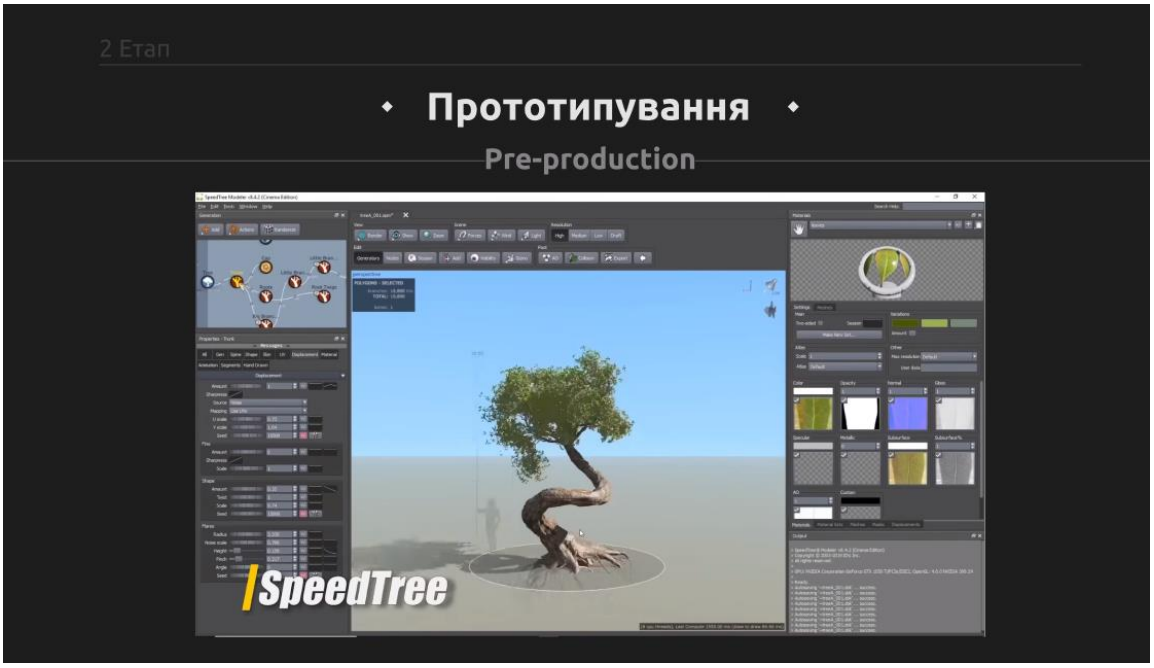


Рисунок 2.10 – Приклад генерації дерев в програмі SpeedTree

Також ігровий контент може закуповуватися на стоках, є чудова бібліотека Megascans (рис.2.11).

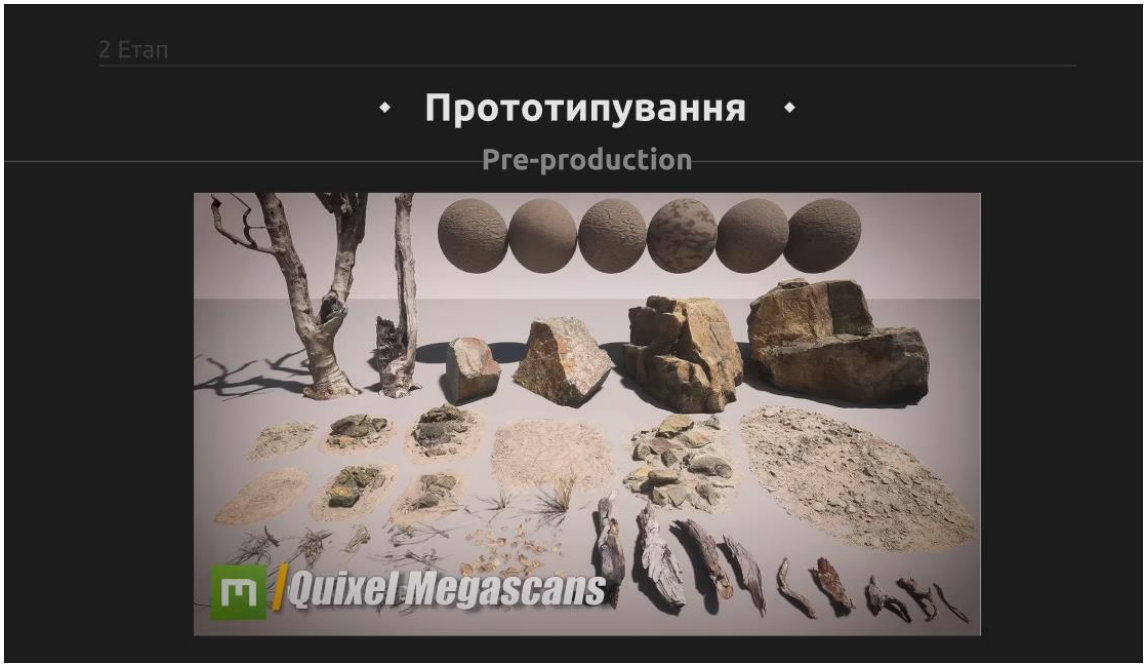


Рисунок 2.11 – Бібліотека Quixel Megascans

Ще можна сканувати об'єкт за допомогою оптичних або лазерних 3d сканерів (рис.2.12).

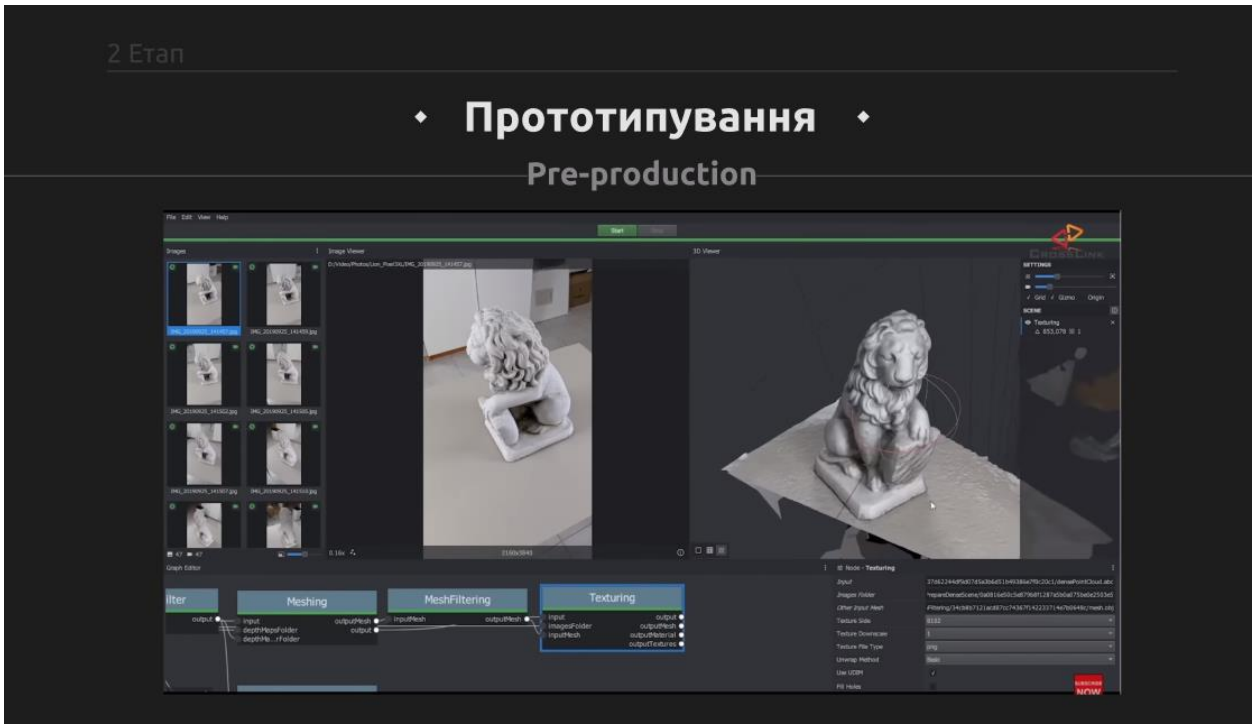


Рисунок 2.12 – Сканування об'єктів за допомогою 3d сканерів

2.2 Етапи створення персонажа

Основні етапи створення персонажа (рис.2.13):

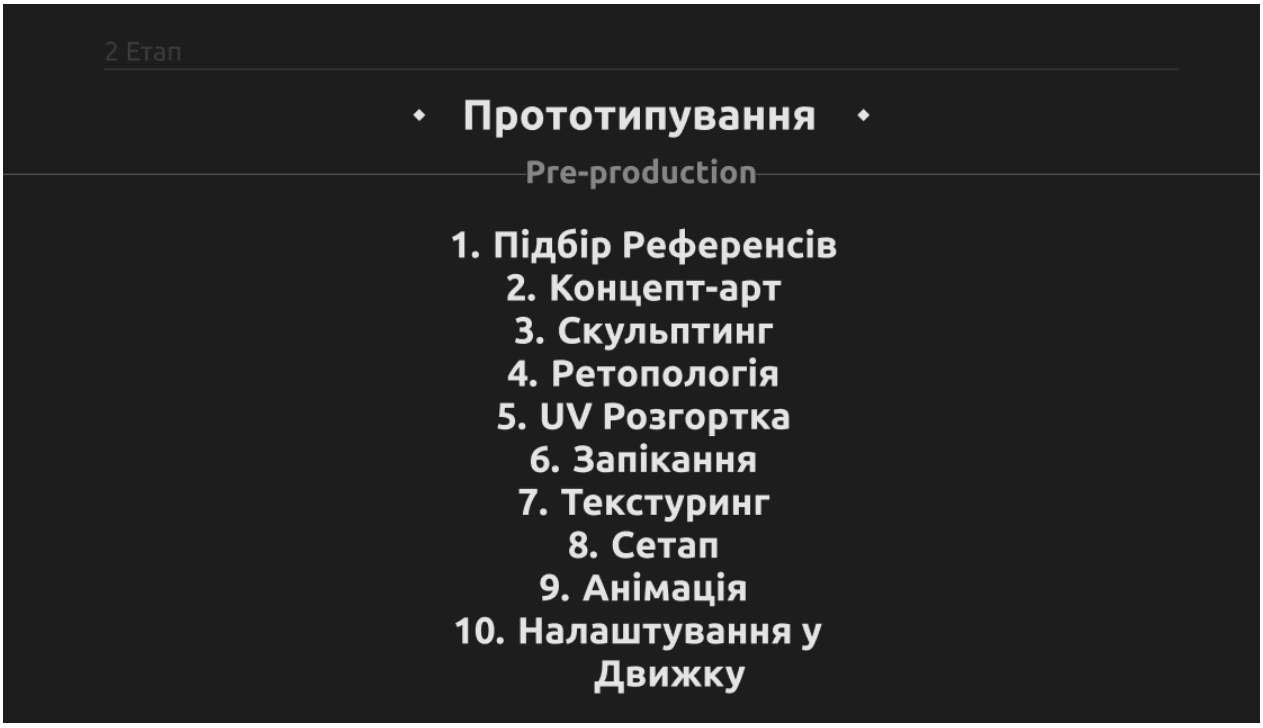


Рисунок 2.13 – Етапи створення персонажів

1.Підбір Референсів

Спершу визначається яким буде персонаж, тварина, монстр чи людина. Далі ці референції збираються до Мудбордів – це композиція з референсів на одну тематику (рис.2.14).

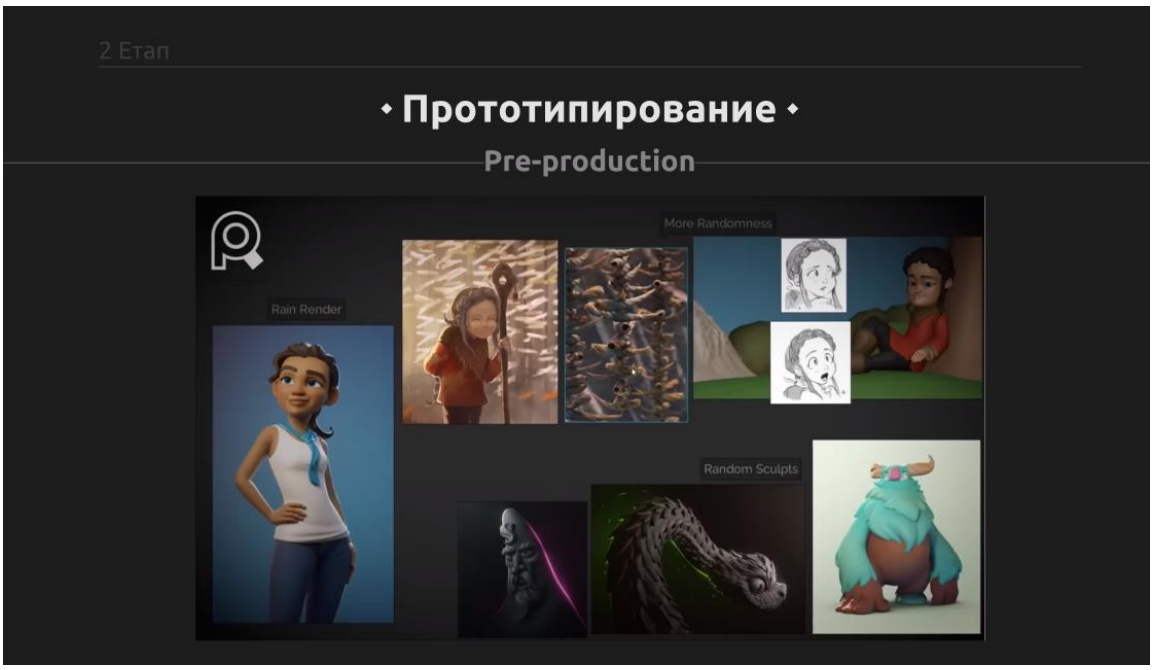


Рисунок 2.14 – Приклад референсів для створення персонажів

2.Концепт-Арт

Художник малює концепт, ескіз та передає 3d моделлеру (рис.2.15).

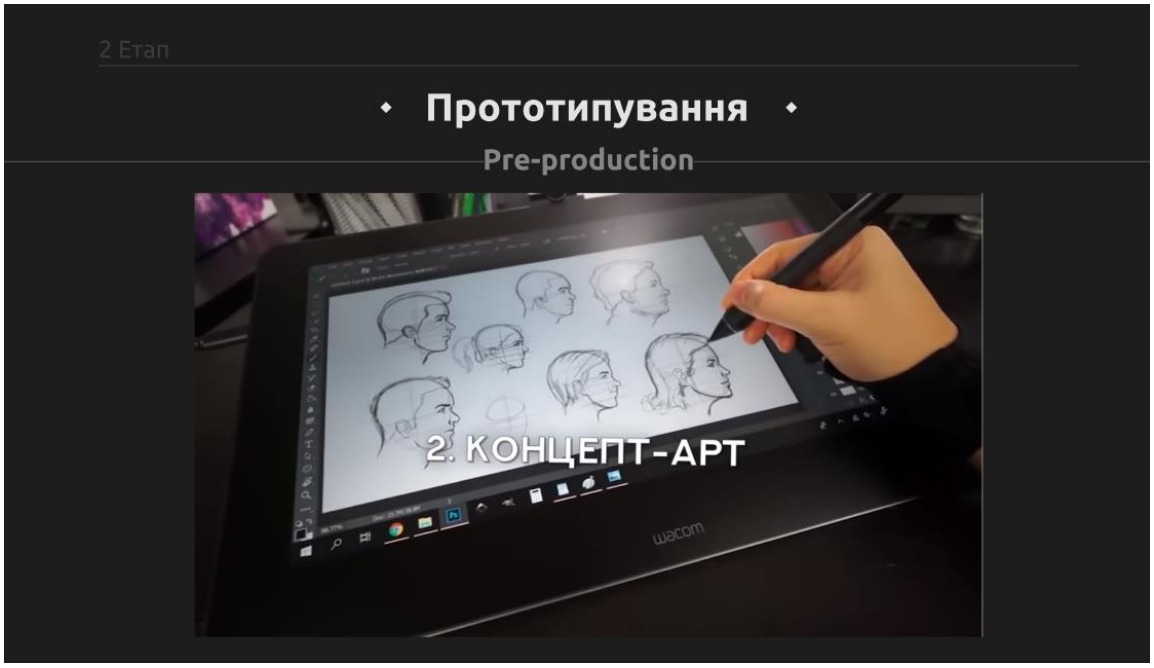


Рисунок 2.15 – Приклад концепт-арту для створення персонажів

3.Скульптинг

Далі моделлер за допомогою скульптингу створює highpoly модель персонажа використовуючи програму ZBrush (рис.2.16).

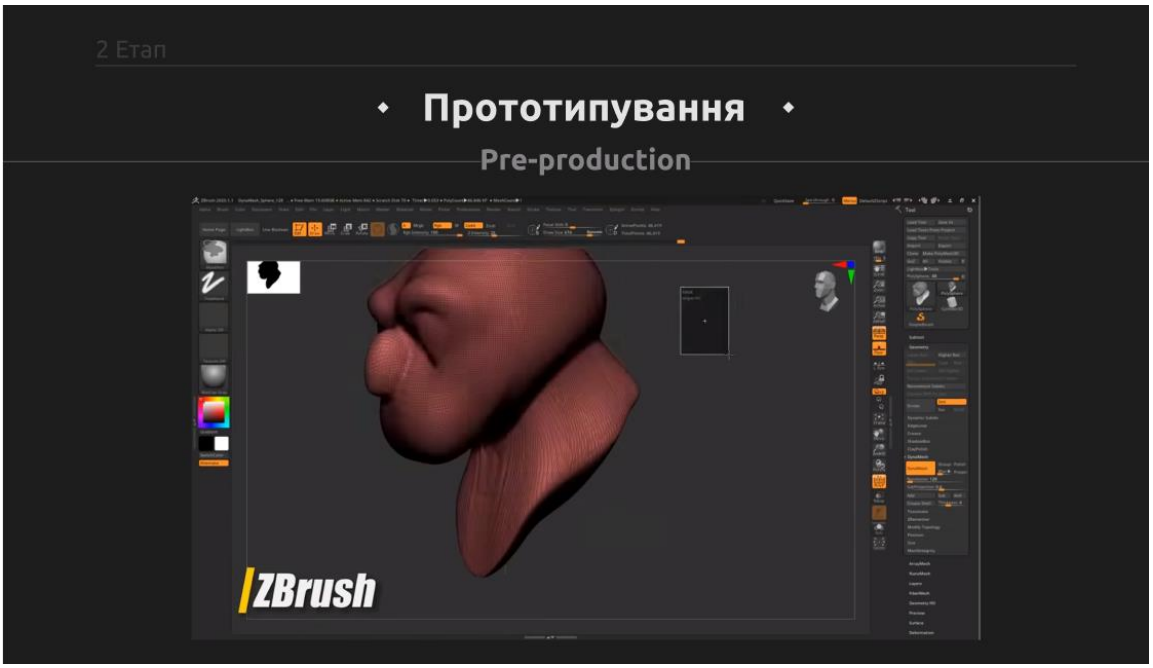


Рисунок 2.16 – Створення highpoly моделі у програмі ZBrush

Скульптинг - це як цифрове ліплення із пластиліну. Щоб моделювати було легше, скульптинг ділять на 3 або 4 ріди (рис.2.17).

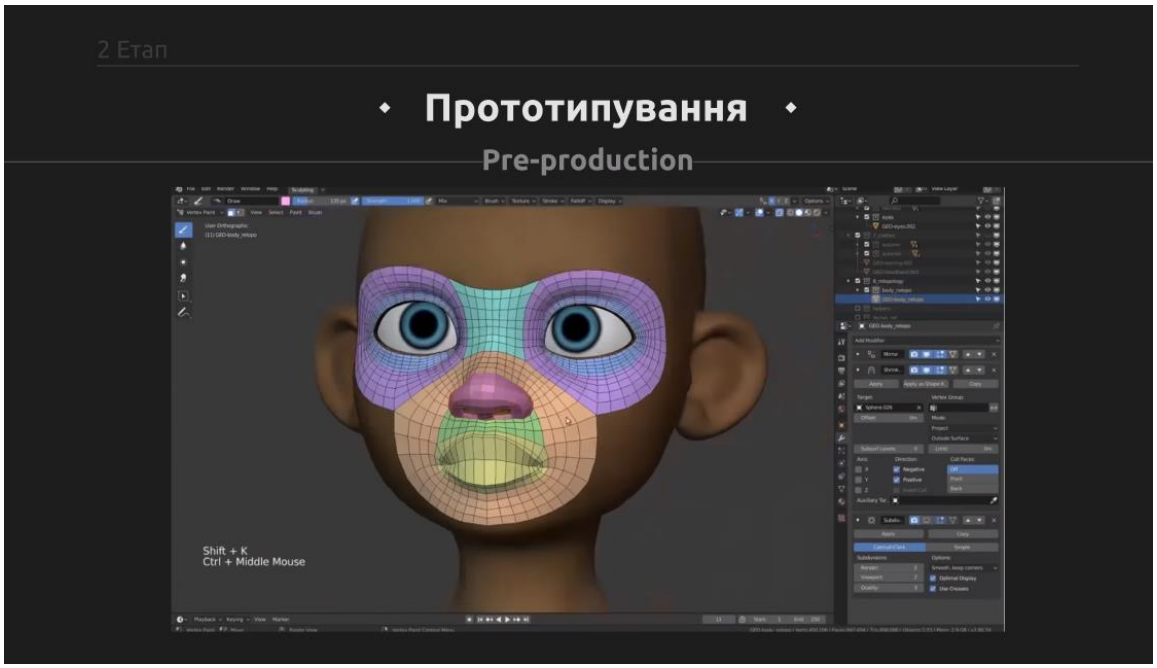


Рисунок 2.17 – Приклад скульптингу

1 Рід - створення болванки персонажа, малюнок, який є анатомією та силуетом.

2 Рід – внутрішнє наповнення та великі деталі.

3 Рід – маленькі деталі та нюанси анатомії.

І ще може бути 4 рід - понад деталізація, це можуть бути пори, зморшки і т.д (рис.2.18).

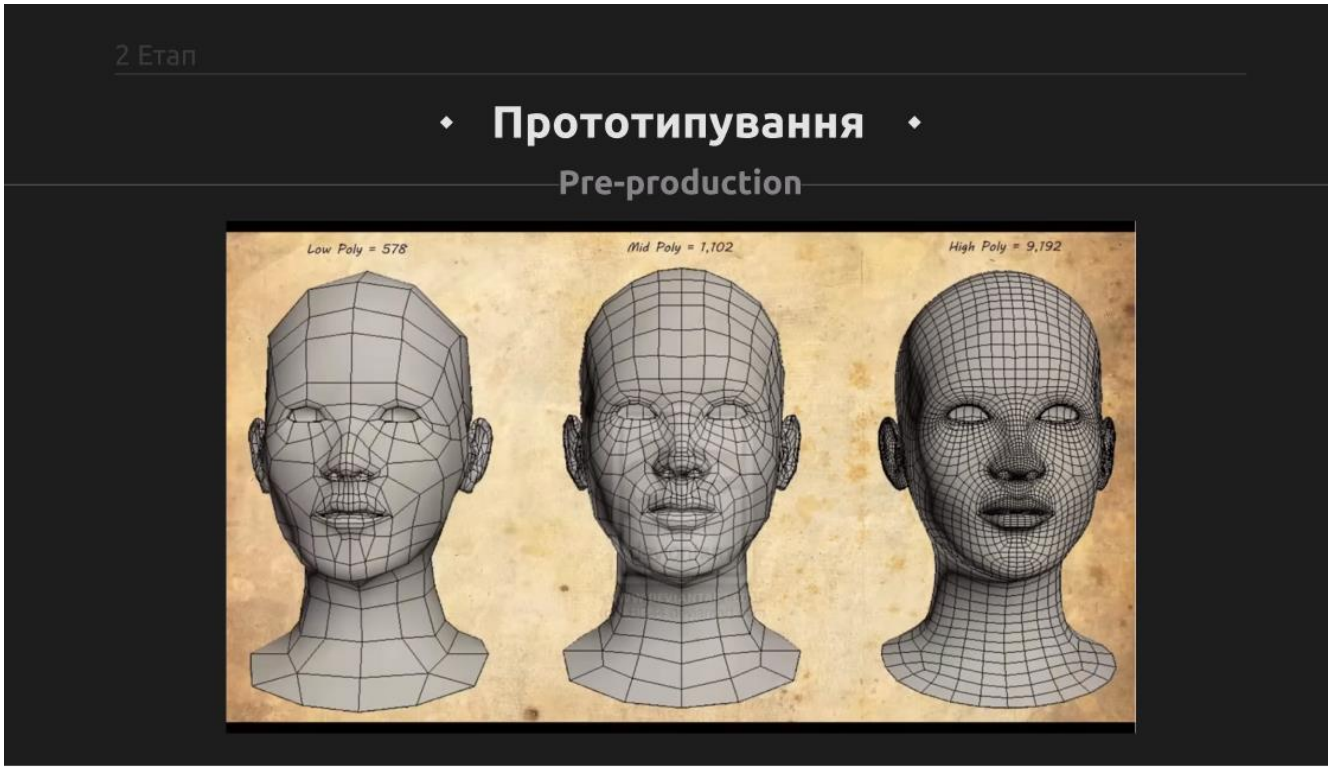


Рисунок 2.18 – Приклад рідів при скульптингу

4.Ретопологія

Далі модельєр робить ретопологію сітки та у результаті з highpoly виходить low-poly модель, вона не така вимоглива до продуктивності. Кількість полігонів на первинній highpoly моделі може обчислюватися мільйонами, 5 або 15 мільйонів, після ретопології, їх кількість зменшується до 20000. Середнє обмеження на проекті 20-80 тисяч і в результаті виходить дві моделі high і low poly (рис.2.19).

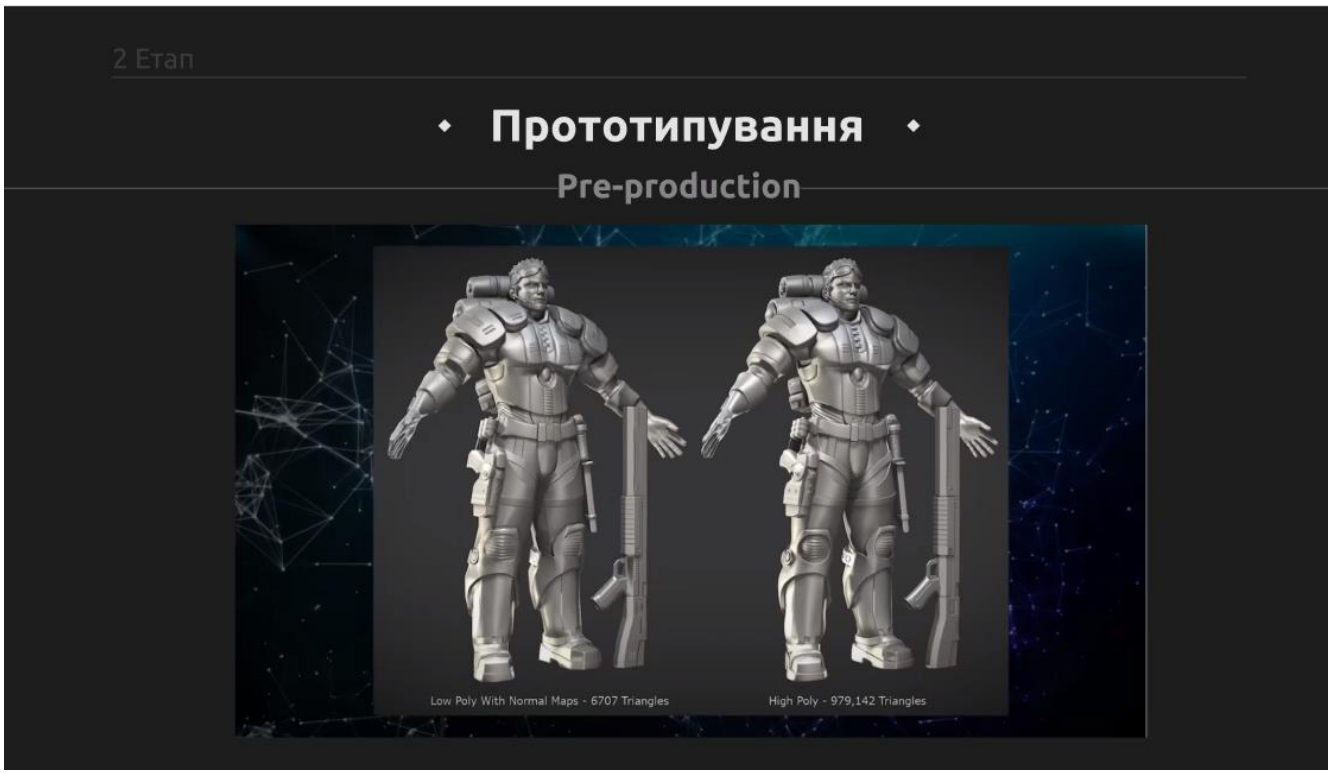


Рисунок 2.19 – Приклад highpoly та low-poly моделей

Highpoly можна використовувати для пререндрів, запікання та cinematic роликів, а low poly йде в саму гру (рис.2.20).

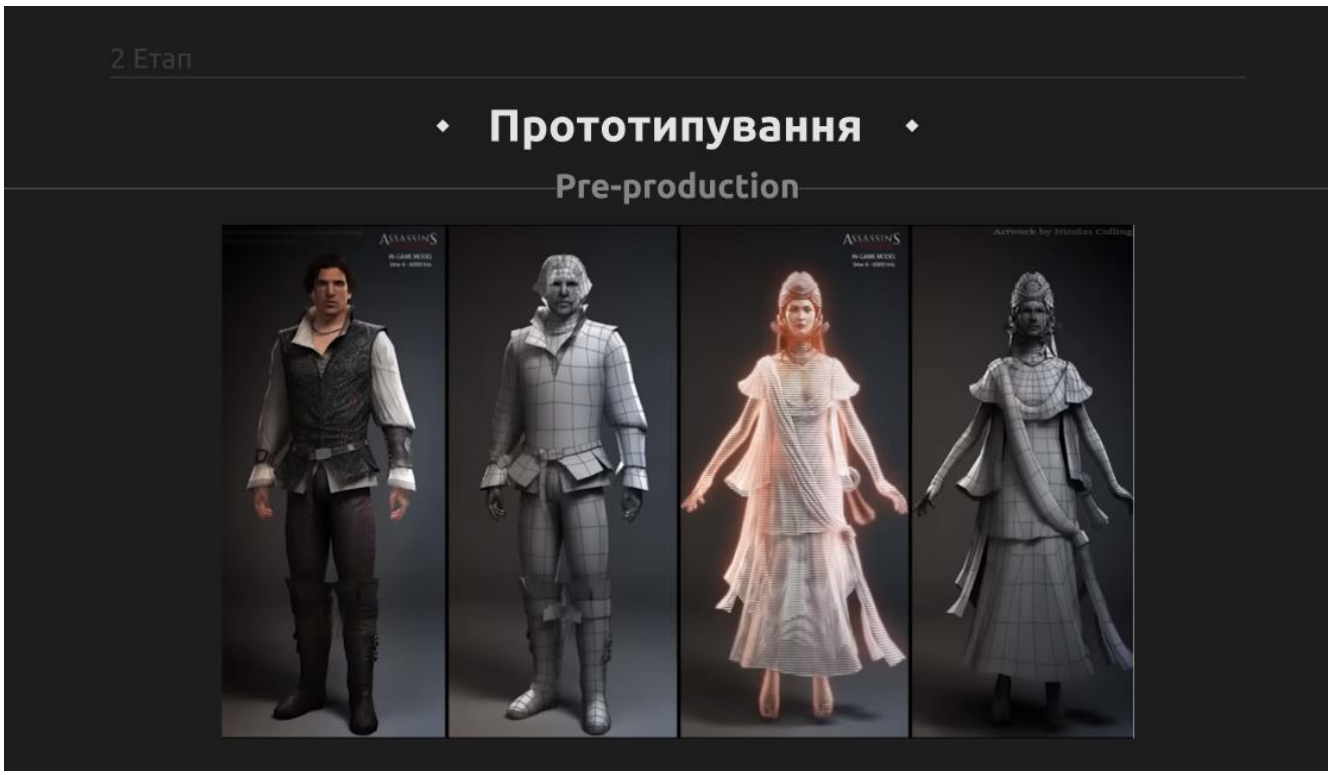


Рисунок 2.20 – Приклад low-poly моделей для гри

Для ретопології підійде будь-який 3d пакет, Блендер, Мая або 3ds max, є навіть спеціальний софт TopoGun (рис.2.21).

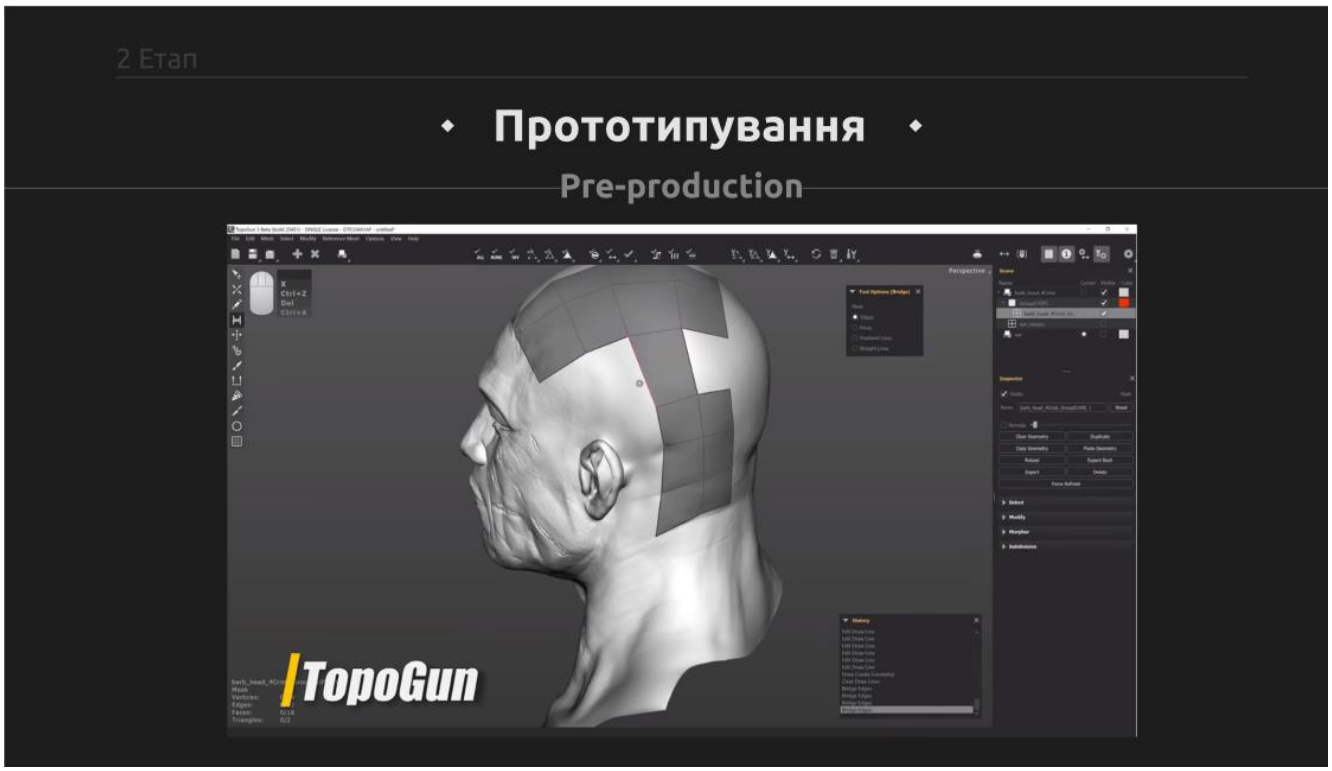


Рисунок 2.21 – Приклад програми TopoGun для ретопології

5.UV Розгортка

Після цього створюється UV карта для майбутніх текстур. Програмам для текстурування та ігровим рушіям потрібна розгортка, ця координатна сітка, яка вказує, де яка текстура на моделі буде відображатися, щоб вони розуміли, яка частина малюнка якій ділянці 3d моделі належить.

3d модель розгортається у площину, розгортку можна робити як вручну, так і автоматично (рис.2.22, рис.2.23).

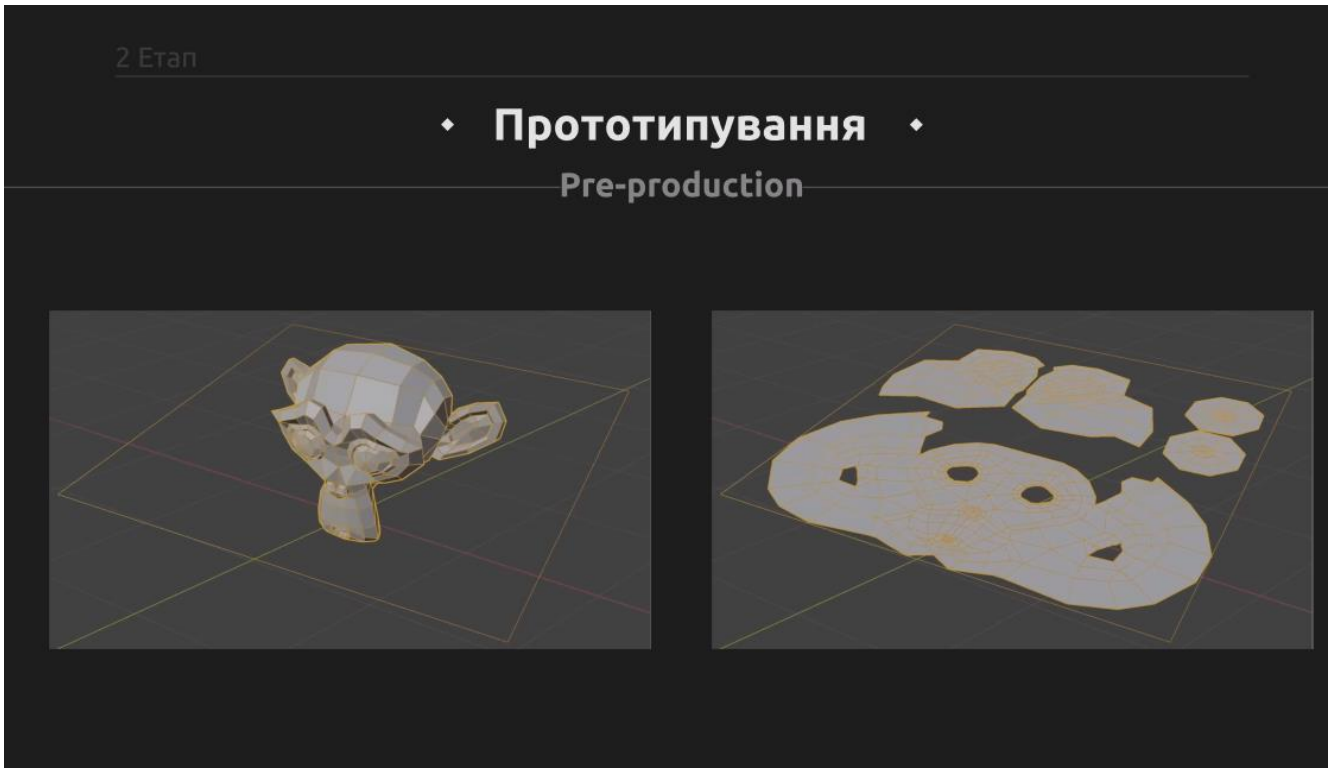


Рисунок 2.22 – Приклад розгортки №1

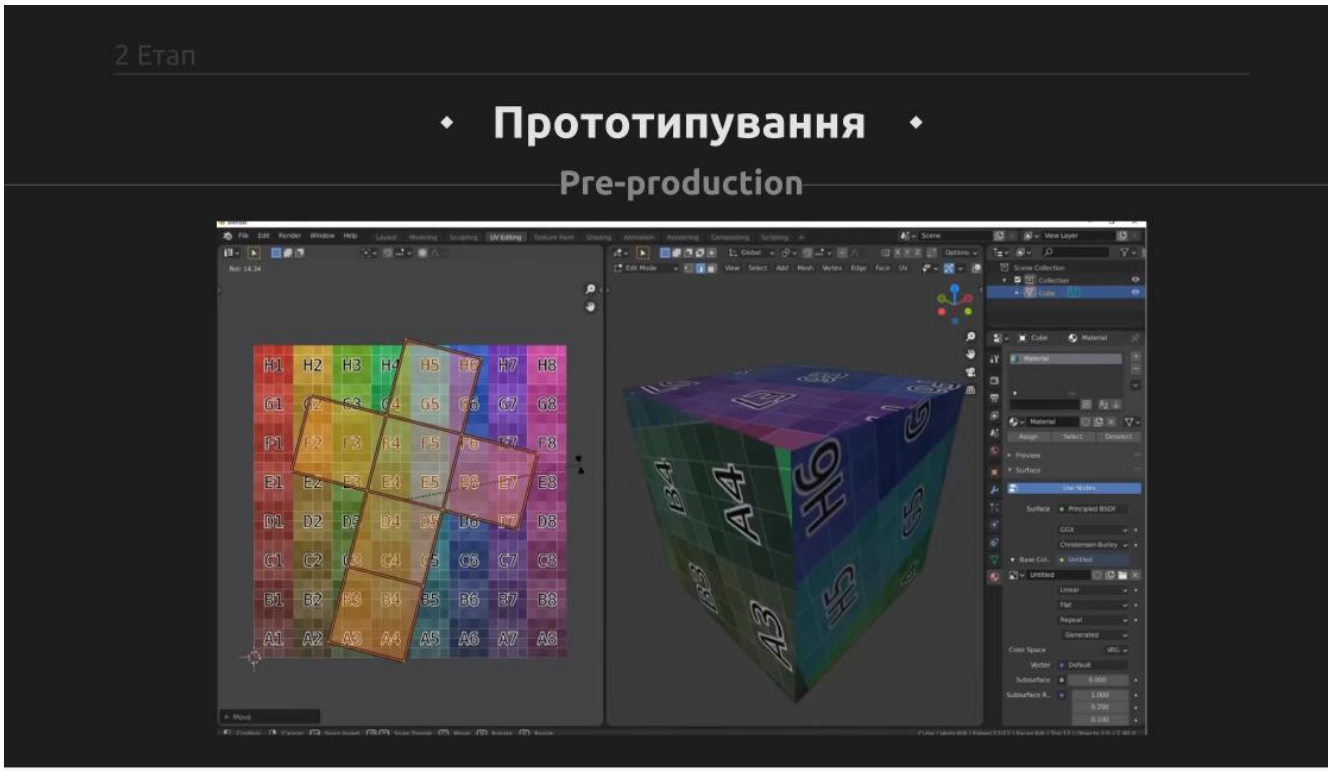


Рисунок 2.23 – Приклад розгортки №2

6. Запікання

Наступний етап, запікання, на цьому етапі деталізацію highpoly моделі можна запекти в карту нормалей, це така текстура об'єму, це плоска ілюзія об'єму як 3d листівки, заодно можна запекти ambient occlusion - це затінення в стиках, отримані від м'якого глобального освітлення.

Простіше кажучи, запікання - це перенесення візуальної інформації з високоюполі на low poly у вигляді текстур (рис.2.24).

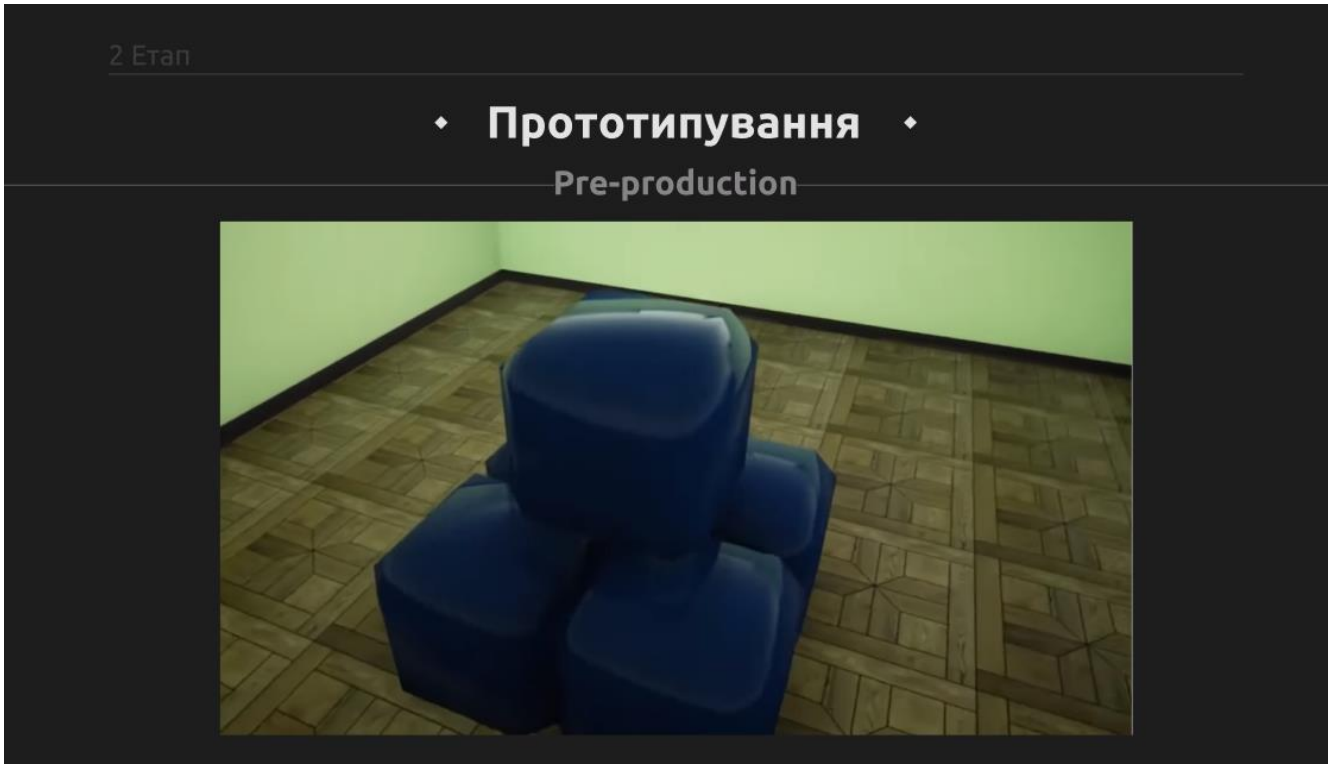


Рисунок 2.24 – Запікання деталізації у карту нормалей

7. Текстуринг

Після того, як персонаж готовий його необхідно розфарбувати, це може робити як сам 3d моделлер, так і художник. Зазвичай у промисловості при цьому використовується Substance Painter (рис.2.25).

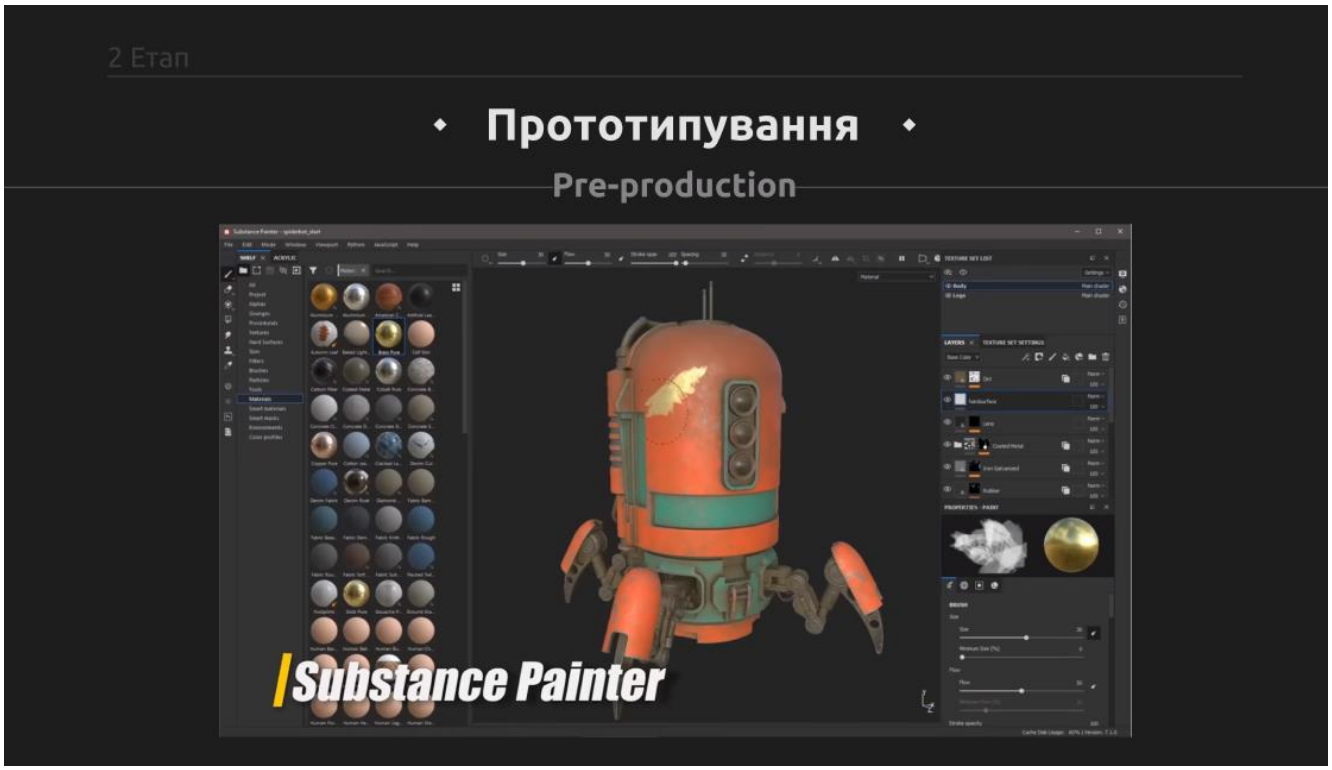


Рисунок 2.25 – Приклад фарбування персонажу у Substance Painter

Всі ці відблиски, потертості – це різні шари текстур, які називають мапами – сюди входять Diffuse, Roughness, Emissive, Normal (рис.2.26).

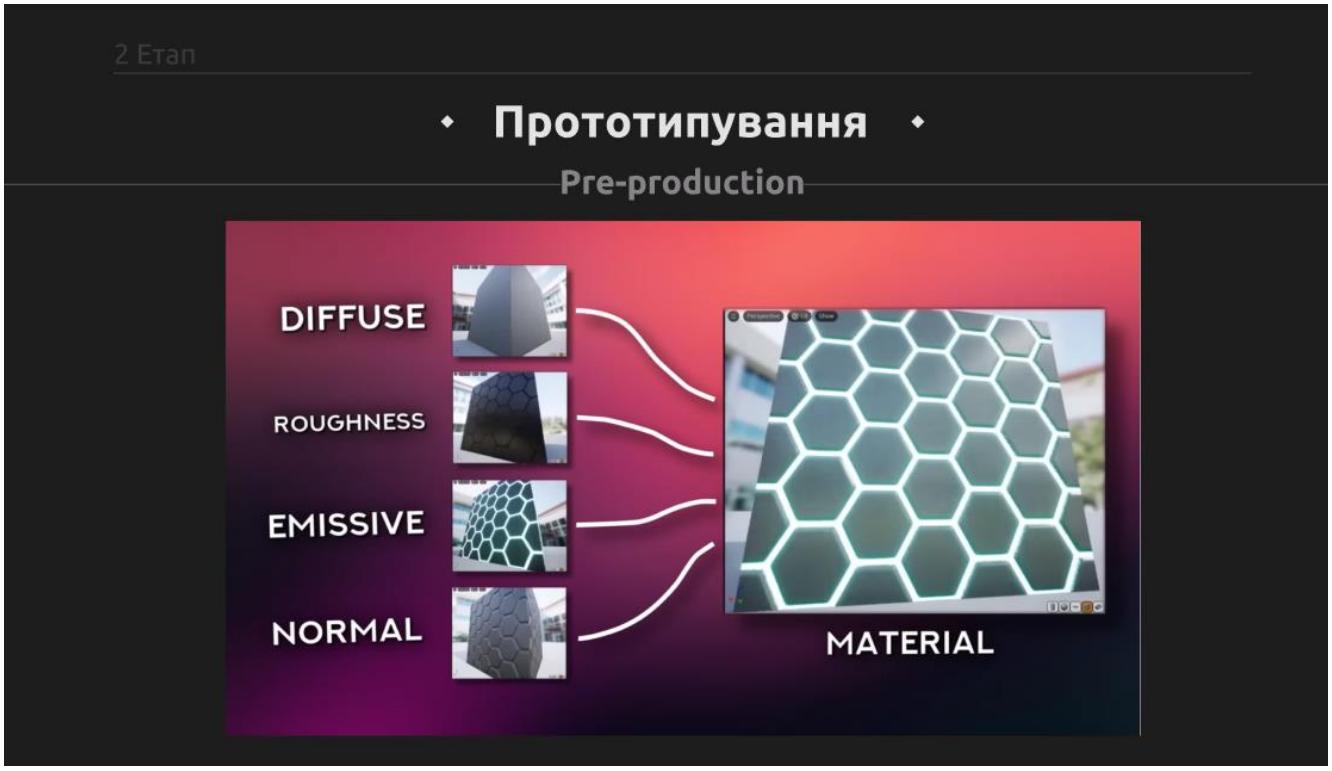


Рисунок 2.26 – Приклад різних шарів текстур або мап, які обєднуються у матеріал

Нормалі також можна створювати на етапі текстурування - ці карти поєднуються в загальний матеріал і надалі цей матеріал буде накладений на модель за допомогою UV координатної сітки (рис.2.27).

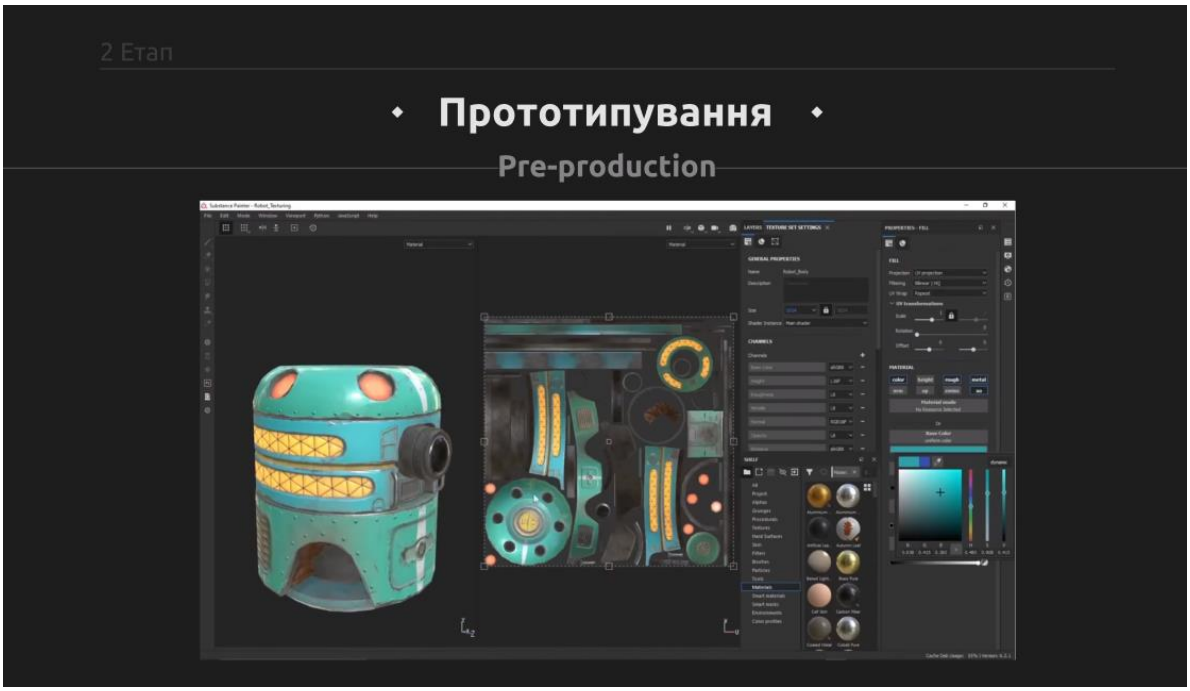


Рисунок 2.27 – Створення нормалей на етапі текстурування

Якщо це дорогий персонаж, то до роботи можуть підключатися художники, які працюють з одягом та художники, які створюють волосся (рис.2.28).

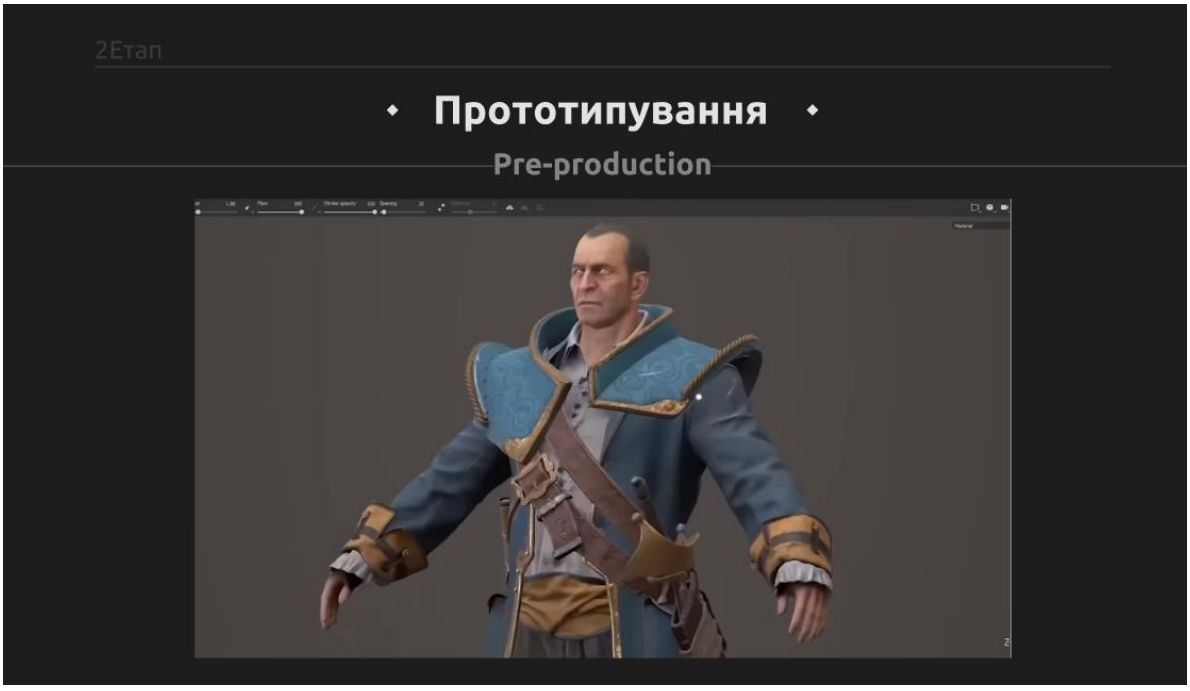


Рисунок 2.28 – Створення одягу і волосяся художниками

8. Сетап

Наступний крок підготовка до анімації. Модель персонажа переносяться в інший 3d пакет, це може бути 3ds max або Мая.

Або для більш фізично точної анімації використовують програму "Cascadeyr (Каскадер)" (рис.2.29).

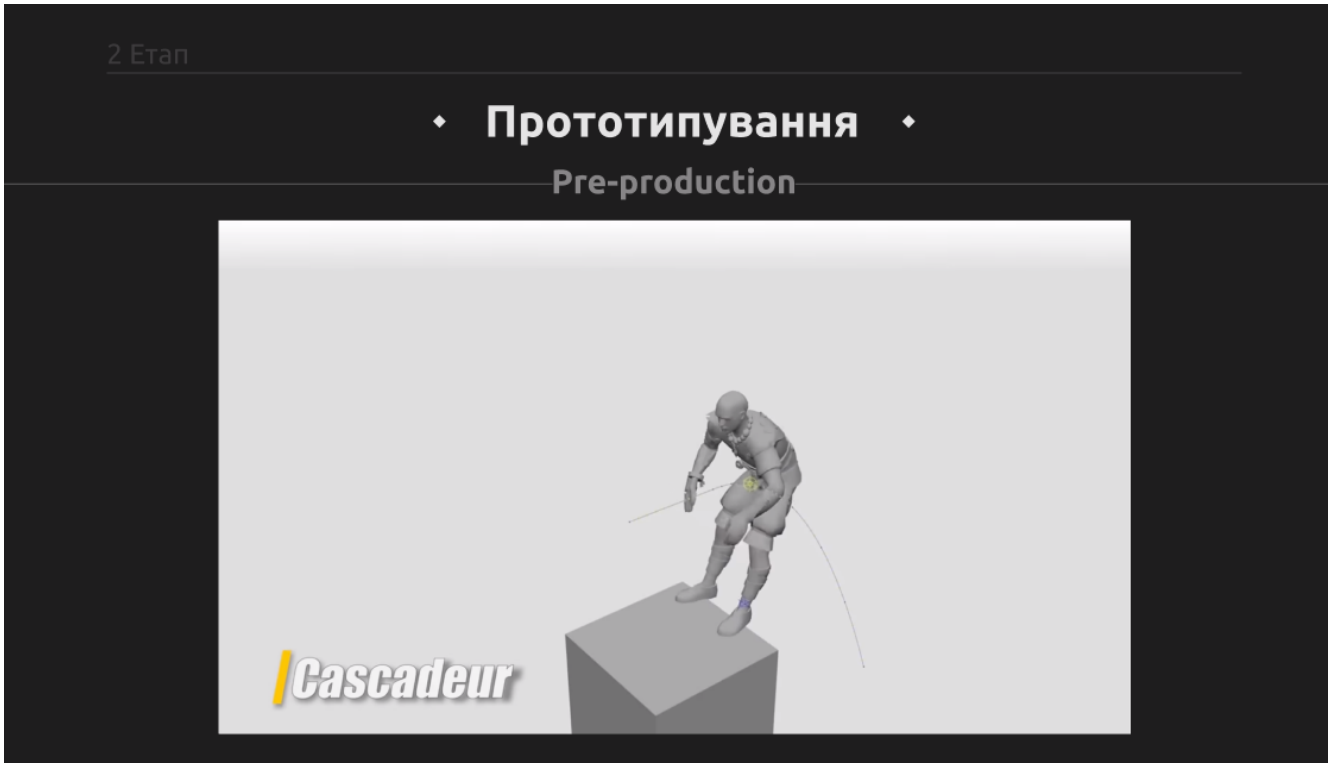


Рисунок 2.29 – Програма Cascadeyr для точної анімації

Для майбутньої скелетної анімації починається процес побудови скелета (Ріггінг).

В одному персонажі в залежності від складності може бути від 20 до 100 кісток і навіть ще більше, якщо знадобиться правдоподібна лицьова анімація, то кількість кісток можна помножити на 2, але а так у дорослої людини приблизно 208 кісток.

Частини скелета, суглоби називаються Джоїнтами (рис.2.30).

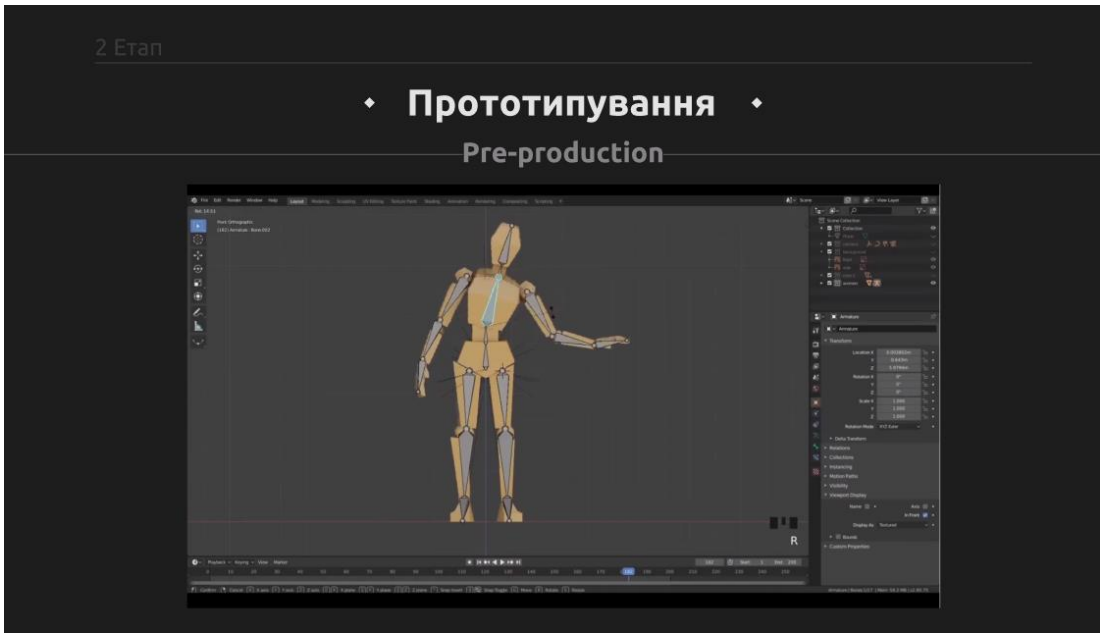


Рисунок 2.30 – Ріггінг та скінінг

Потім готовий скелет потрібно прикріпити до тулуба, цей процес називається skinning і всім цим може займатися окрема людина.

9. Анімація

Після чого персонаж переходить до рук аніматора. Анімація починається з розкадрування та використовуються анімації по ключових кадрах, перший ключ це вихідне положення, а другий ключ кінцеве та кадри між ключами автоматично плавно підлаштовуються (рис.2.31).

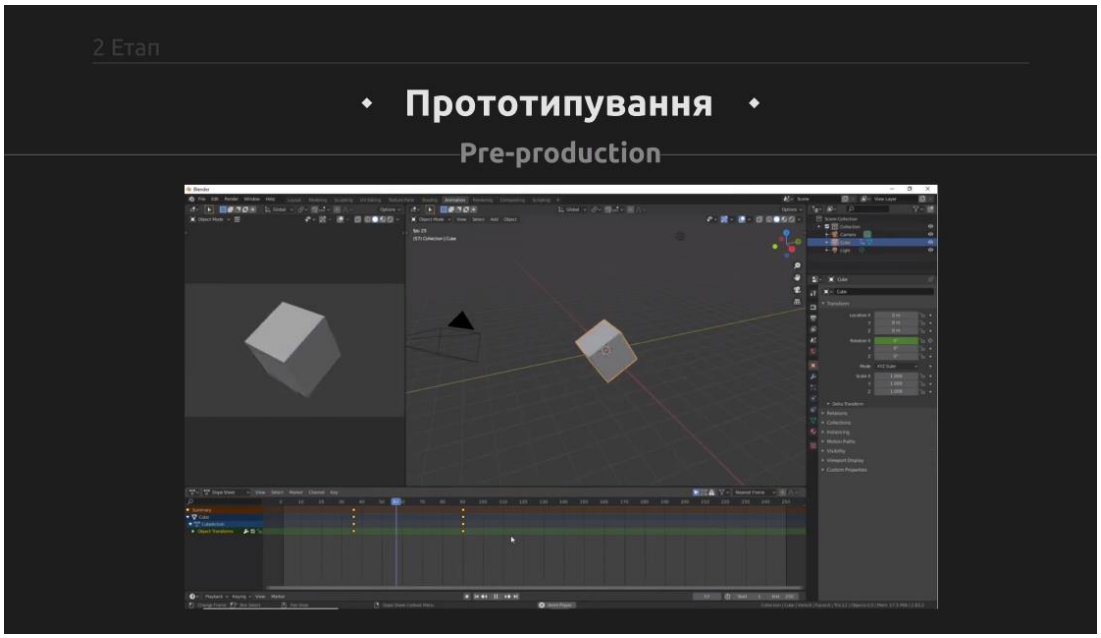


Рисунок 2.31 – Приклад анімації з використанням ключів

Реалістичну фізіологію людських рухів складно створити в такий спосіб вручну, тому для захоплення руху використовується спеціальний костюм motion capture, це костюм на якому розташовується велика кількість датчиків, а потім рух цих датчиків фіксуються камерами (рис.2.32).

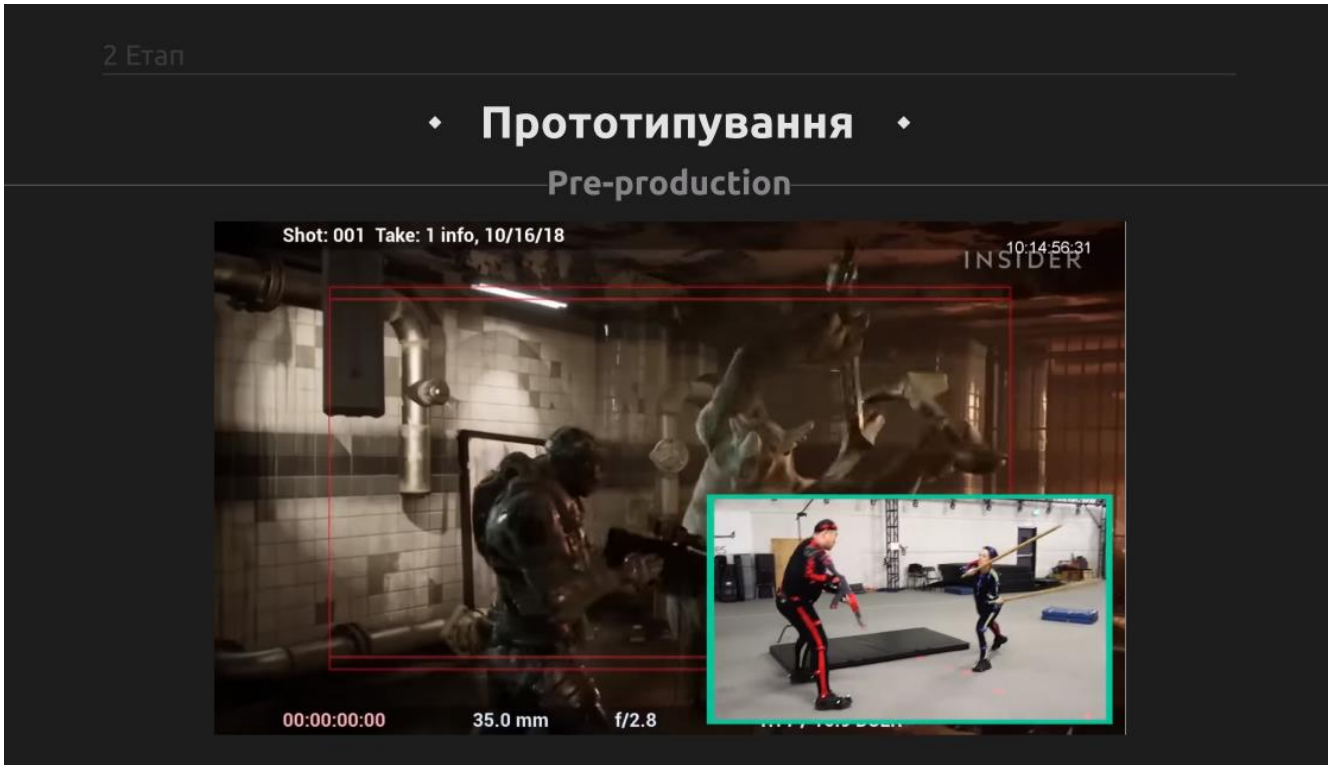


Рисунок 2.32 – Приклад використання спеціальних костюмів для анімації

4 основні анімації, які створюють аніматори - бездіяльність, хода, атака та смерть. Ще для захоплення анімації може використовуватися новіша і дешевша технологія, вона ж без маркерної системи захоплення рухів, захоплення анімацій за допомогою відзнятого відео, тут вже працюють нейромережі.

Що ж до лицьової анімації, існують різні способи для застосування лицьової анімації, один з таких, це риг на основі морфів або Blend Shape. Створюється безліч копій однієї голови і потім на кожній змінюється емоція, наприклад, у однієї голови закритий рот, а в іншої відкритий, таким чином це дві ключові емоції і геометрія на проміжних кадрах автоматично підлаштовується за допомогою математичної інтерполяції, тут мінус у цьому, що цей вид клуні не можна застосовувати інших персонажах (рис.2.33).

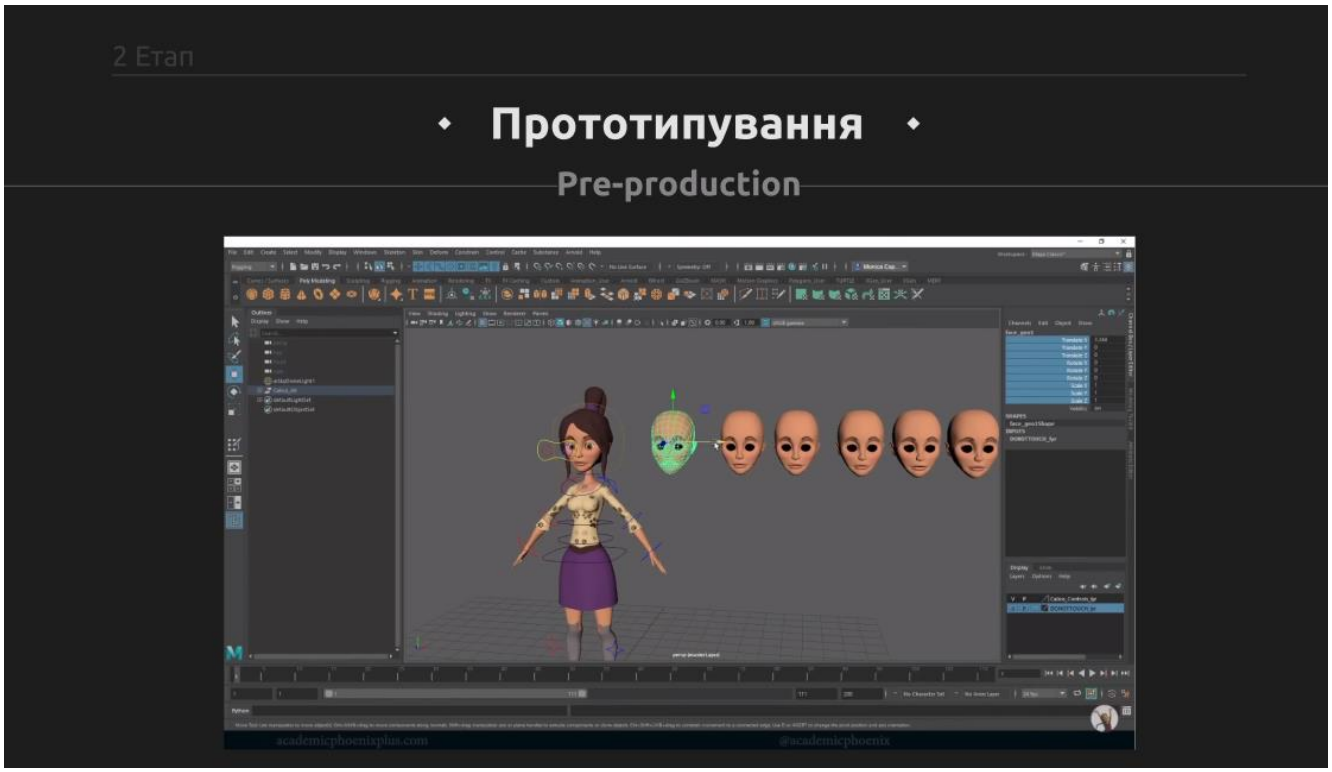


Рисунок 2.33 – Приклад лицьової анімації за допомогою Blend Shape

Набір таких копій виразів обличчя називають ключовими позами.

Інший метод застосування анімації, це ріг на основі кісток, він менш гнучкий, зате можна застосовувати до великої кількості персонажів.

Найбільш ефективний змішаний клуняк, що включає в себе елементи морфінгу і кісток. Тут аніматор має повний контроль за мімікою персонажа.

Створення того, що дозволить керувати анімаціями, зробити персонажа маріонеткою і називається ригом. Потім для зручності до нього можна додати контролери.

Подібну технологію FACS застосували у half-life 2, кожна частина особи має свій набір анімацією і щоб відтворити унікальну міміку, потрібно просто скомбінувати потрібні варіанти (рис.2.34).

За допомогою даної технології можна анімувати будь-якого персонажа і не потрібні дорогі технології Mock Up, але це всього лише способи дозволяють анімувати, а де ж брати самі анімації, тобто дані, це можна робити вручну за ключовими кадрами, за допомогою звукових файлів це здійснюється здопомогою нейронної мережі.

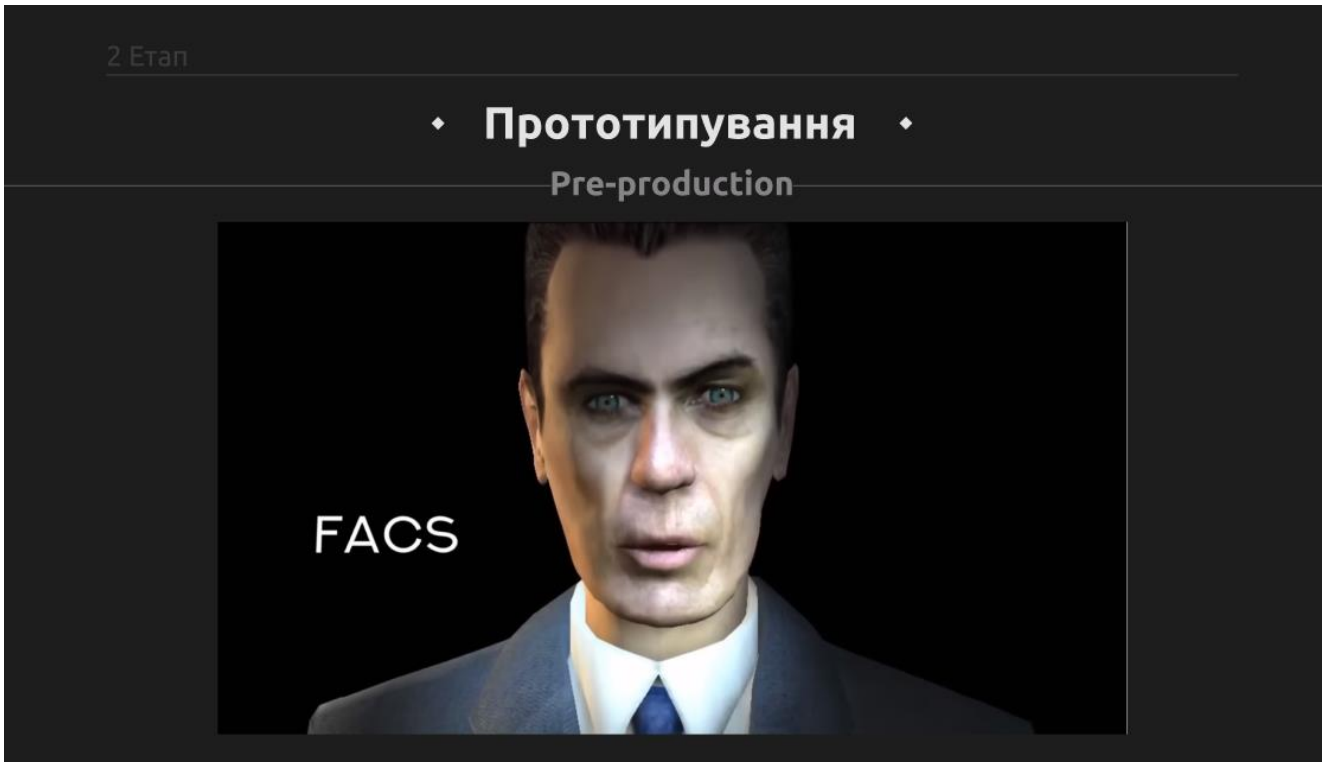


Рисунок 2.34 – Приклад використання технології FACS для анімації

Інший спосіб - це маркерна система захоплення лицьової анімації Performance Capture. У ролі маркерів можуть виступа ти кульки наклеєні на обличчя, фарба або крапки намальований фломастером (рис.2.35).

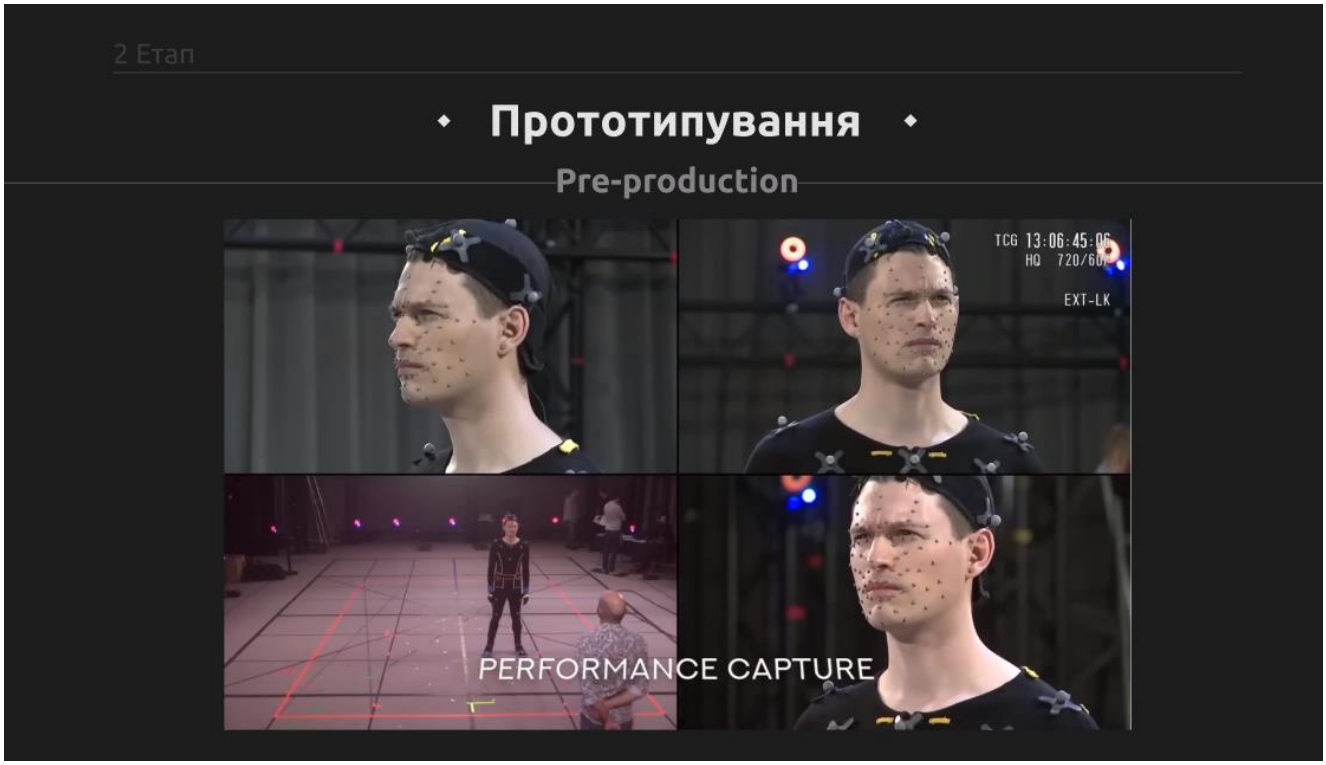


Рисунок 2.35 – Система захоплення лицьової анімації Performance Capture

Взагалі лицьова анімація, це дороге задоволення, але програма MetaHuman покликана прискорити та здешевити цей процес. У ній можна створити персонажа немов у якійсь rpg грі та додати йому анімацію за допомогою смартфона камери (рис.2.36).

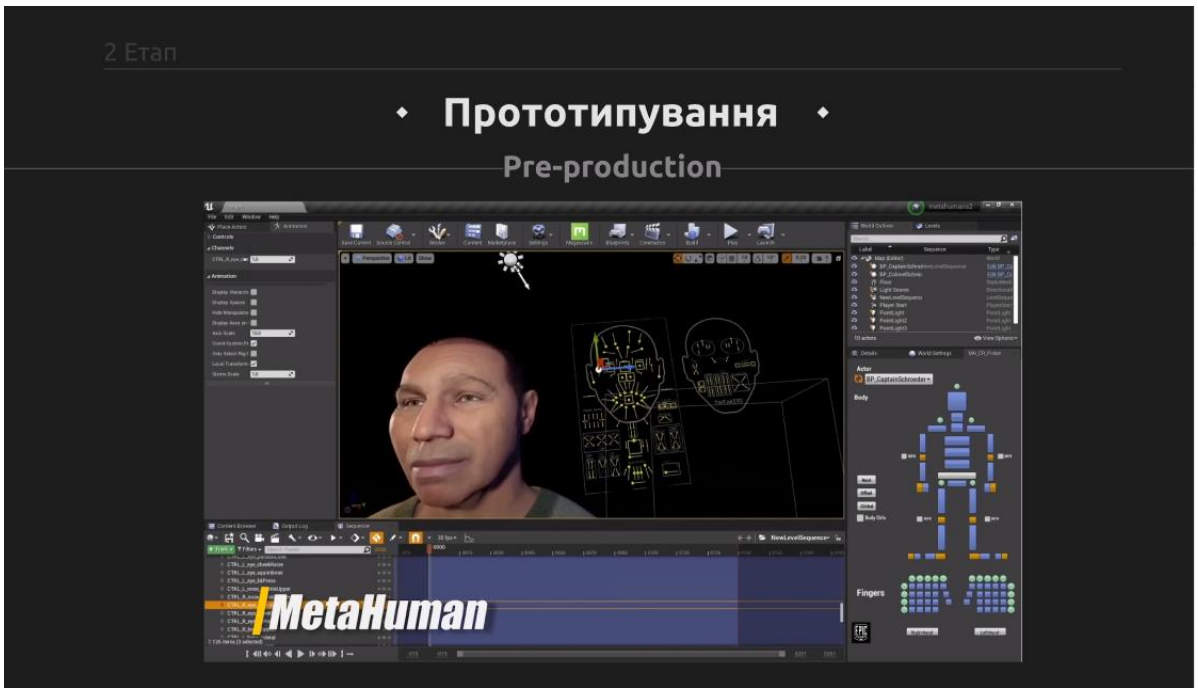


Рисунок 2.36 – Прорама для лицьової анімації MetaHuman

Самого персонажа можна зробити в спеціальному редакторі на кшталт Character Creator, а модель обличчя навіть можна зробити з фотографії (рис.2.37).

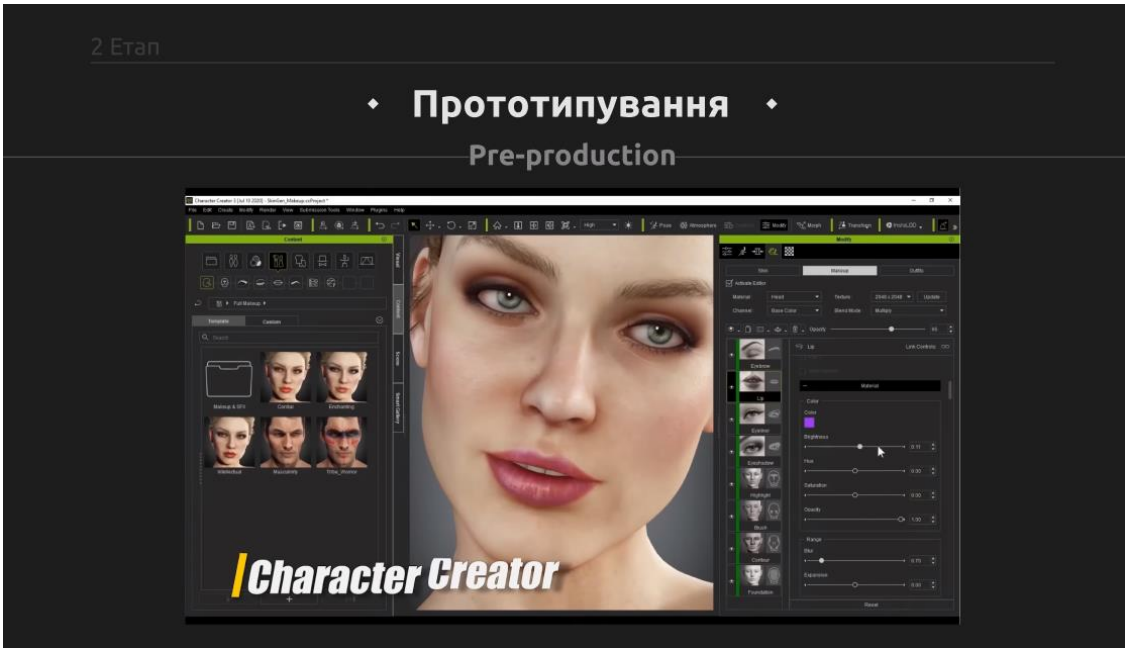


Рисунок 2.37 – Створення персонажа у редакторі Character Creator

10. Налаштування у Рушії

На завершення готового персонажа імпортують у рушій і вже там починається робота з його налаштування, потрібно прив'язати анімації до дій, просто запрограмувати, прив'язати персонажа до штучному інтелекту чи додати діалоги. На додаток до всього цього, є просунутий вид анімації, це процедурна вона ж динамічної анімації, весь рух підлаштовується під оточення, це настроюється методами движка.

Усі ці етапи виконуються послідовно і після завершення кожного виконується перевірка.

2.3 Етапи створення ігрового рівня

Основні етапи створення ігрового рівня (рис.2.38):

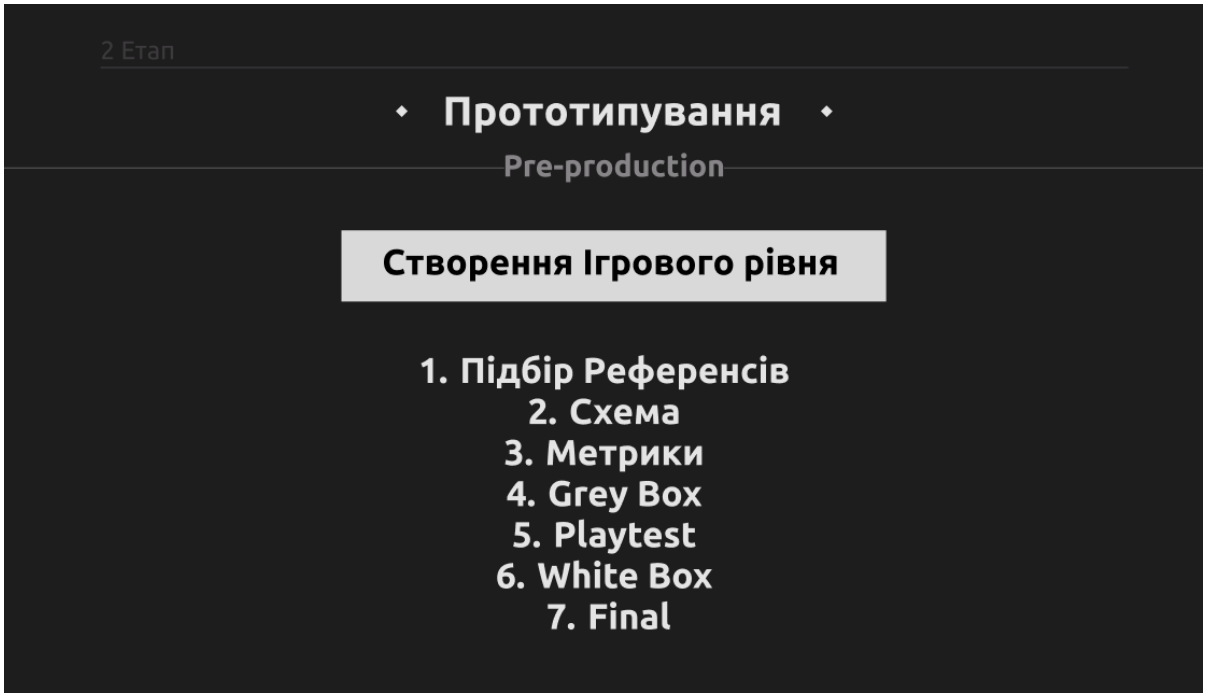


Рисунок 2.38 – Етапи створення ігрового рівня

Перший крок - це вибір референсів, це зображення для натхнення, або команда виїжджати в потрібну локацію і робить знімки (рис.2.39).

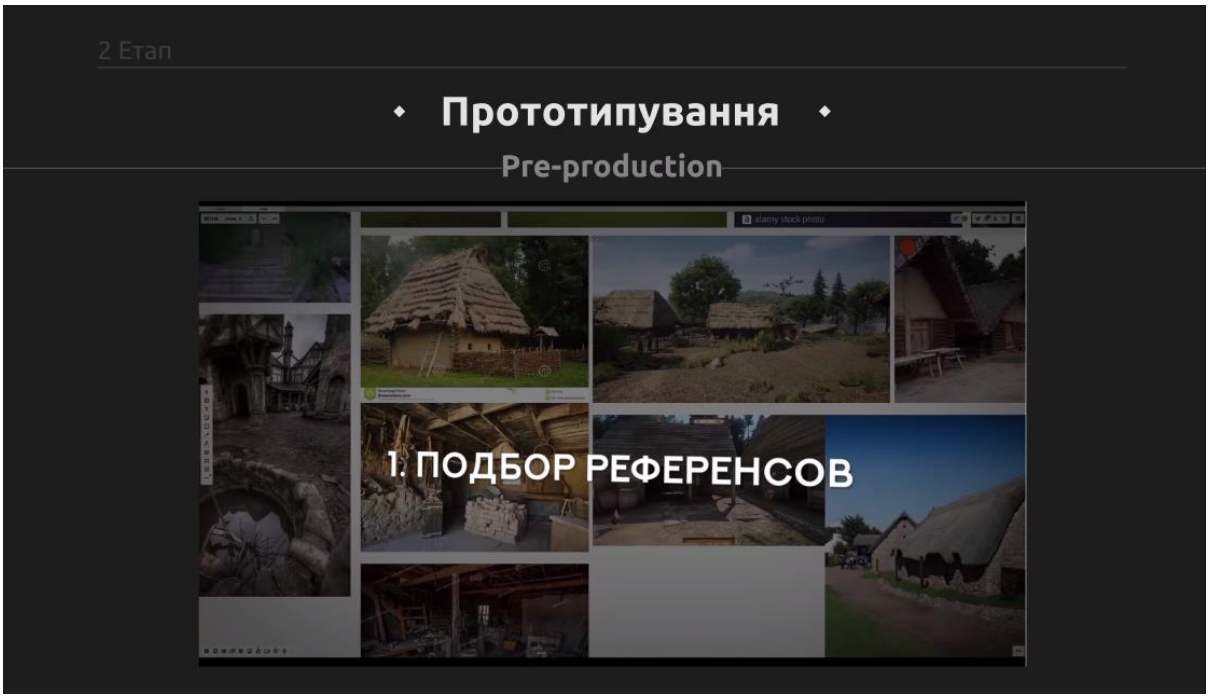


Рисунок 2.39 – Референси для створення ігрового рівня

Більшість рівнів починається з прототипу намальованого на папері, складається список локацій та відзначаються івенти, де скільки ворогів, які події прив'язані до точки на локації та інше (рис.2.40).

Рисунок 2.40 – Намальвані на папері прототипи

2.Схема

Зображення схкми пыд час прототипування гри (рис.2.41):

Рисунок 2.41 – Схема

У результаті карти на папері більше схожа на схему. Для ігор з одиночною кампанією може бути три види схем побудови рівнів (рис.2.42).

Рисунок 2.42 – Види схем побудови рівнів

Перший - Лінійний - тут гравець рухається з точки А в точку Б, зрідка відхиляюся від маршруту, але постійний рух уперед швидко набридне гравцю, тому дизайнер намагається ускладнити шлях.

Це робиться для того, щоб гравець у міру просування вивчав рівень, тим самим звертав увагу на деталі, це посилює занурення.

Бажано робити на рівні такі місця, де гравець міг відпочити від складної гри, тобто потрібно чергувати складні етапи і легші.

Другий вид – кластерний – багато відгалужень від центральних хабів. Такий вид рівнів часто використовують у іграх жанру Immersive Sim.

Третій вид – Павутина – багато сплетених між собою шляхів, це дає свободу гравцю. Більше підходить для ігор із відкритим світом.

У мультиплеєрних іграх левелів дизайнер працює над балансом рівня, унеможливлює перевагу однієї команди над іншою. Наприклад, щоб одна команда не змогла швидше захопити крапку, лише тому що їй швидше до неї дійти (рис.2.43).

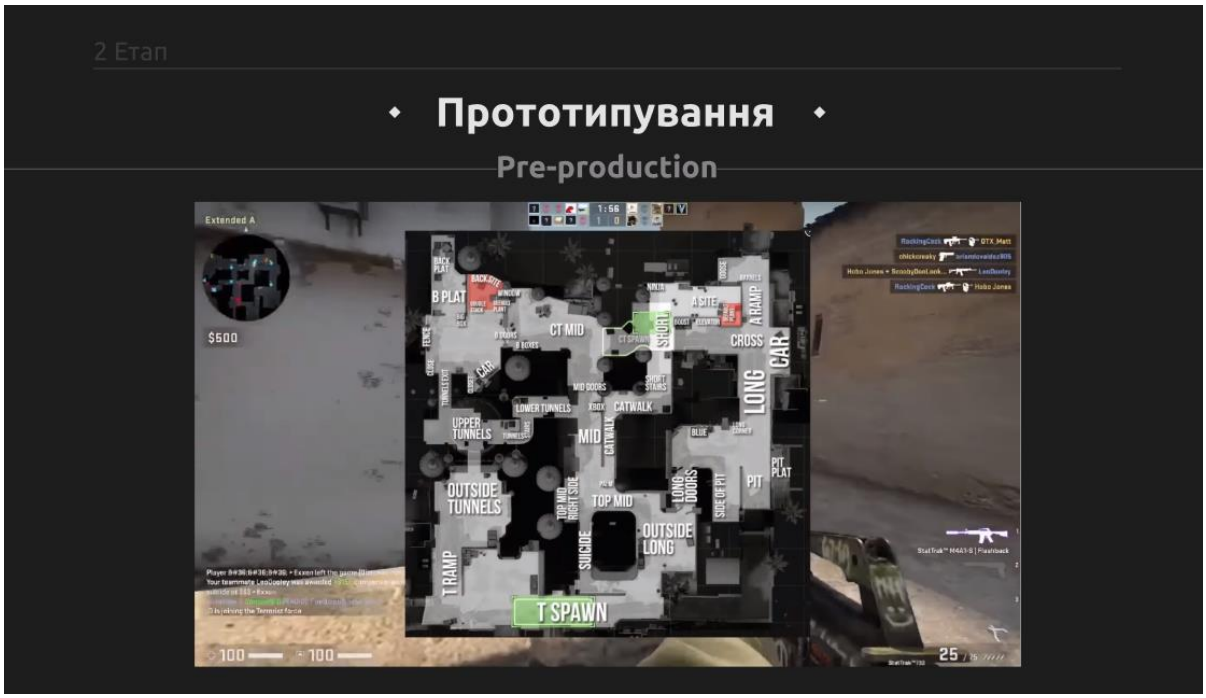


Рисунок 2.43 – Дизайн левелу у Мультиплеєрних іграх

3.Метрики

Після прототипування на папері, левелів дизайнер разом із гейм-дизайнером формують метрики, це шаблони розмірності для проектування рівнів, подібні дані надалі дозволять визначати, де гравець зможе пройти, сховатися чи піднятися (рис.2.44).

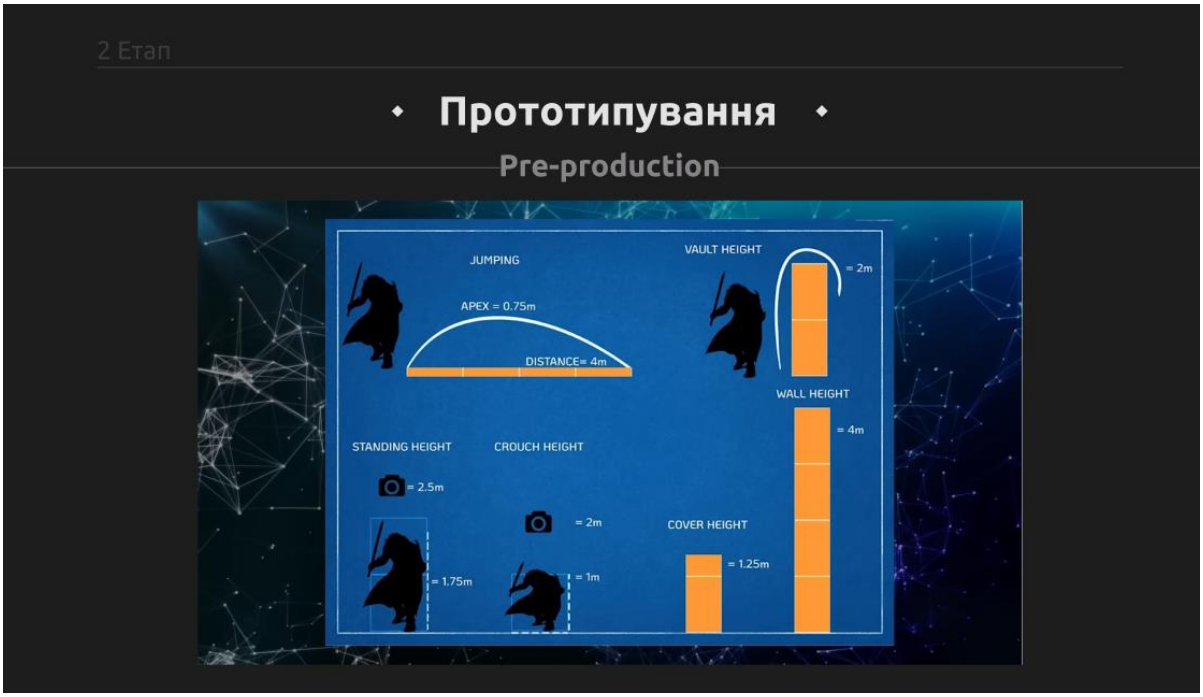


Рисунок 2.44 – Шаблон метрик на папері

4.Grey Box

Визначившись із метриками левелів дизайнер починає збирати Block Out і заразом виставляється просте освітлення. Block Out - це скелет карти, що складається, з примітивної геометрії.

Це може створюватися спеціальними інструментами всередині рушія, наприклад, як BSP Geometry в Unreal Engine. Тут перевага над подібною геометрією із 3d пакета полягає в тому, що її завжди можна відредагувати прямо у самому рушії (рис.2.45).

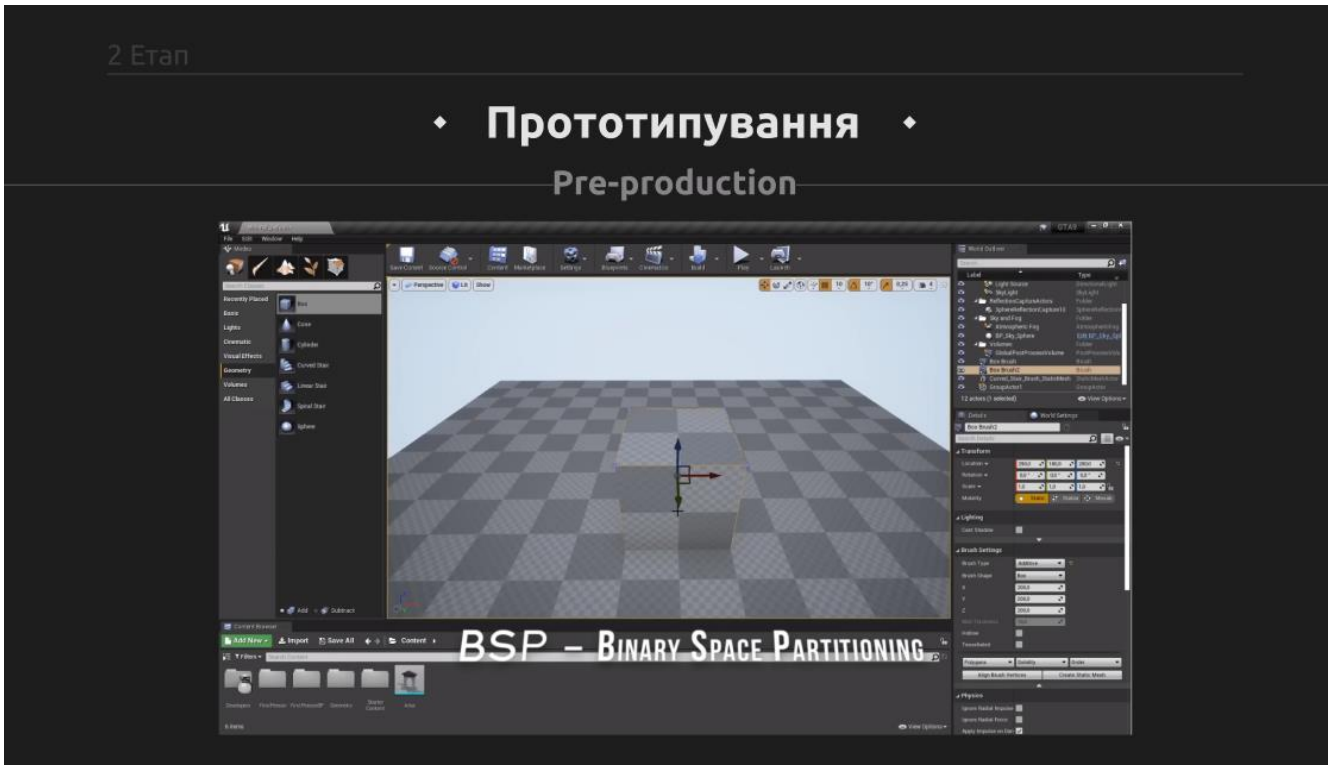


Рисунок 2.45 – BSP Geometry

Первинний Block Out називається Grey Box або Deigner Block Out. На даному макеті перевіряється геймплей без візуальних складових. Надалі ці примітиви будуть замінені на гарні 3d моделі (рис.2.46).

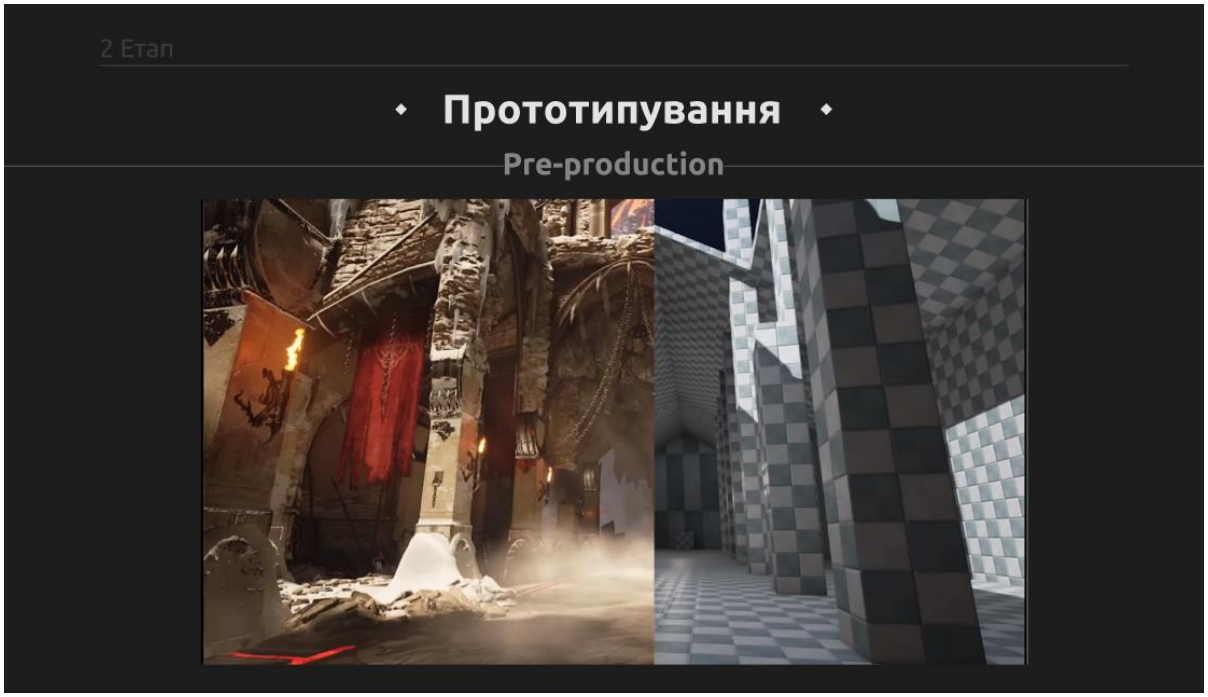


Рисунок 2.46 – Замінені 3d моделі

Ці болванки BSP можна конвертувати в Mesh, просто в просту 3d модель, а потім експортувати в 3d пакет для подальшої деталізації.

Якщо потрібно ландшафт, то його також можна створити за допомогою інструментів усередині рушія, або геометрію рівня можуть повністю імпортувати зі стороннього 3d пакета на зразок World Creater (рис.2.47).

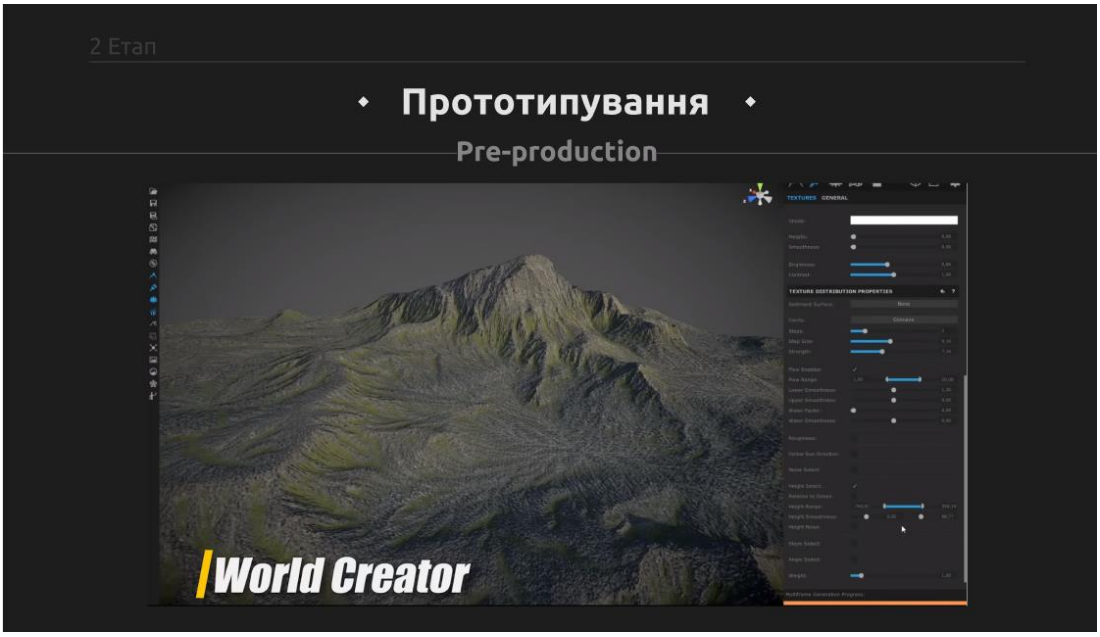


Рисунок 2.47 –3d пакет World Creater

Потім до рівня додаються тригери, це такі невидимі зони, які активують події (рис.2.48).

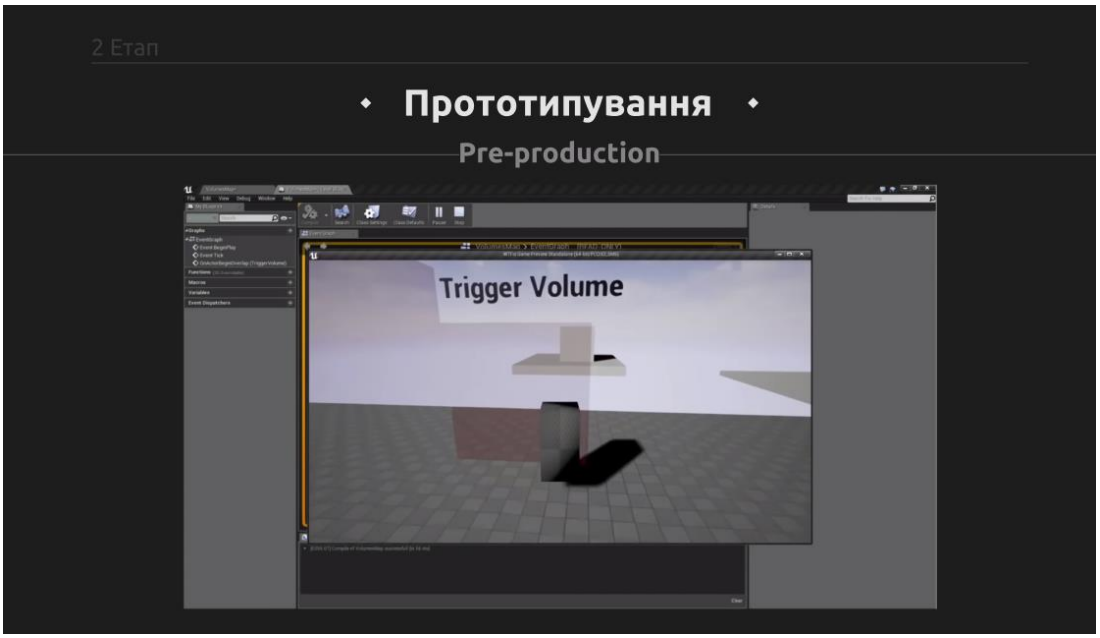


Рисунок 2.48 – Тригер звуку

Також на рівень додається середовище, в якому може орієнтуватися штучний інтелект, просто це об'єкт у вигляді невидимої зони, воно автоматично визначає про геометрію в собі, площини якими можуть переміщатися боти, щоб у подальшому штучний інтелект міг вибудовувати маршрути вздовж цих поверхонь (рис.2.49).

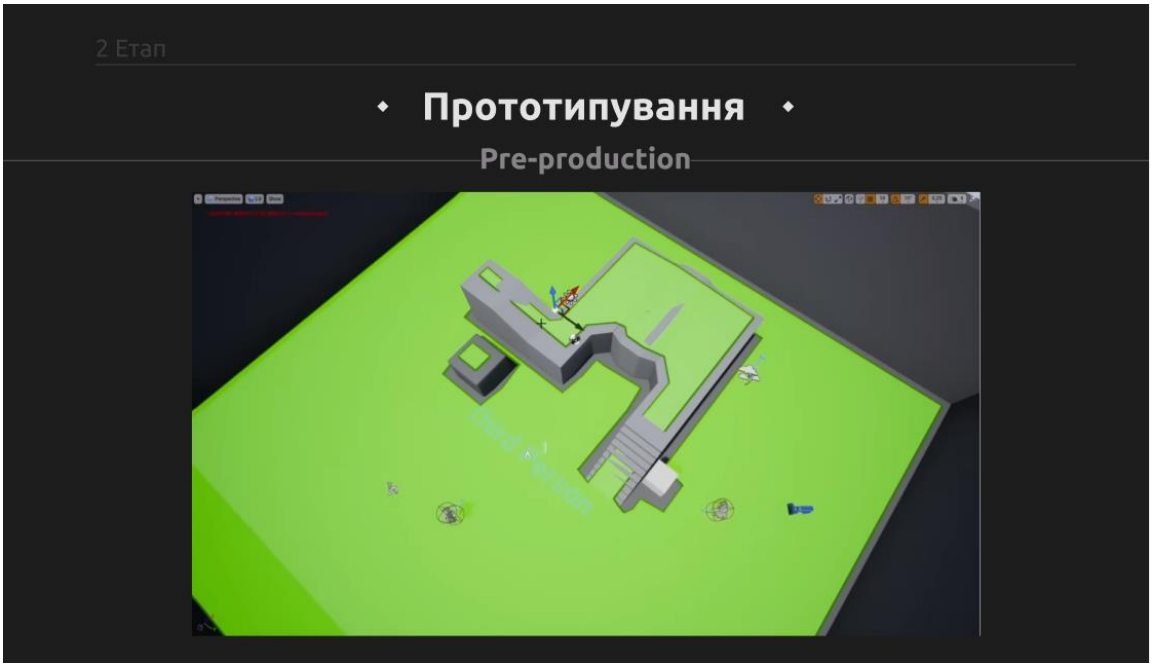


Рисунок 2.49 – Невидима зона для штучного інтелекту

Додаються точки появи персонажів - спавнери, але робиться це тільки для перевірки більше, точне налаштування робиться на фінальній стадії.

5.Playtest

Після компонування чорнового рівня рівня, грей боксу, проводиться геймплей тест і виправляються недоліки. Взагалі етап тестування починається з самого першого моменту розробки гри.

Після перевірок вся геометрія рівня заморожується.

6.White Box

Друга стадія прототипування рівня це white box, він же Art Block Out, у ньому художники опрацьовують художню частину рівня, роблять начерки, чорновий варіант (рис.2.50).

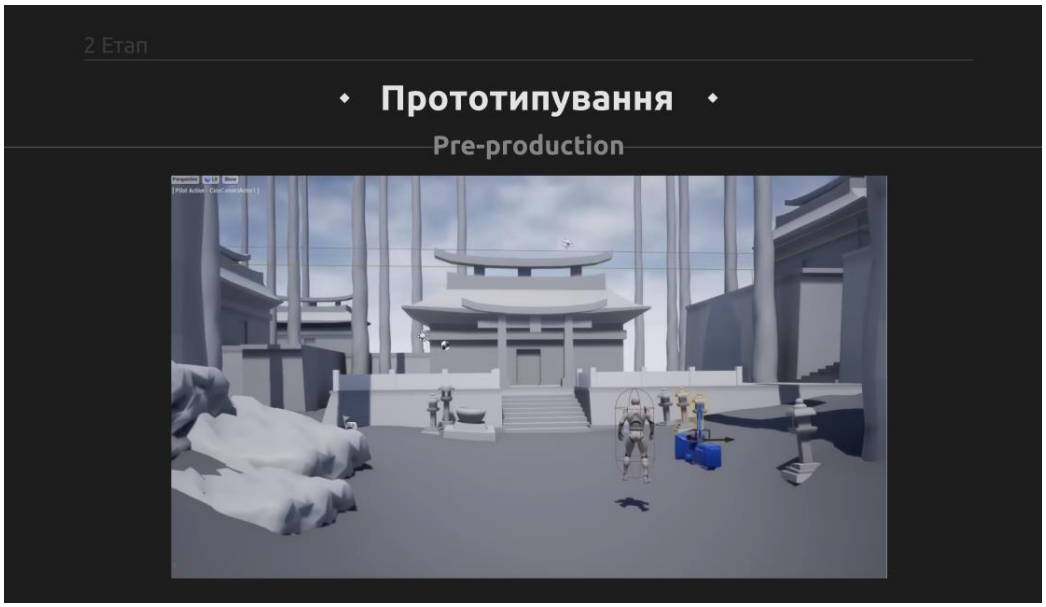


Рисунок 2.50 – Етап Art Block Out

На етапі white box до процесу підключаються художники по оточенню та концепт художники.

Художники по оточенню вже працюють над візуальною частиною рівня, вони заповнюють світ тимчасовими моделями placeholder-ами, щось середнє між болванками із Grey Box та кінцевим варіантом 3d моделі.

Якщо простіше висловлюватись, то художник створює щось з нуля, а дизайнер по-різному комбінує те, що створив художник, складає композицію.

На рівень додається чорнова ігрова логіка (рис.2.51).

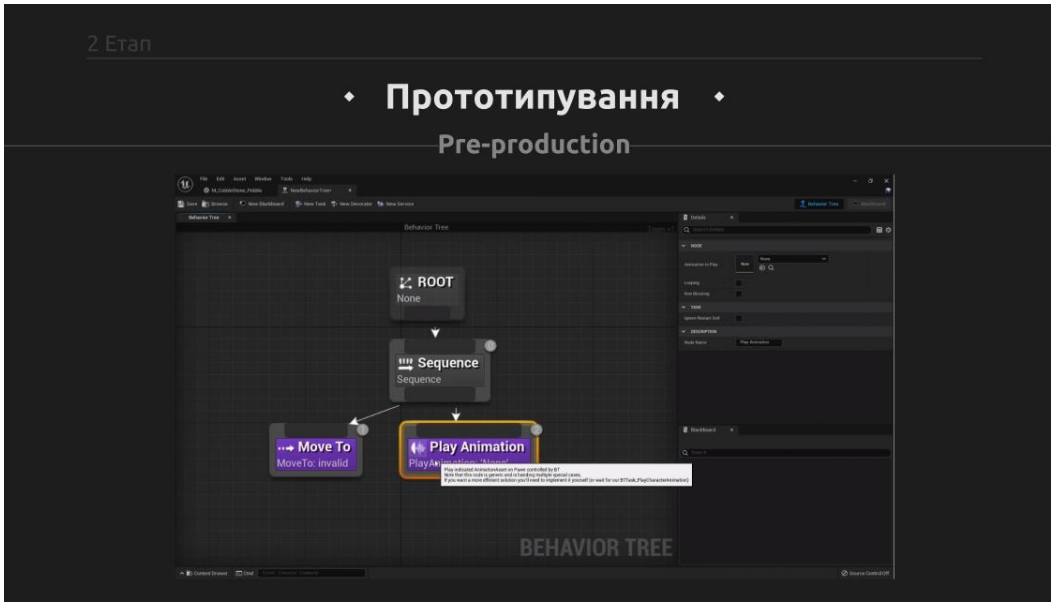


Рисунок 2.51 – Приклад додання логіки на рівень

Художники по оточенню можуть використовувати кіти, Kit – це такий набір ассетів, можна сказати однотипних. Кіт використовують як конструктор, складаючи композицію. Яскравий приклад – модульний будинок (рис.2.52).

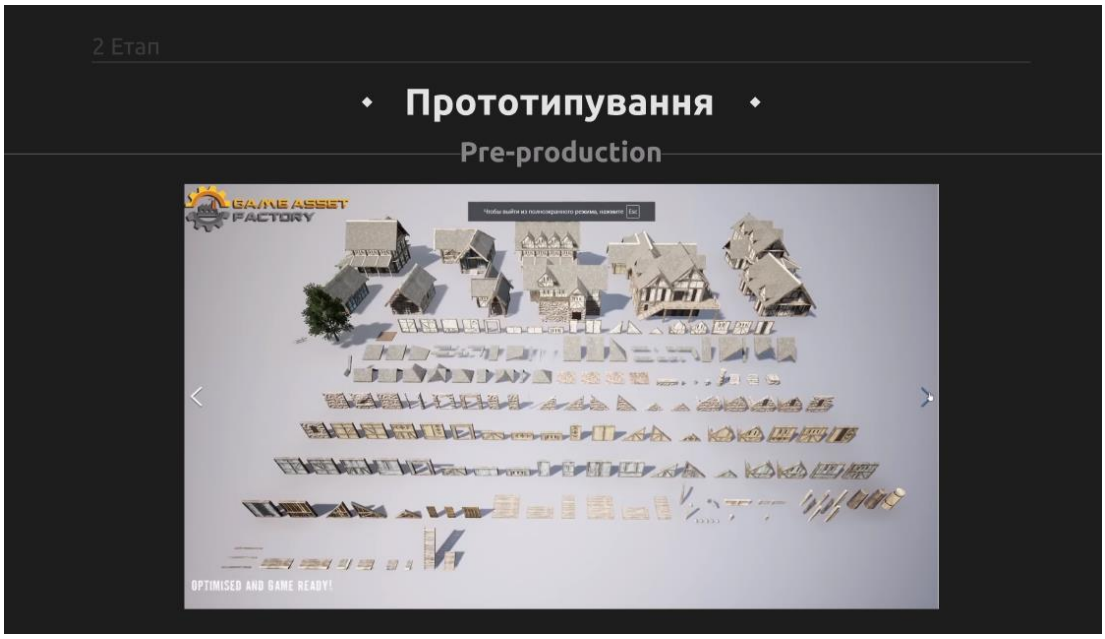


Рисунок 2.52 – Приклад набору ассетів

Якщо ігровий рушій дозволяє, то можна розділяти рівень на шари, наприклад, в одному міститься тимчасова геометрія, Block Out, в іншому художники працюють над оточенням, візуальна частина і вже в іншому шарі аудіо дизайнери працюють над звуковим оточенням (рис.2.53).

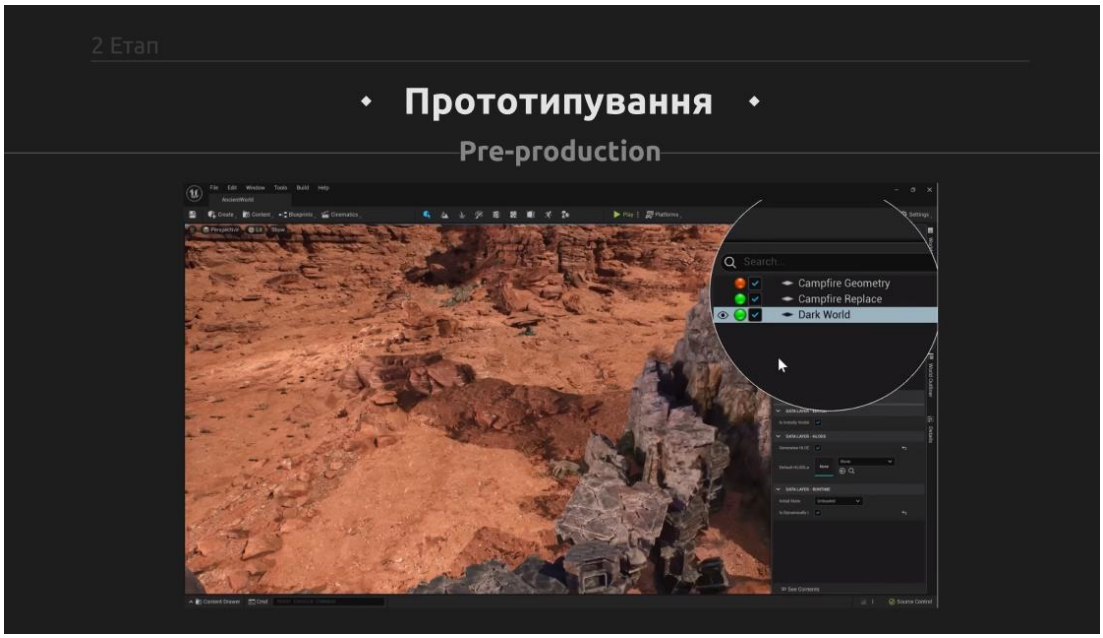


Рисунок 2.53 – Приклад розділення рівнів на шари

Коли рівень готовий, його знову перевіряє левел дизайнер.

7.Final

Потім настає третя, фінальна стадія, завершення. Тимчасовий placeholder замінюються на фінальні 3d моделі, видаляється практично вся BSP геометрії на догоду реалізму.

Художник по оточенню створює середні та дрібні об'єкти, труби, бочки та ящики і потім їх текстурує (рис.2.54).

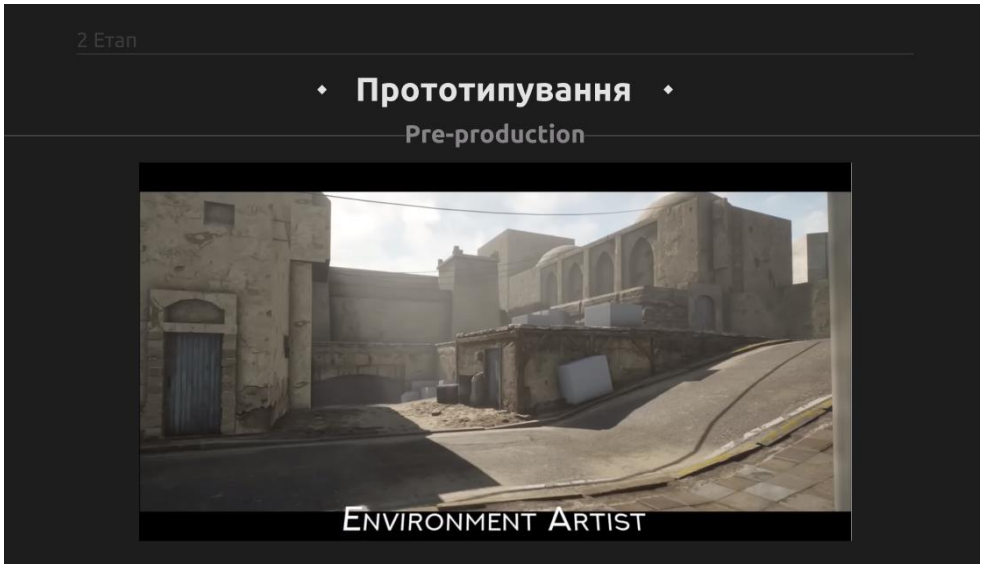


Рисунок 2.54 – Приклад роботи художника по оточенню

Дерева, трави та інші рослини можуть окремо створювати художники за рослинністю (рис.2.55).

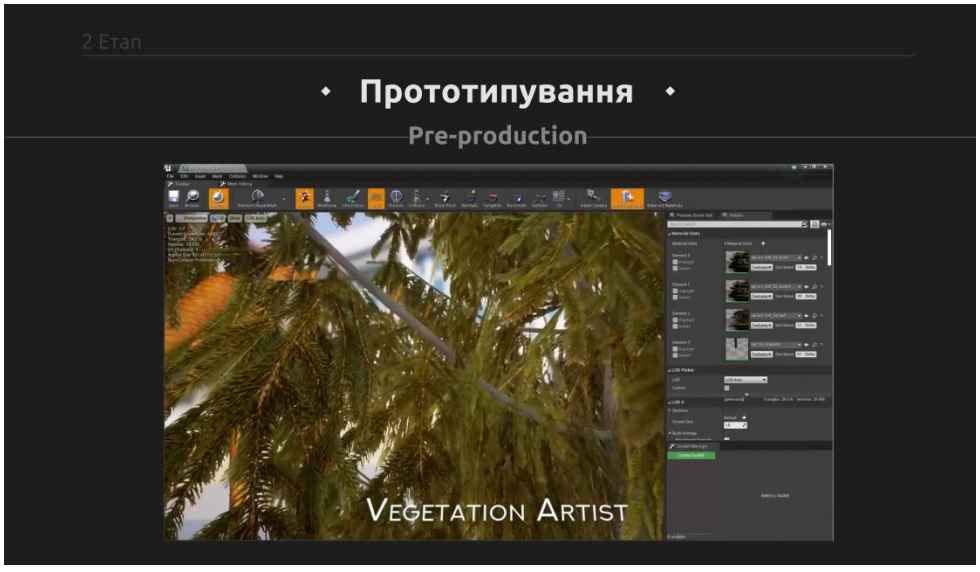


Рисунок 2.55 – Приклад роботи художника за рослинністю

Є ще така професія як левелев художник, він створює ландшафт, річки, дороги, гори, зазвичай всередині движка і наносить текстури на ландшафт і деталі, розставляє те, що зробили художники оточенню, ставить вази на тумбочки, заповнюють світ різними дрібницями, зелень розмазується ландшафтом інструментами движка, заразом налаштовує світло (рис.2.56).

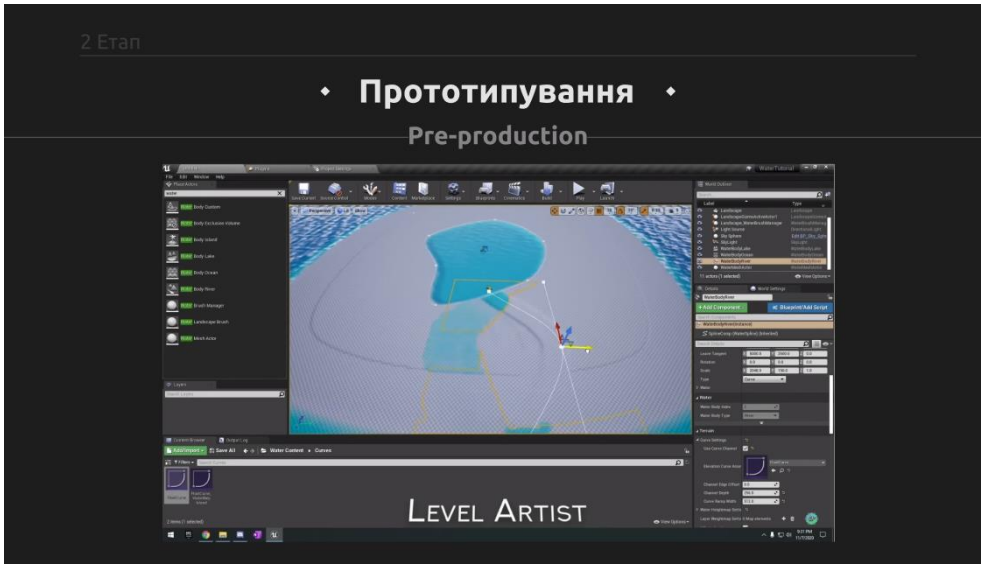


Рисунок 2.56 – Приклад роботи левелев художник

Level та Environment художники можуть допомагати або взаємозамінні один одного. Назва та обов'язки цих художників залежать від студії.

Розташуванням об'єктів також можна займатися левел дизайнер (рис.2.57, рис.2.58).

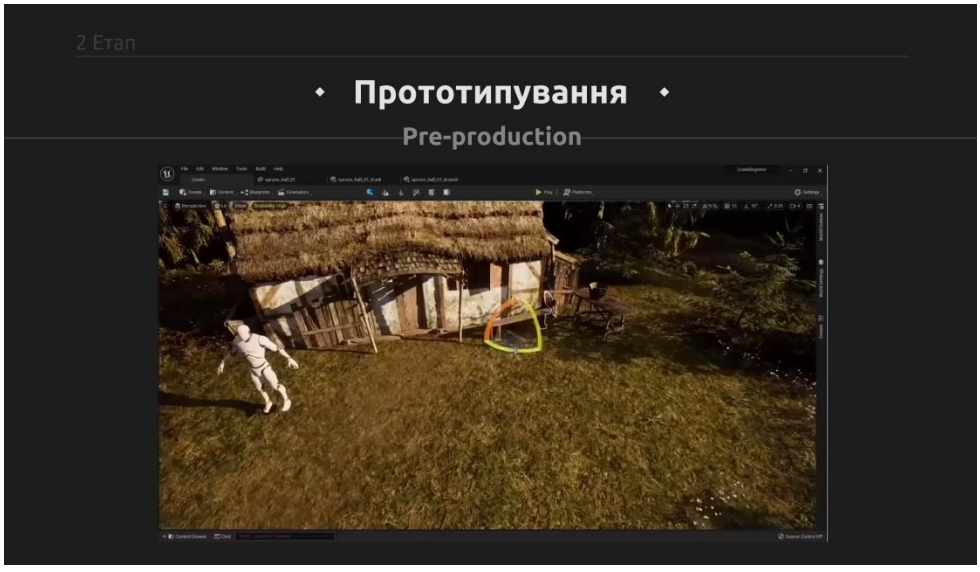


Рисунок 2.57 – Приклад розташування об'єктів левел дизайнером №1

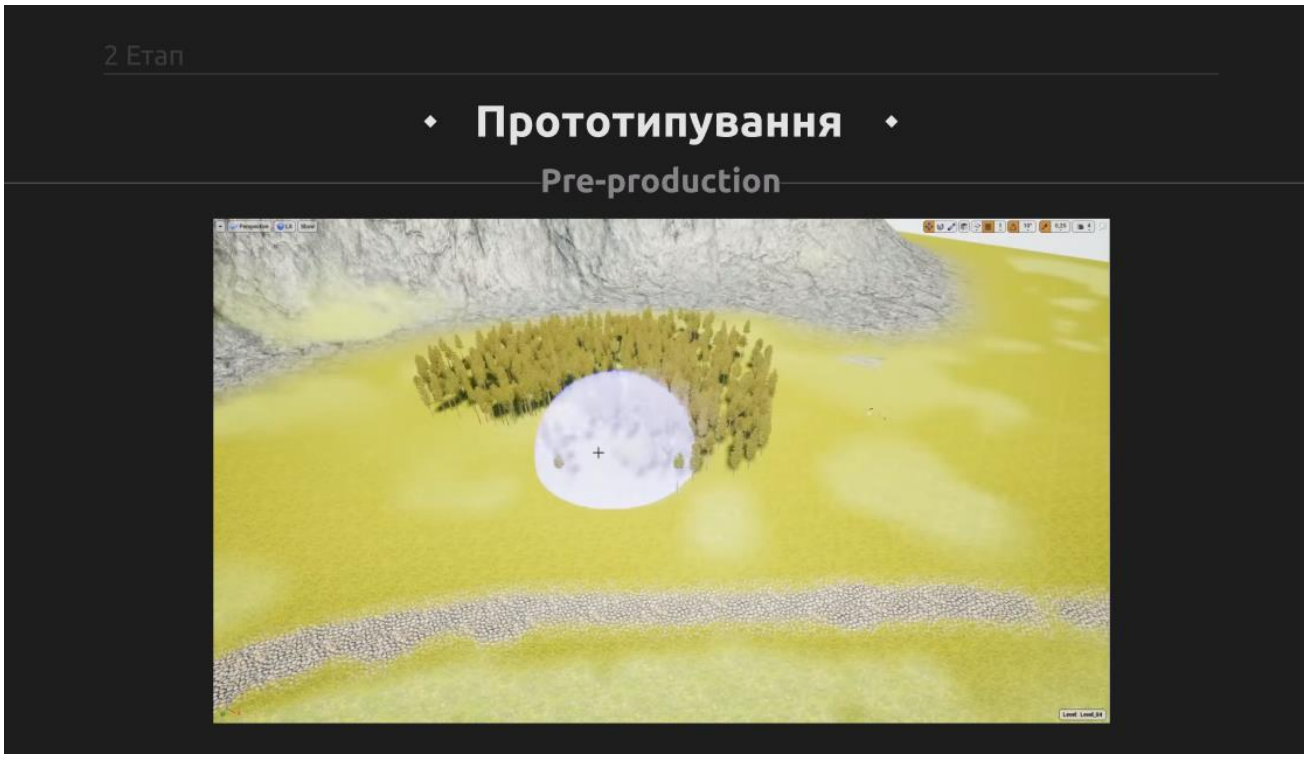


Рисунок 2.58 – Приклад розташування об'єктів левел дизайнером №2

Є така професія, як Наративний дизайнер, він займається розповіддю через оточення, робить рівень, скажімо так, живим, додає якісь композиції, записки, підказки, диктофони (рис.2.59).

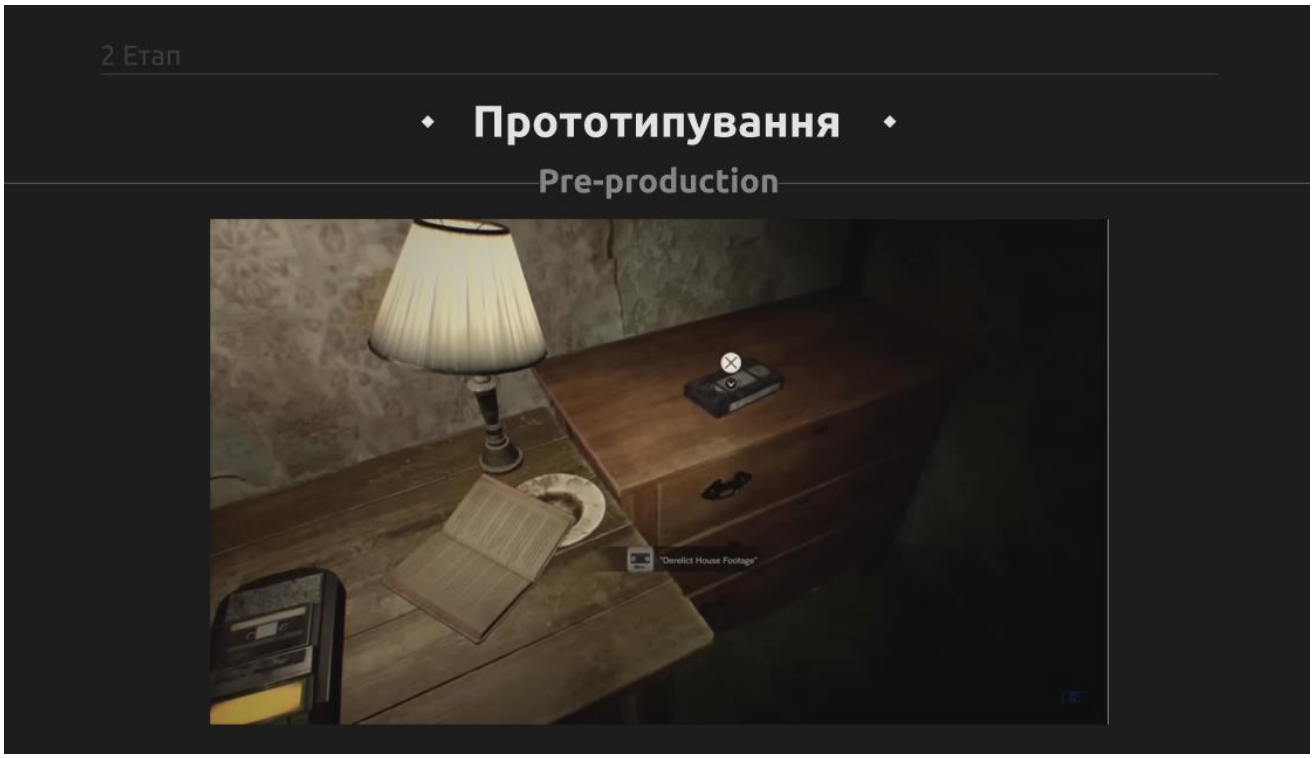


Рисунок 2.59 – Приклад роботи наративного дизайнера

Вода також робиться за допомогою інструментів рушія, найпростіший варіант це площина з текстурою, складніший - варіант тривимірний об'єм із змінною геометрією поверхні (рис.2.60).

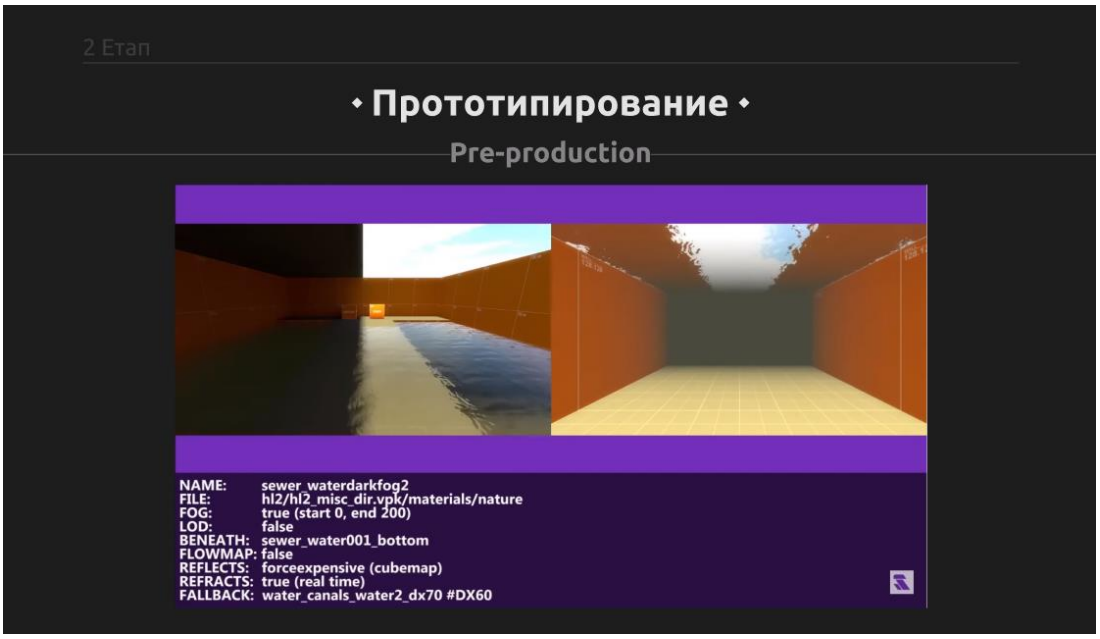


Рисунок 2.60 – Простий варінт створення води у рушії

У сучасних іграх, гравець фізично може впливати на цю поверхню, створюється це через шейдери. звичайного вода це тривимірний об'єм, зона в просторі, коли гравець потрапляє до неї, змінюється фізична модель і також постобробка (рис.2.61).

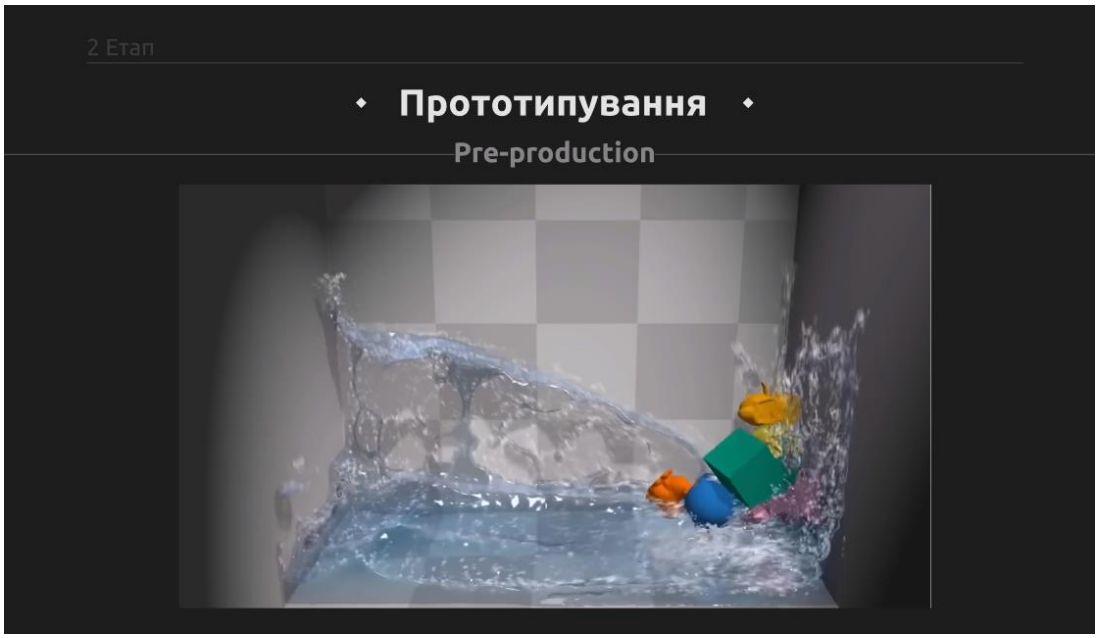


Рисунок 2.61 – Скдладний варінт створення води у рушії

Найбільш просунутий варіант води і, звичайно ж, ресурси витратний, це вода на частинках. Поки що її найбільше використовують на кінематографі.

На рівнях для мультиплеєра усувається візуальний шум, тобто робиться так, щоб гравці краще помічали один одного на рівні, тому дрібні деталі на кшталт сміття зсуваються до стін (рис.2.62).

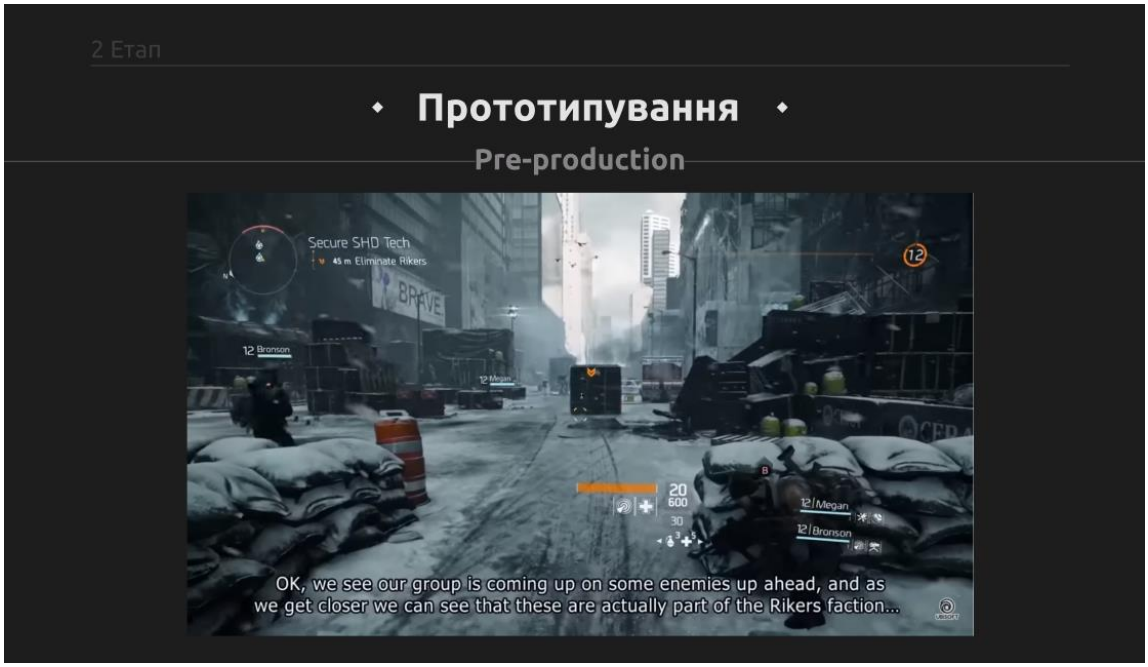


Рисунок 2.62 – Приклад усування візуального шуму

Висвітленням може займатися окрема людина Lighting Artist (рис.2.63).

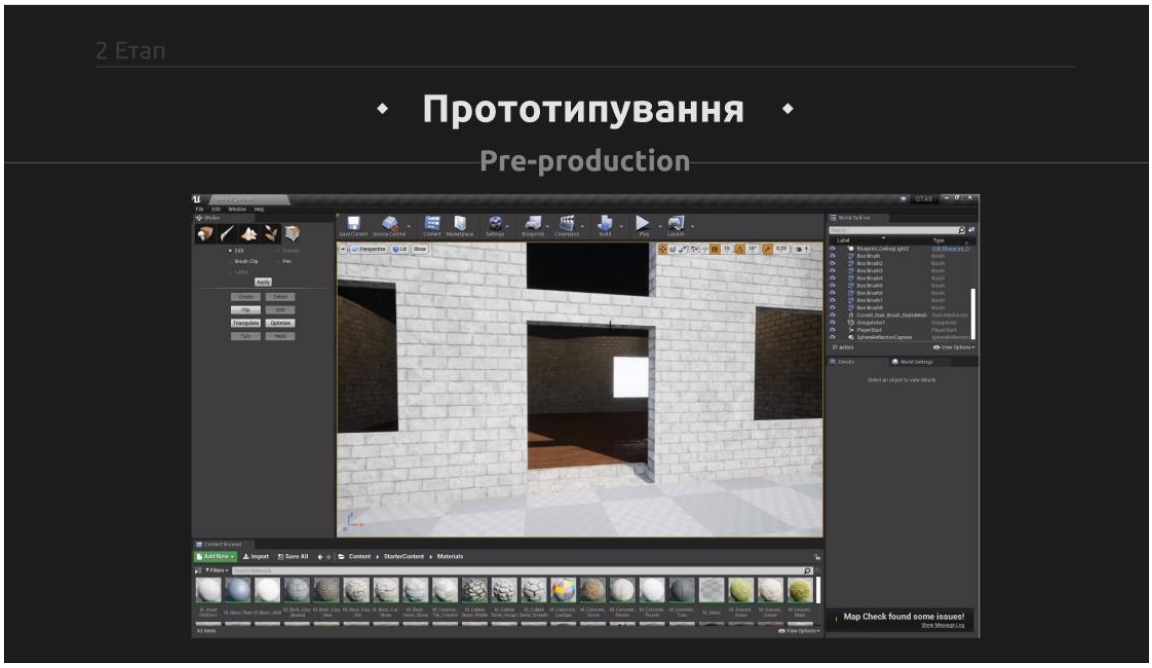


Рисунок 2.63 – Приклад роботи зі світлом у рушії

Світло може заздалегідь розраховуватись і запікатися у спеціальні light map (рис.2.64).

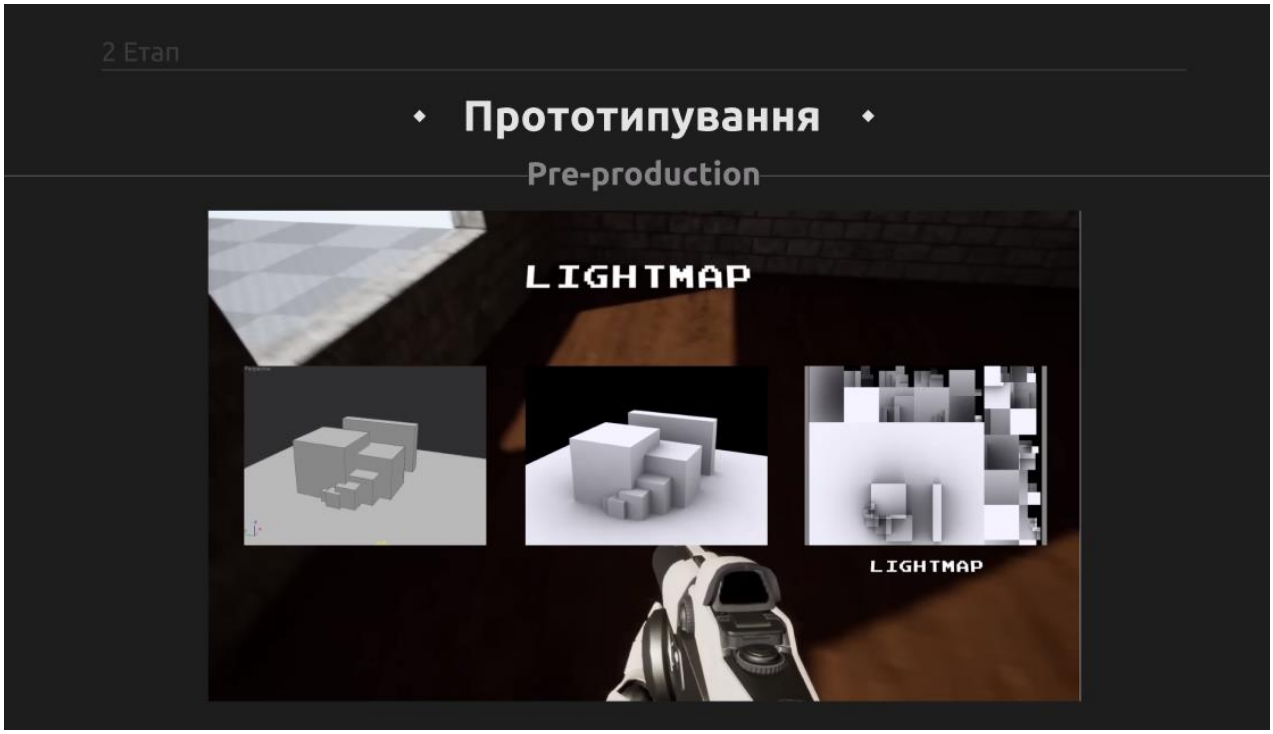


Рисунок 2.64 – Запікання світла у light map

Або бути динамічним, що працює в реальному часі, це методи Ray tracing (рис.2.65).

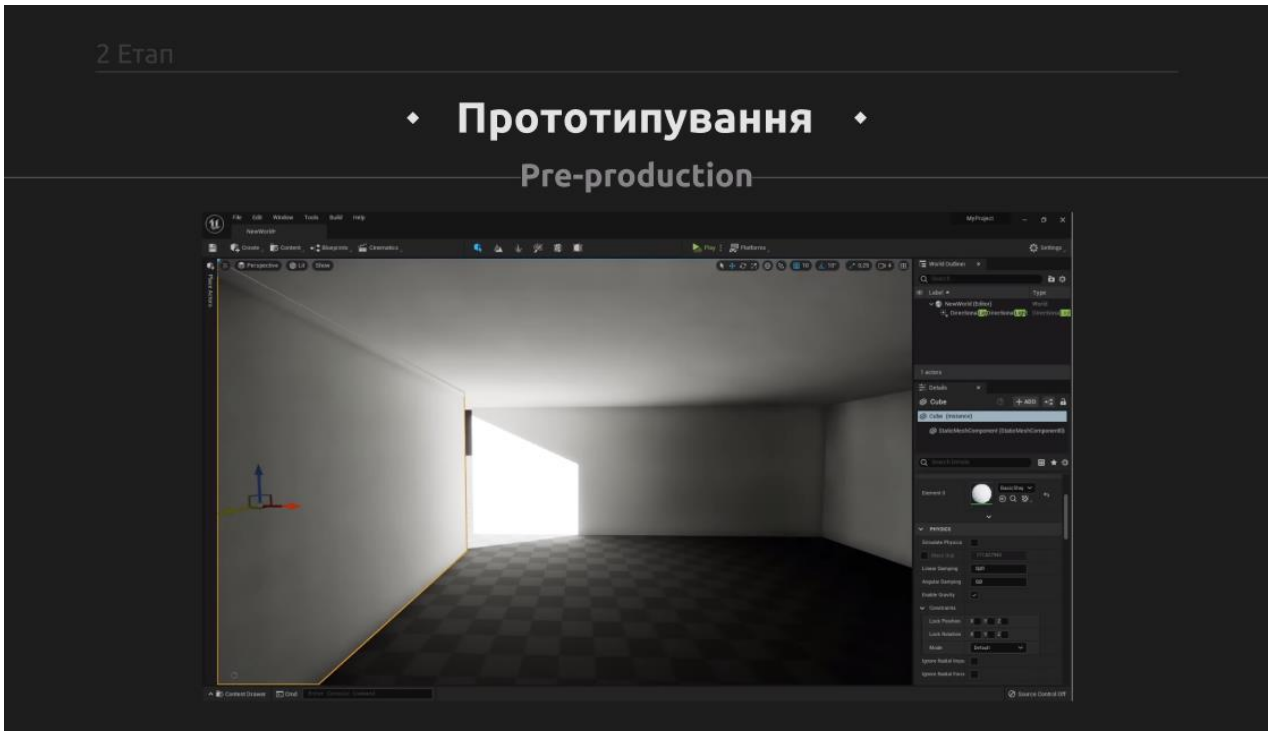


Рисунок 2.65 – Динамічне світло Ray tracing

Також є гібридні методи. З відображеннями так само, як і з освітленням, вони можуть заздалегідь запікатися у спеціальні карти Cubemap (рис.2.66).

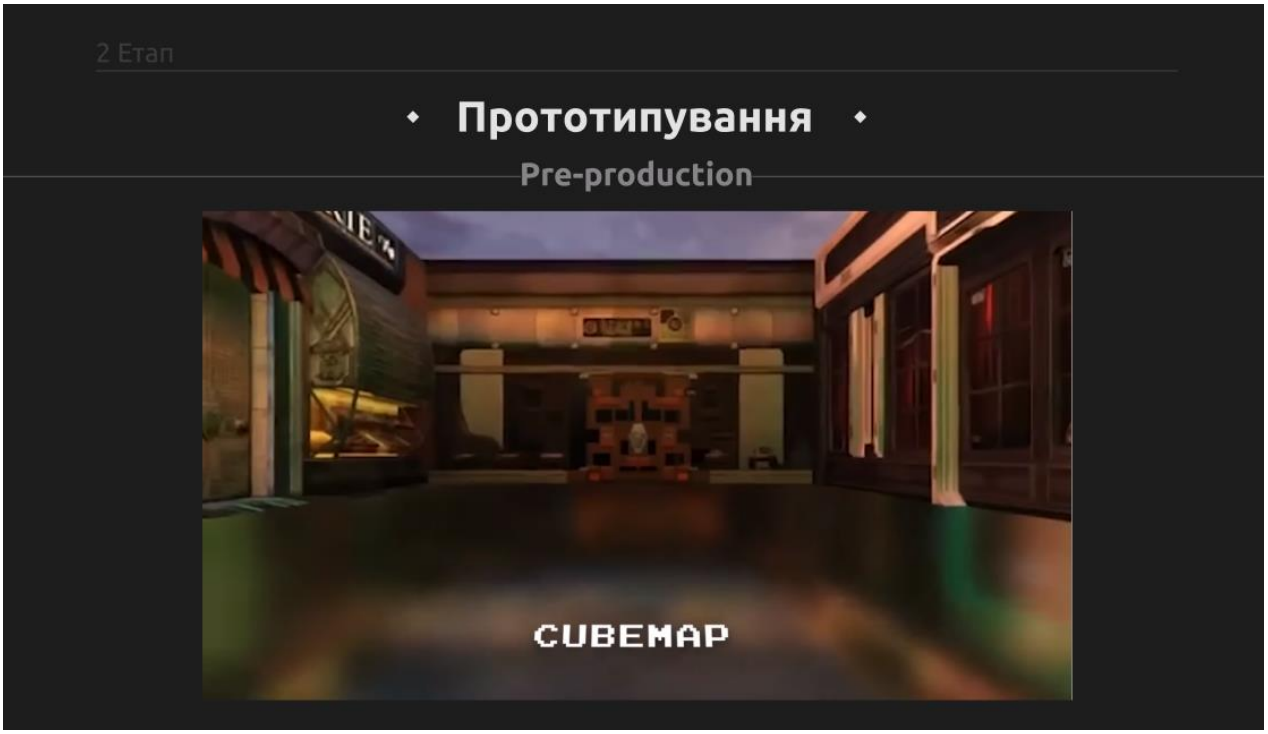


Рисунок 2.66 – Гібібридний метод створення світла у карти Cubemap

Технологія SSR найпоширеніша на даний момент, це відображення в реальному часі, але тільки в полі зору (рис.2.67).

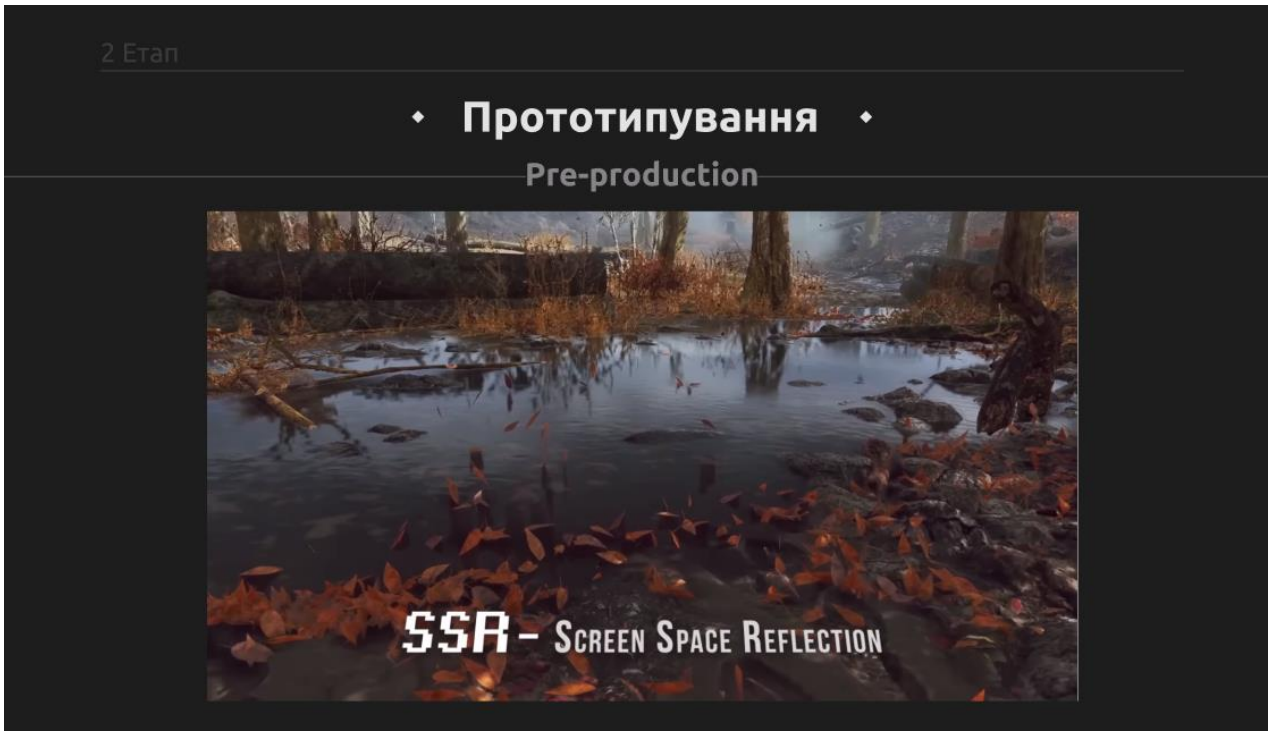


Рисунок 2.67 – Технологія SSR

Найпросунутіший спосіб це технологія Ray tracing та Ray Casting (рис.2.68).

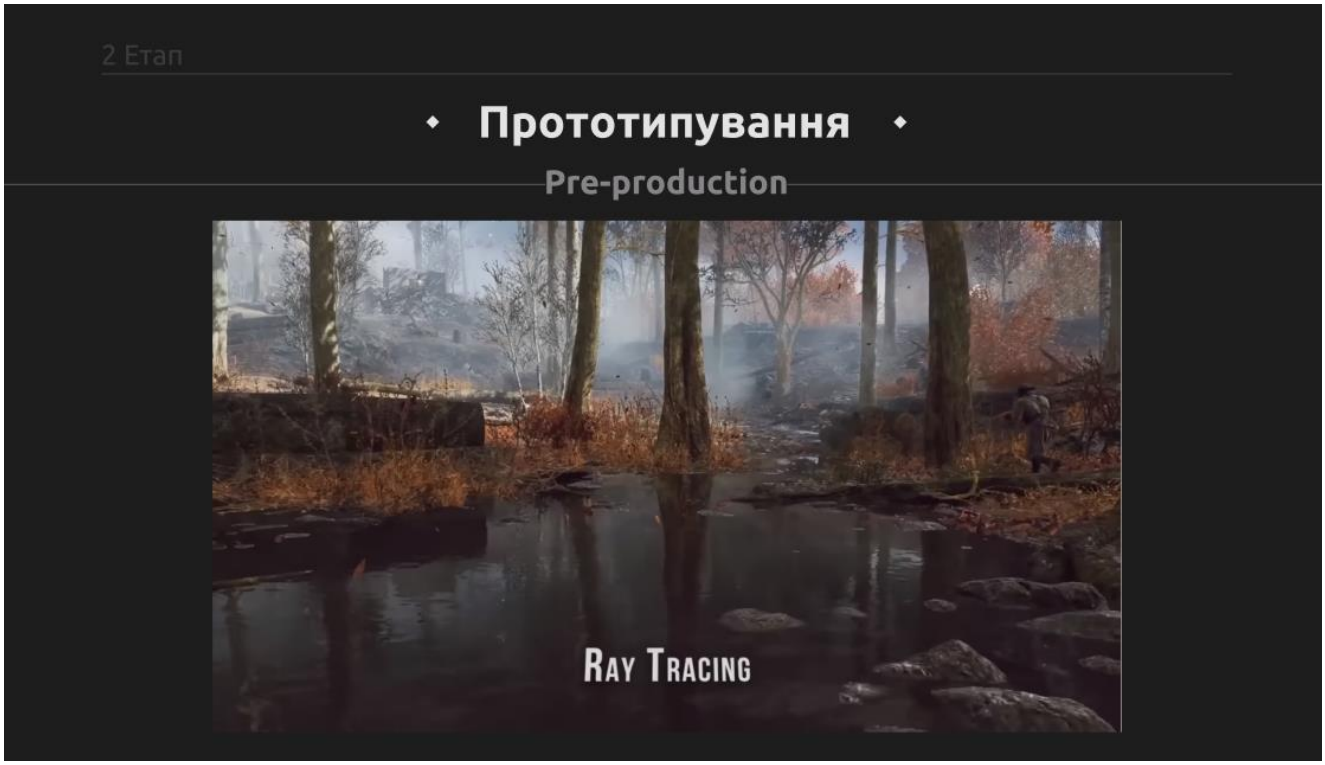


Рисунок 2.68 – Ray tracing

Небо в іграх це просто величезна сфера з текстурою, також ця сфера може виконувати роль глобального освітлення, що світить всередину себе (рис.2.69).

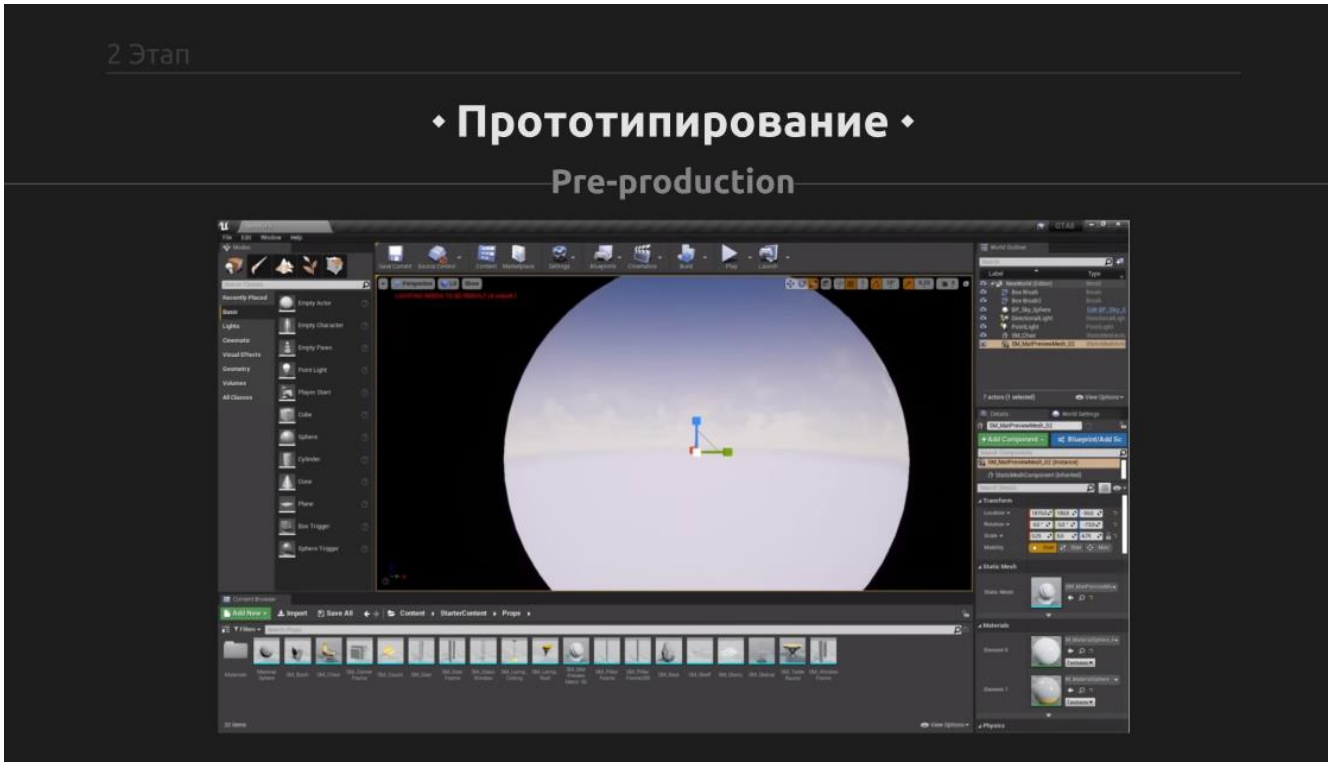


Рисунок 2.69 – Приклад створення неба у іграх

Фізика в іграх також працює завдяки ігровому движку, волосся на персонажах, тканини, самі персонажі називається це Rag doll (ганчір'яна лялька). Система руйнувань, фізичні частки та транспорт це все робиться засобами ігрового рушія (рис.2.70).

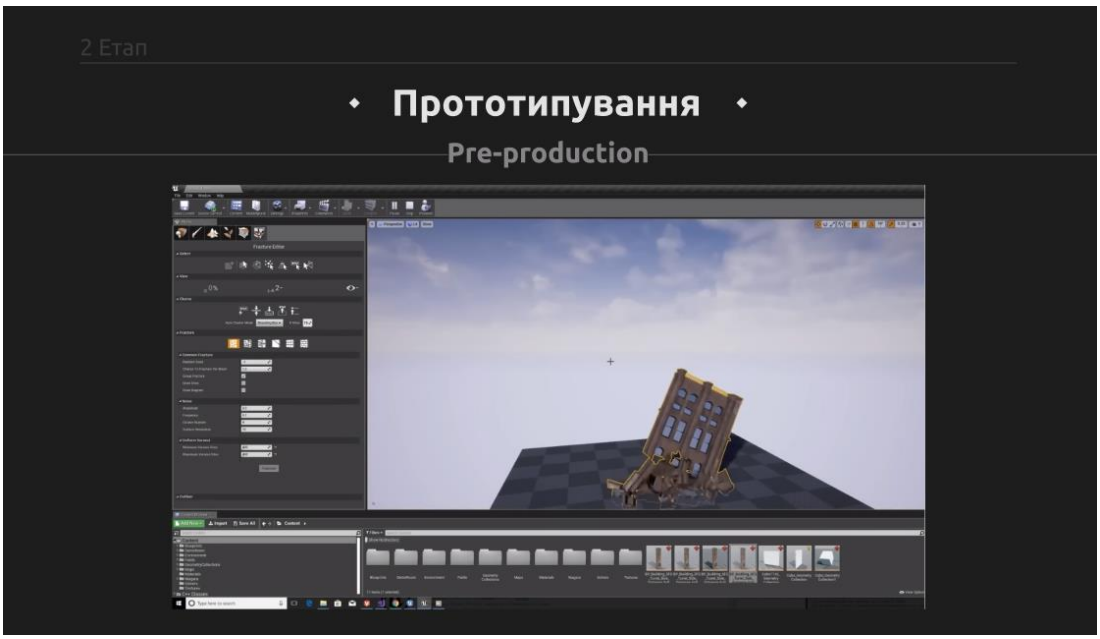


Рисунок 2.70 – Фізика в іграх (приклад руйнування будівлі)

Коли рівень готовий, його заморожують із цього моменту вже забороняється додавати на рівень нові пропси, тобто дрібні об'єкти, бочки та ящики, сміття, також видаляються непотрібні асети (рис.2.71).

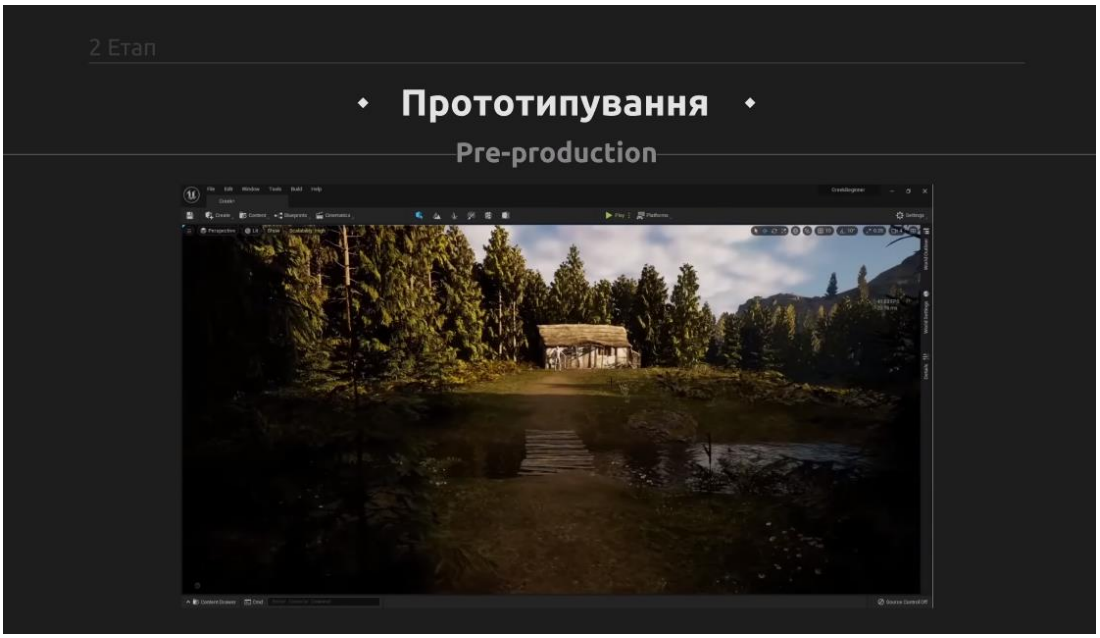


Рисунок 2.71 – Готовий заморожений рівень

На завершення роблять фінальний тест, проводиться оптимізація продуктивності, для цього використовується профайлінг, це інформація про об'єкти, скільки вони споживають ресурсів (рис.2.72).

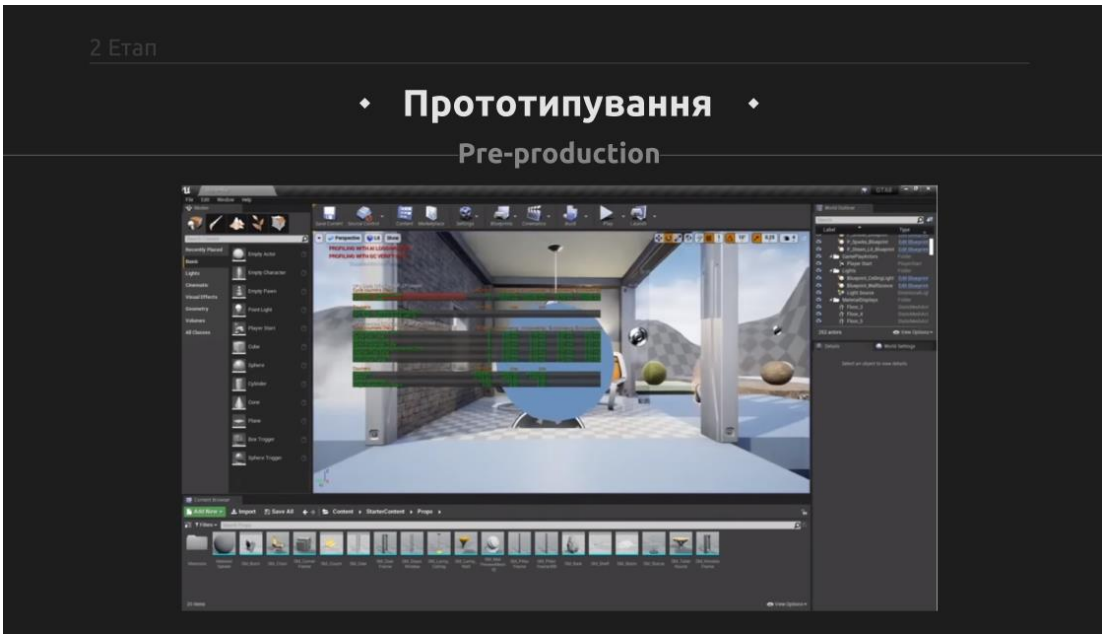


Рисунок 2.72 – Фінальний тест для оптимізації

Налаштовуються рівні деталізації - це коли при віддаленні від камери гравця у об'єктів знижується кількість полігонів, це потрібно для оптимізації.

Рівні формують навколишній світ гри, причому досить великий і потрібно його якось розділяти, щоб не перевантажувати пам'ять, один із старих і перших способів - це підвантаження рівнів, коли гравець проходить один, завантажуються інший це, просто окремі карти і тут треба чекати.

Один великий світ поділяється на фрагменти, чанки і завантажуються лише ті, які поблизу, щоб приховати вдалині чанки, розробники використовують хитрість, це туман чи таку хитрість, як пляшкова шийка - Occlusion culling. При рендерингу забирається все, що потрапляє поза увагою камери (рис.2.73).

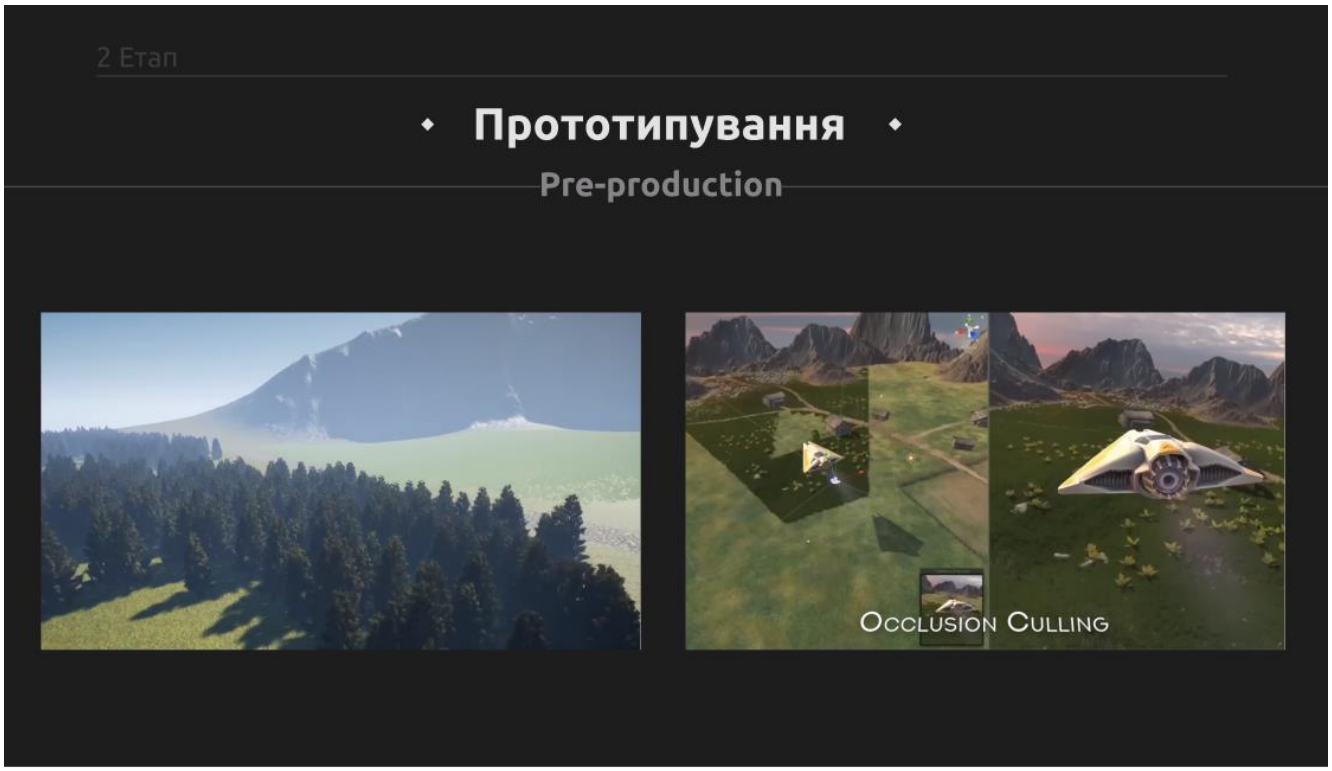


Рисунок 2.73 – Спосіб підвантаження рівнів Occlusion culling

Другий спосіб який частіше використовують у сучасних іграх - це підвантаження рівнів у реальному часі, яскравий приклад технологія world Composition з Unreal engine і її нова версія world partition (рис.2.74).

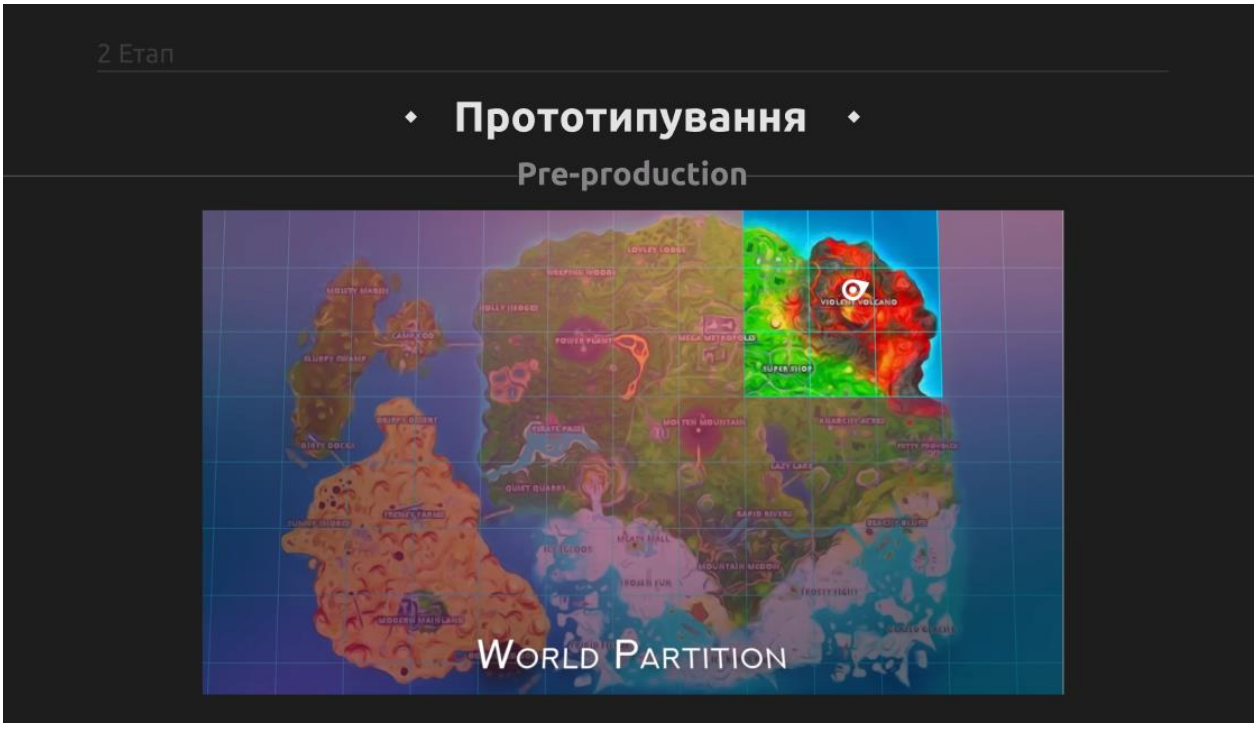


Рисунок 2.74 – Спосіб підвантаження рівнів World Partition

Або можна створити вузькі місця при переході з однієї локації на іншу і поки гравець протискується через неї, підвантажується наступний рівень і вивантажується попередній, це відбувається в реальному часу. Такі прийоми можна зустріти в серії ігор Tomb Raider (рис.2.75).

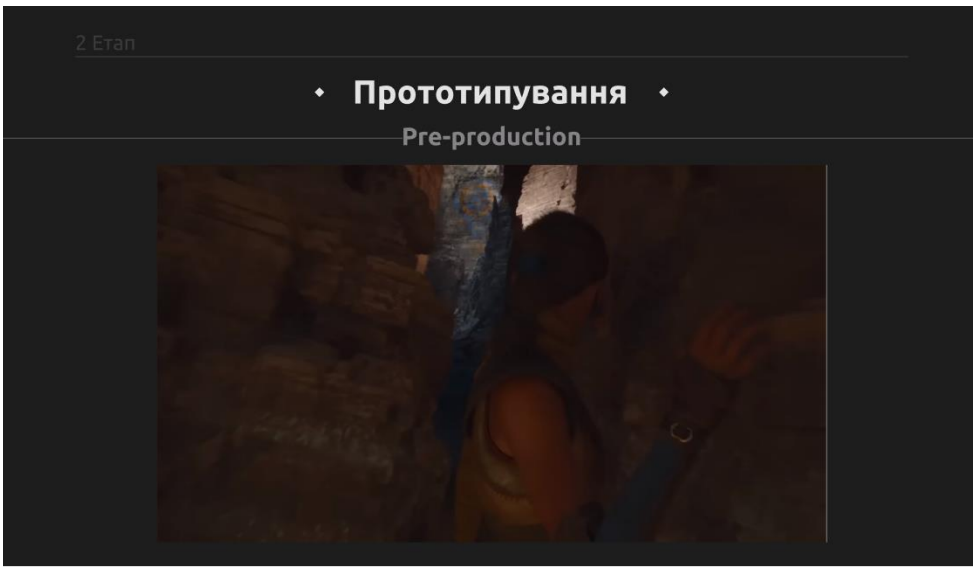


Рисунок 2.75 – Спосіб підвантаження рівнів LOD - level of details

І третій вид, який найкраще підходить для відкритих світів, ця система LOD - level of details. Гравець зверху здатний побачити весь світ, але щоб не навантажувати систему дрібні об'єкти зникають, а у великих знижується кількість полігонів і також роздільна здатність текстур. Таких рівнів деталізації може бути 3-4, а то й більше і в міру наближення деталізація збільшується (рис.2.76).

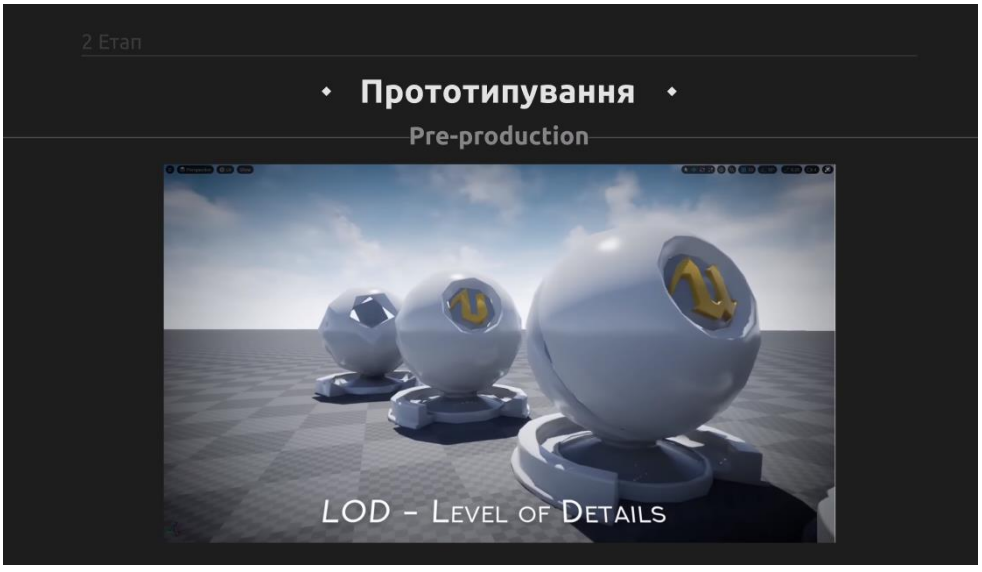


Рисунок 2.76 – Рівень деталізації з технологією LOD

2.4 Звук

Композитори пишуть музичний супровід, звукоінженер може збирати бібліотеки вже готові рішення або генерувати свої за допомогою ПЗ, можна записувати звуки у студії або команди виїжджає на полігон записує зброю або транспорту.

Актори озвучки записують голоси за сценарієм, потім аудіо-програміст підключає це все до гри (рис.2.77).

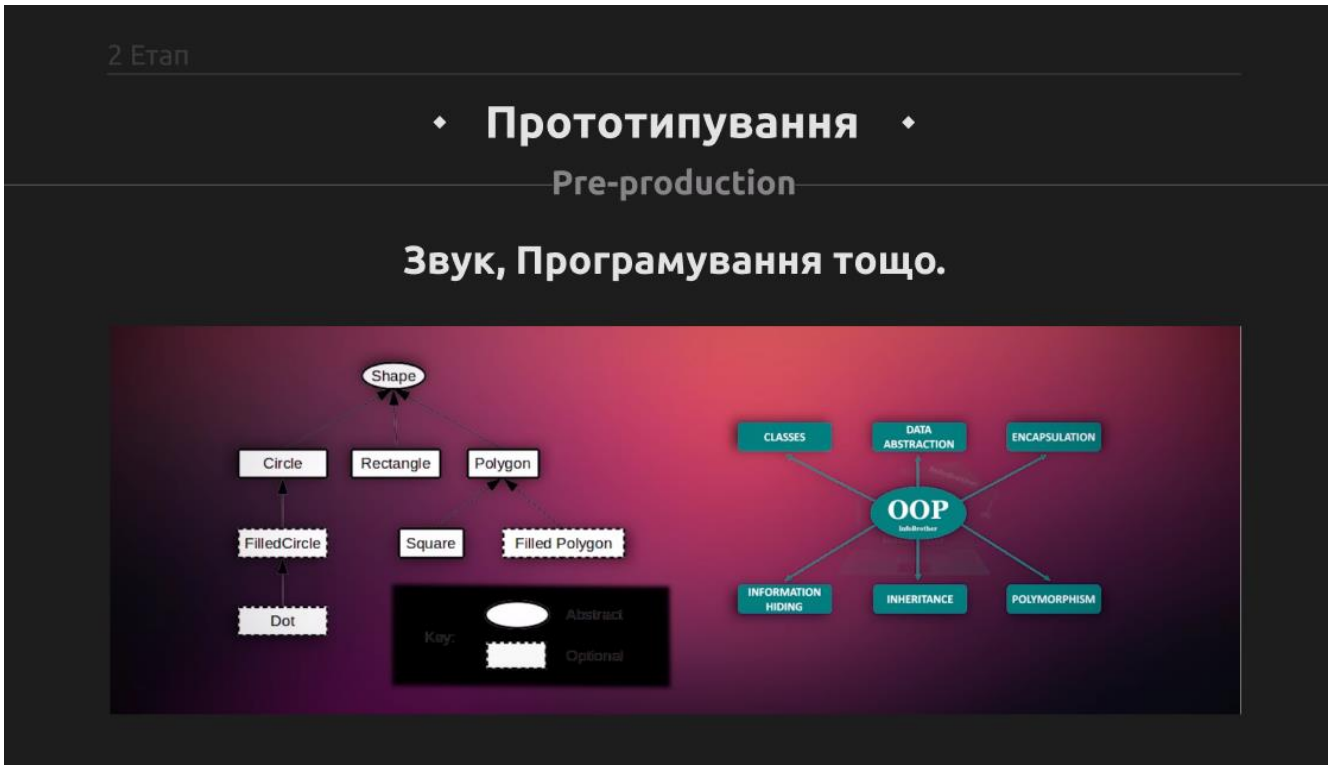


Рисунок 2.77 – Підключення аудіо-програмістом звуку до гри

2.5 Програмування

Коли фінальний контент потрапляє у гру, його необхідно зробити грабельним, тобто запрограмувати. Як зазвичай виглядає робочий процес програміста, як і всі згадане, програмний алгоритм починається з опису і обговорюється, що потрібно зробити.

Потім складається блок-схема, вона описує алгоритми чи процеси як візуальної структури, у якій окремі кроки зображені як блоків різної форми і ці

блоки з'єднані між собою лініями, які вказують напрямок послідовності виконання програми (рис.2.78).

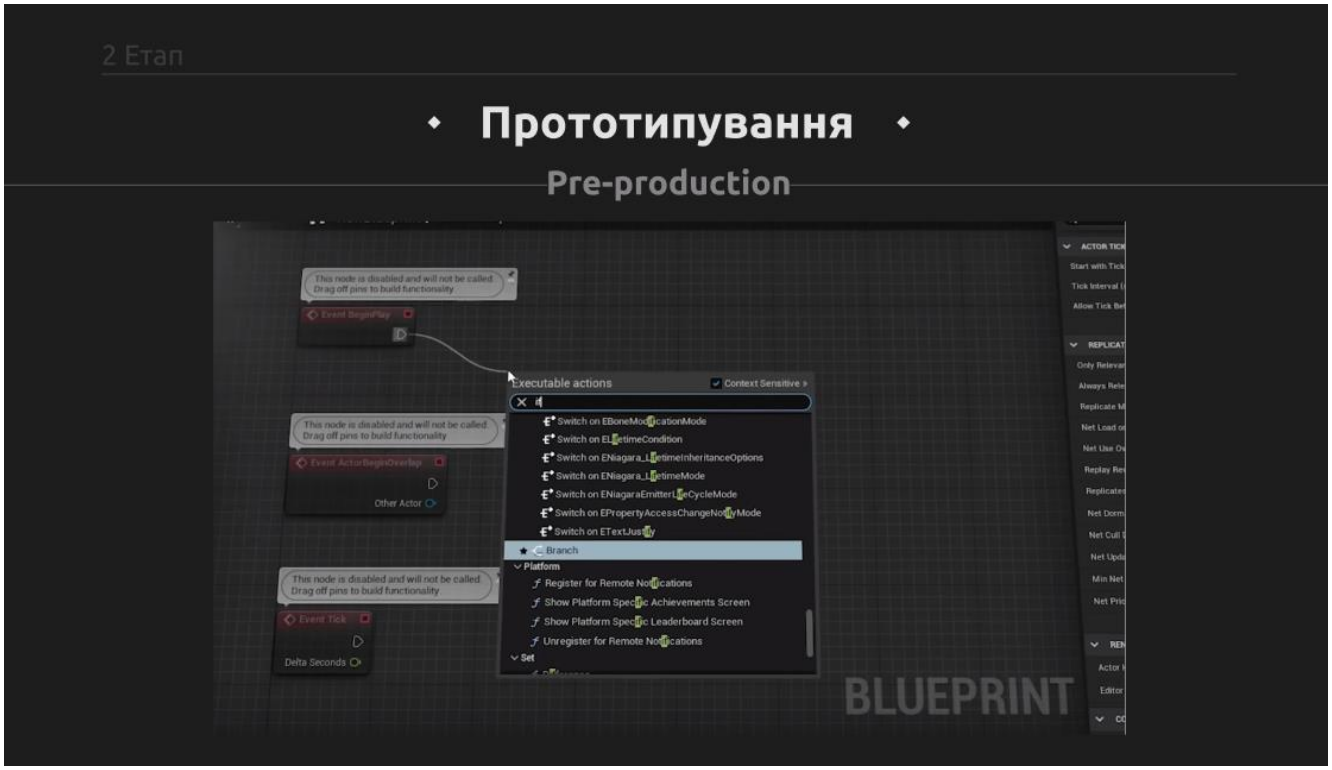


Рисунок 2.78 – Програмування контенту гри

Скомпонувати блок-схему, програмісти приступають до роботи, зазвичай в ігрових двигунах використовують мови програмування C + + або C # разом з надбудовами самого рушія.

Для графіки до основної мови підключаються бібліотеки, на кшталт direcx, opengl або vulkan і шейдерні бібліотеки, для відтворення різних ефектів на кшталт відблисків, затінків.

Зважаючи на те, що в іграх потрібно дуже багато програмувати, процедурний методом, де послідовно виконуються оператори та підпрограми, для великих масштабів цей спосіб незручний і тому розробники ігор використовують методи об'єктно-орієнтованого програмування.

У двох словах, як це працює, допустимо вам потрібно створити зброю, ви створюєте базовий клас - Зброю і привласнюєте їй базові властивості зброї, наприклад такі як шкода, дальність стрілянини, кількість патронів і потім, якщо вам

потрібно створити велику кількість зброї, ви не робите щоразу заново нову, а просто успадкуєте від базового класу і підставляєте туди потрібні параметри.

Щоб зробити розробку ще простіше, використовується візуальне програмування в рамках самого движка, тут замість коду ви маніпулюєте блок схемами, які називаються вузли – ноди, це полегшує поріг входження, звичайно, це не окрема мова програмування, а просто візуальна надбудова над основною.

Для мобільних ігор можуть використовуватися мовою java або Swift.

Геймплей програміст виконують основну роботу, втілює в життя всі ігрові механіки, це те, що пожвавлює світ і дозволяє гравцеві взаємодіяти з ним.

І програміст створює штучний інтелект, ця модель поведінка ігрових персонажів, ботів тут так само можна використовувати нову систему Behavior Tree - Дерево поведін, тут в залежності від ситуації описується подальша модель поведінки.

Завдяки такій нодовій системі з логікою можуть працювати і дизайнери не торкаючись коду, є ще мережеві програмісти, програміст анімації, інтерфейсів та ще багато інших.

3 ІНШІ ЕТАПИ

3.1 Вертикальний зріз та виробництво гри

Коли основний контент готовий, розробники формують вертикальний зріз гри, це шматочок гри без самої гри, розробники заглядають у майбутнє, плануючи як виглядатиме гра.

Пара рівнів, рядків коду та персонажів стане майбутнім каркасом для всієї гри.

Вертикальний зріз, він же демо версія, може виконуватися на етапі продакшену – виробництва гри, тобто під час основної розробки. Вертикальний зріз демонструють інвесторам.

Це найскладніша і найтриваліша стадія розробки, це основний і найтриваліший процес розробки, який вже триває роками, це наповнення контентом, створюється все більше рівнів, 3d моделлер роблять ще більше моделей, налаштовується фізика, штучний інтелект, якщо це платформер то її може бути дурнішим, а для стратегії в реальному часі ртс потрібен більш розумний і опрацьований, для мультиплеєрів і зовсім не потрібний хіба що для роботів.

Саунд-дизайнер додає звуки.

Наступний крок додавання графічного інтерфейсу він же G.U.I. Від його виду також залежить, чи сподобається гра гравцю чи ні.

Візуальна частина інтерфейсу створюється в будь-якому редакторі 2d і потім компонується і програмується в самому движку.

Більшість етапів розробки може бути паралельно. У результаті кожен із відділів розробників прикручує до гри свою частину.

На завершальному етапі, ближче до релізу робиться робота над помилками – фінальна частина – яка називається – полішинг – тут художники вже не створюють щось нове, а покращують старе, збирається

ударна команда, загальне сили якої кидаються рішення конкретної серйозної завдання чи проблеми.

Поки гра ще не зібрана до кінця, можна випустити альфа-версію, а нумерація версії – це послідовність чисел 1 цифр – це Major версія, вона ж стабільна версія, друге число Minor – молодша
версія, якісь великі зміни, третя кількість дрібних змін - Micro версія.

Може бути більше трьох значень, ще молодша версія може бути позначена літерою a, b, c і так далі.

3.2 Тестування та підготовка до релізу

Коли гра готова, настає стадія тестування і цим займаються QA відділ, головне їхнє завдання не шукати помилки всередині коду, а перевіряти чи можна пройти гру до кінця чи ні і виявляти, що заважає повноцінному проходженню.

Коли гра готова можна провести закритий бета-тест, а потім і відкритий, доки йде доопрацювання гри, видавець теж не сидить на місці і займається піаром гри, випускається ігровий трейлер, заразом перед релізом проект може взяти участь у виставках.

Заставки в іграх можуть робити як у вигляді кат-сцен зібраних і зарендерованих на движку так і у вигляді прорендерів cinematic роликів, використовуючи високіполі моделі і фізично коректне освітлення.

У середньому гру робити чотири роки, в Інді студіях гру можуть розробляти 10-30 чоловік, в AAA студіях 300 і вище, а якщо говорити про загальну кількість співробітників, то їх кількість може перевищувати 1000, бюджет хорошої AAA гри може становити 200 мільйонів, а то й вище, і майже половина цього бюджету може піти на маркетинг..

3.3 Оперування

І фінальний етап - це оперування гри, її просування - це може виконуватися на стороні видавця.

Видавцем може бути стороння чи дочірня компанія. Видати локалізує гру, перекладає різними мовами, рекламує її та займається продажами, забираючи шматочок прибутку, приблизно 30%.

У результаті гра доходить до кінцевого споживача, потім вже розробник створює патчі та dlc.

4 СТВОРЕННЯ ІГРОВОГО КОНТЕНТУ

4.1 Етапи створення моделей

Етапи створення 3D-моделі капібари:

1. Пошук референсів для точного відтворення форми та пропорцій.

Перш ніж почати моделювання, потрібно знайти високоякісні зображення та відео капібар під різними кутами (спереду, збоку, зверху тощо). Потрібно запам’ятати пропорції їх тіла, характерні риси, текстуру хутра, будову скелета. Саме хороші референси забезпечують точність моделі, та допоможуть передати суть капібари.

2. Початок роботи над моделлю з базової форми, задаючи основну структуру. Починаємо моделювання капібари, створивши приблизний контур її тіла. Використовуйте прості геометричні фігури, такі як куби, сфери та циліндри, щоб визначити голову, тулуб, ноги та обличчя. У Blender ми можете використовувати функцію Mirror Modifier, щоб зберегти симетрію, особливо для тіла, голови та ніг. На цьому етапі блокується основна форма, тому поки що не треба турбуватися про дрібні деталі (рис.4.1).

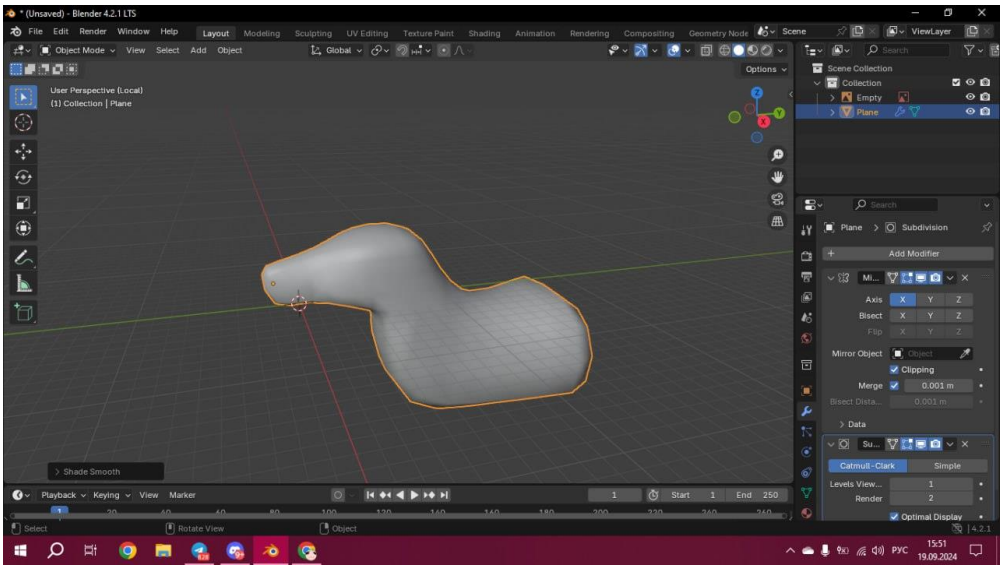


Рисунок 4.1 – Базофа форма і структура моделі

3. Деталізація обличчя для надання характерних рис капібари.

Коли основна форма готова, переходимо до вдосконалення ключових характеристик капібари. Зосередимось на:

- Округла форма голови і великий ніс.
- Розташування і розмір очей і вух.
- Визначення вигинів тіла, тонкий горб на спині та міцні ноги. На цьому етапі ми можете скористатися інструментами скульптингу Blender або змінити сітку, додавши додаткові петлі по краях, щоб отримати бажану форму (рис.4.2).

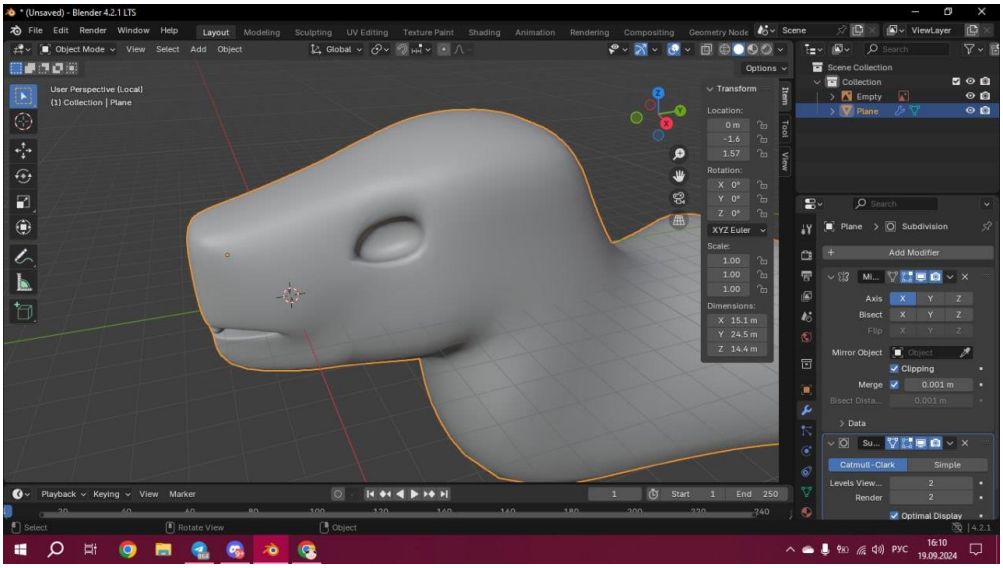


Рисунок 4.2 – Деталізація обличчя моделі

4. Створення та деталізація вух.

Тепер нам слід уточнити нашу сітку, додавши більше геометрії, де це необхідно, щоб отримати більш плавні криві та більше чіткості. Починаємо ліпити або додавати дрібніші деталі, як-от зморшки на шкірі, форми стоп і загальну форму тіла. Такі інструменти, як Subdivision Surface, можуть допомогти збільшити гладкість сітки, не додаючи зайвої геометрії одночасно.

5. Моделювання та доопрацювання ніг та інших частин тіла.

Тепер працюємо над деталізацією окремих частин тіла:

- Ноги та ступні: сформуємо кігті та пальці.
- Обличчя: додаємо дрібніші риси, як-от ніздрі, повіки та тонкі деталі шкіри.

- Вуха та хвіст: капібари мають маленькі округлі вуха та маленький хвостик. Також перевіряємо, що вони пропорційні та чітко визначені (рис.4.3).

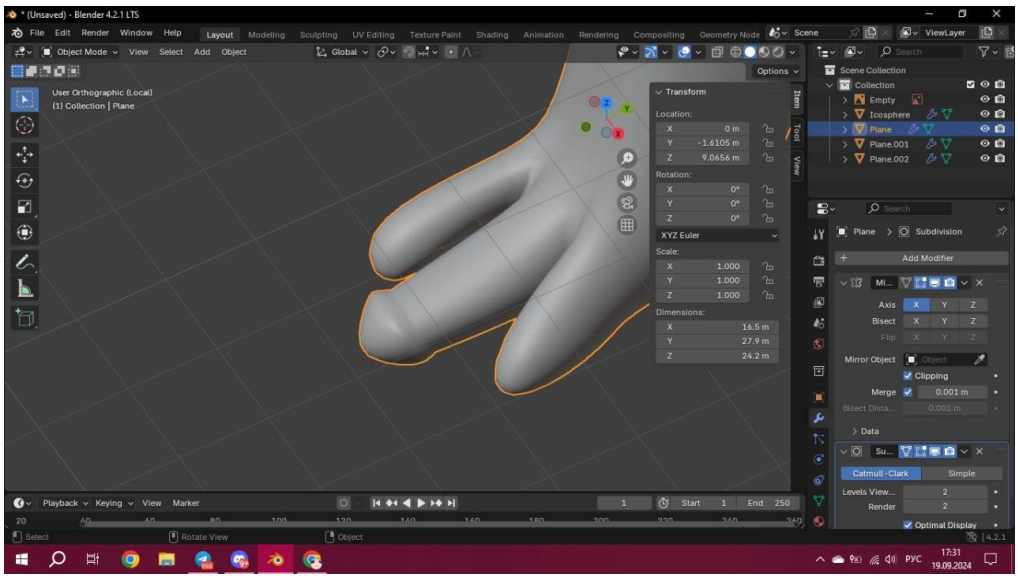


Рисунок 4.3 – Формування ноги моделі

6. Розгортка моделі для подальшої роботи з текстурами.

Після завершення скульптингу з високою деталізацією, робимо ретопологію. Цей процес передбачає створення низькополігональної версії моделі шляхом спрощення геометрії сітки, зберігаючи загальну форму та деталі. Ретопологія забезпечує кращу продуктивність під час анімації та текстурування. Інструменти Blender Shrinkwrap Modifier і Quad Draw можуть допомогти в цьому процесі (рис.4.4).

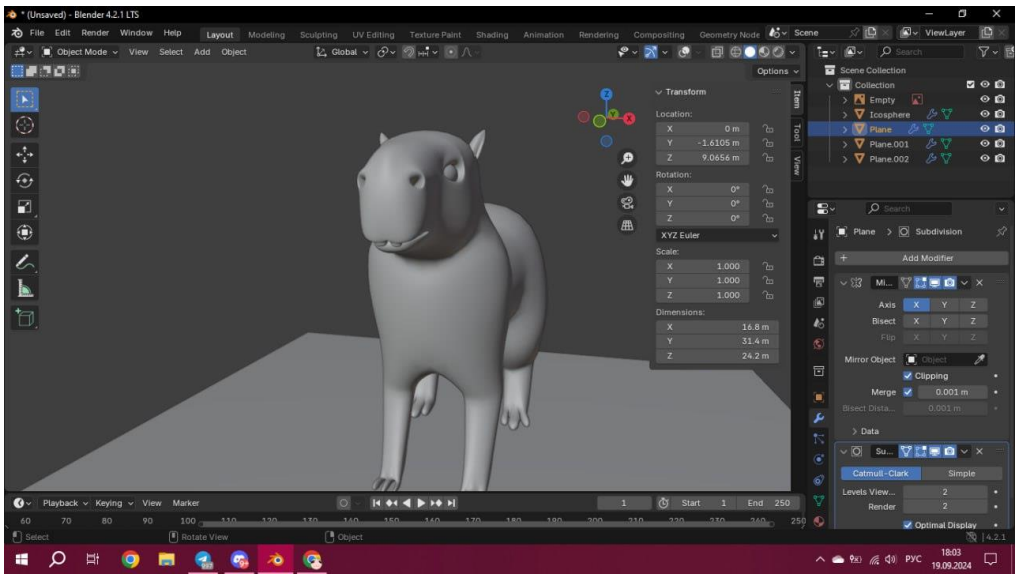


Рисунок 4.4 – Завершення деталізації моделі і підготовка до ретопології

7. Створення текстур надання моделі реалістичного виду.

Після завершення розробки моделі переходимо до UV розгортки. Нам потрібен чистий UV макет і оптимізований для фарбування текстури. Ми можете скористатися такими інструментами, як Smart UV Project у Blender, або розгорнути вручну.

Потім створюємо текстури (рис.4.5):

Рисунок 4.5 – Текстура моделі

- Текстура основного кольору: використовуємо референси, щоб намалювати реалістичний колір хутра та відтінки шкіри. Капібари зазвичай мають суміш коричневого та сірого хутра.
- Карта нормалей: додає дрібні деталі поверхні, як-от напрям хутра чи шорсткість шкіри, без додавання геометрії.
- Карта шорсткості: визначає, які області матові, а які більш блискучі, наприклад, злегка вологий ніс або світлові очі.

8. Створення скелета моделі (ригінг) для підготовки до анімації.

Створємо скелет (оснастку), щоб підготувати капібару до анімації. Почнемо з додавання арматури та поміщяємо кістки всередину сітки. Основні групи кісток будуть включати:

- Хребет для тіла.
- Ноги і стопи для ходьби.

- Голова і шия для рухів головою.
- Вуха та обличчя, якщо ми плануємо анімувати міміку.

Використовуємо функцію Automatic Weights у Blender, щоб призначити вершини сітки кісткам, а потім вручну налаштуйте Automatic Weights для кращого контролю.

9. Фінальна перевірка сумісності моделі з Unreal Engine (рис.4.6).

Після ригінгу (Створення скелета моделі для анімації) експортуємо модель (у такі формати, як FBX або OBJ) та імпортуйте її в Unreal Engine, щоб протестувати її в ігровому середовищі. Перевіряємо:

- Геометрія моделі на будь-які похибки або деформації.
- Анімації, якщо вони плавні та поводяться належним чином.
- Текстури та матеріали для коректного вигляду при різному освітленні.
- Цей останній етап гарантує, що модель готова до використання.

Рисунок 4.6 – Готова модель

4.2 Етапи створення ігрового інтерфейсу

Етапи:

1. Референси жанру та механік
2. Вайрфрейми
3. Пошук заглушкових зображень (арта)
4. Візуальні референси UI
5. Скетчі UI / пошук стилю
6. Полішинг

Це всі кроки, які входитимуть до процесу створення навчального проекту. Навчальний проект – це не справжній комерційний проект, а той, який ви придумали, щоб продемонструвати свої навички.

Наша робота буде розбита на 6 етапів. Ось короткий огляд кожного з них:

Референси жанру та механік. Спершу ми обираємо жанр нашого проекту, наприклад, RPG, шутер, гонки, мобільна гра, "match 3", батлер тощо. Потім шукаємо подібні проекти і збираємо референси, щоб створити вайрфрейми та мати перед очима типові для цього жанру елементи. Це важливий етап, адже якість нашої роботи залежатиме від того, наскільки вона виглядає як справжня гра, з продуманим інтерфейсом і логікою.

Вайрфрейми. Після збору референсів створюємо вайрфрейми і продумуємо, що саме буде демонструвати наш проект. У навчальних проектах ми самі визначаємо всі механіки.

Пошук зображень-заглушок і арту. Інтерфейс потребує візуальних елементів, тому важливо підібрати відповідний арт. Це може бути викликом, особливо для навчальних проектів, де слід знайти якісні зображення-заглушки.

Якщо ви обрали казуальний стиль, то можна зробити попапи з розмитим фоном та іконками. Для складніших ігор потрібні візуально схожі зображення, щоб проект виглядав максимально реалістично.

Візуальні референси для UI. У навчальному проекті ми використовуємо два набори референсів: один для жанру та механік, на основі яких створюємо UX, а другий – для стилістики інтерфейсів, на яких базуємо візуальну частину.

Скетчі UI та пошук стилю. Це етап визначення напряму дизайну, або Art Direction, де ми шукаємо найкращий стиль для нашого інтерфейсу, вносимо зміни і вдосконалюємо його.

Полішинг. На завершальному етапі ми підганяємо всі елементи під розміри, робимо їх піксельно точними, вирівнюємо по сітці та перевіряємо відповідність піксельній сітці.

Референси жанру та механік:

Цей етап простий – ми відвідуємо різні сайти з базами ігрових інтерфейсів, обираємо жанр гри та збираємо референси з потрібними механіками, елементами та контентом у схожих проектах (рис.4.7).

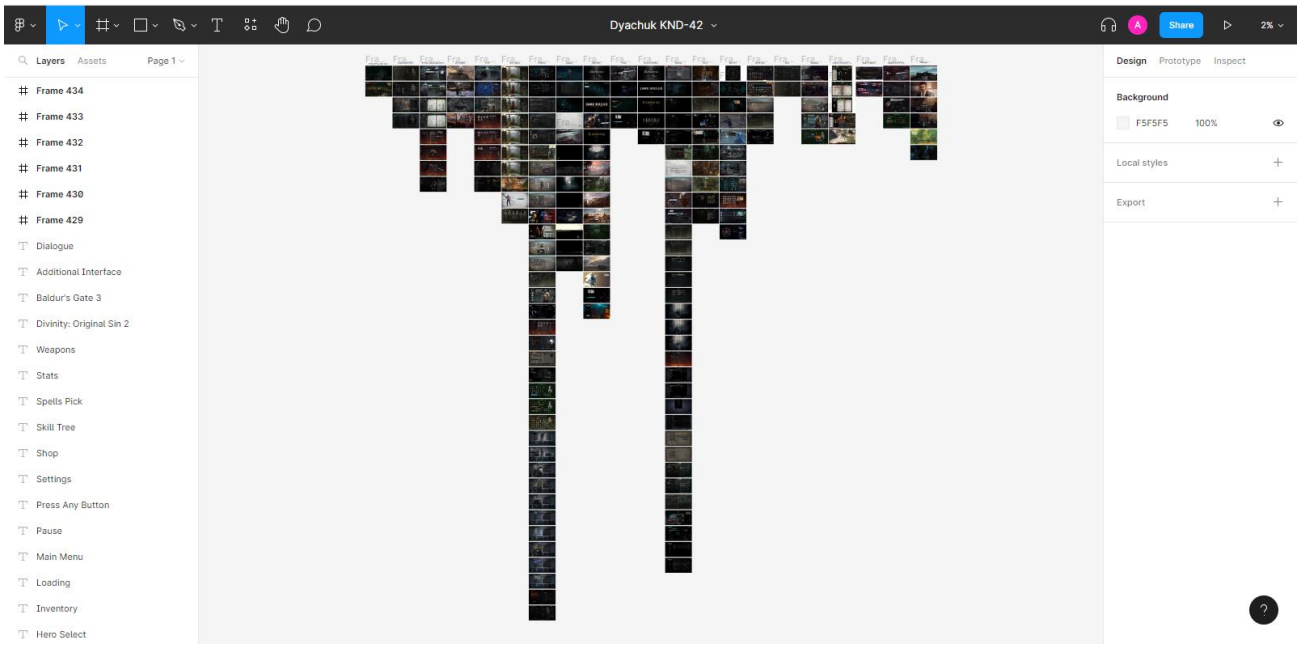


Рисунок 4.7 - Референси жанру та механік у Figma

Вайрфрейми:

На цьому етапі ми ще не займаємось візуальним стилем гри. Спершу зосереджуємось на функціоналі, створюємо основу проєкту, його «кістяк». Лише

після того, як усе продумано, на наступних етапах додаємо візуальну «оболонку» (рис.4.8, рис.4.9).

Тобто зараз я створюю вайрфрейми, не прив'язуючись до конкретного стилю – фентезі, сай-фай, кіберпанк, реалізм тощо. Це не має значення, адже на підготовлений кістяк можна накласти будь-яку стилістичну оболонку.

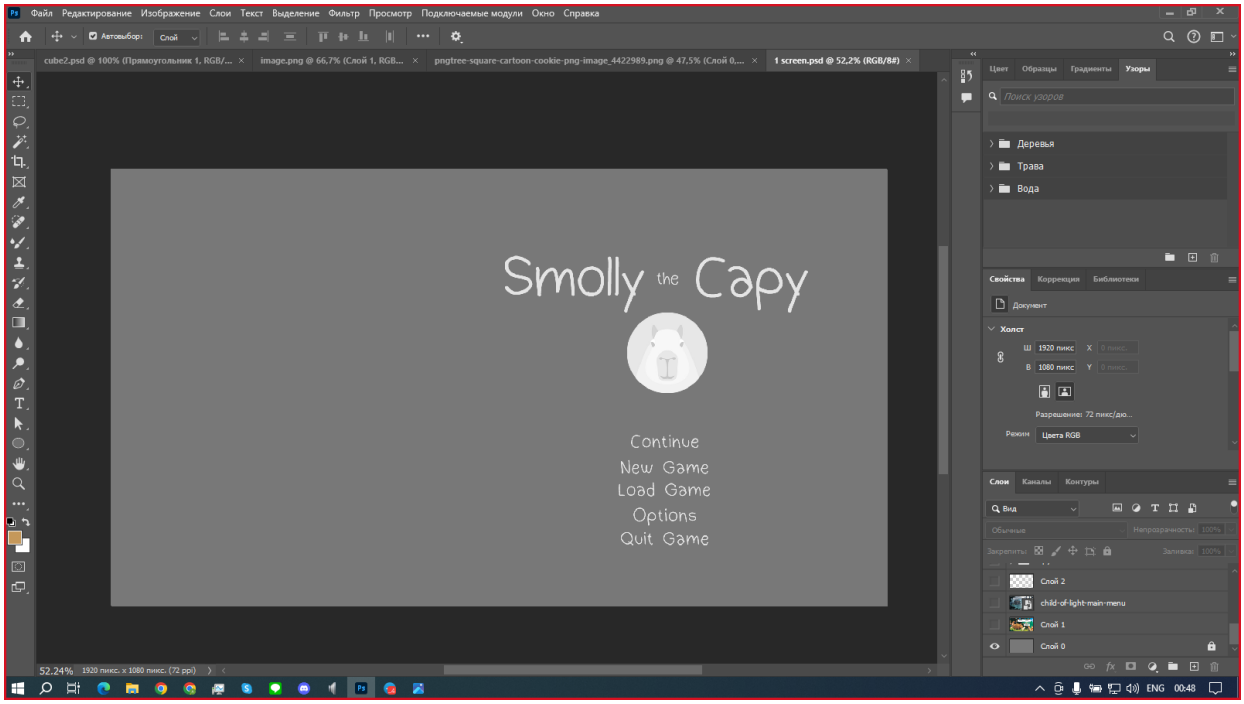


Рисунок 4.8 - Вайрфрейми головного меню

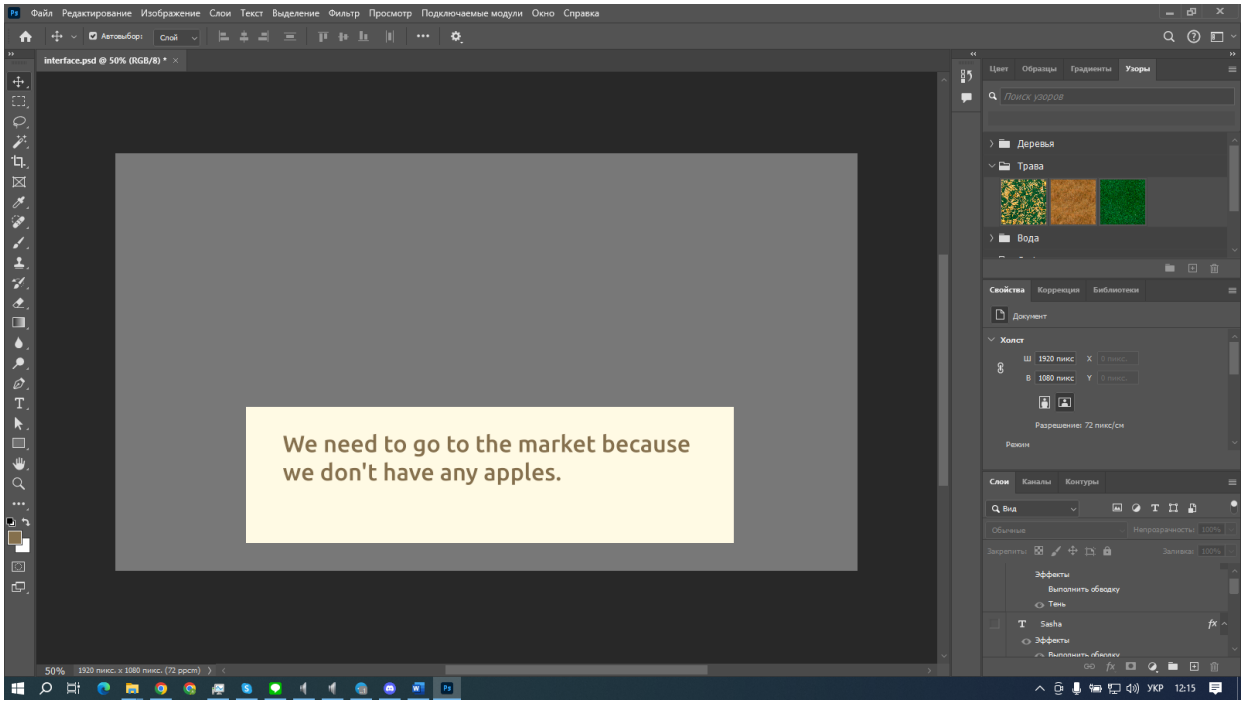


Рисунок 4.9 - Вайрфрейми меню діалогу

Візуальні референси UI (рис.4.10):

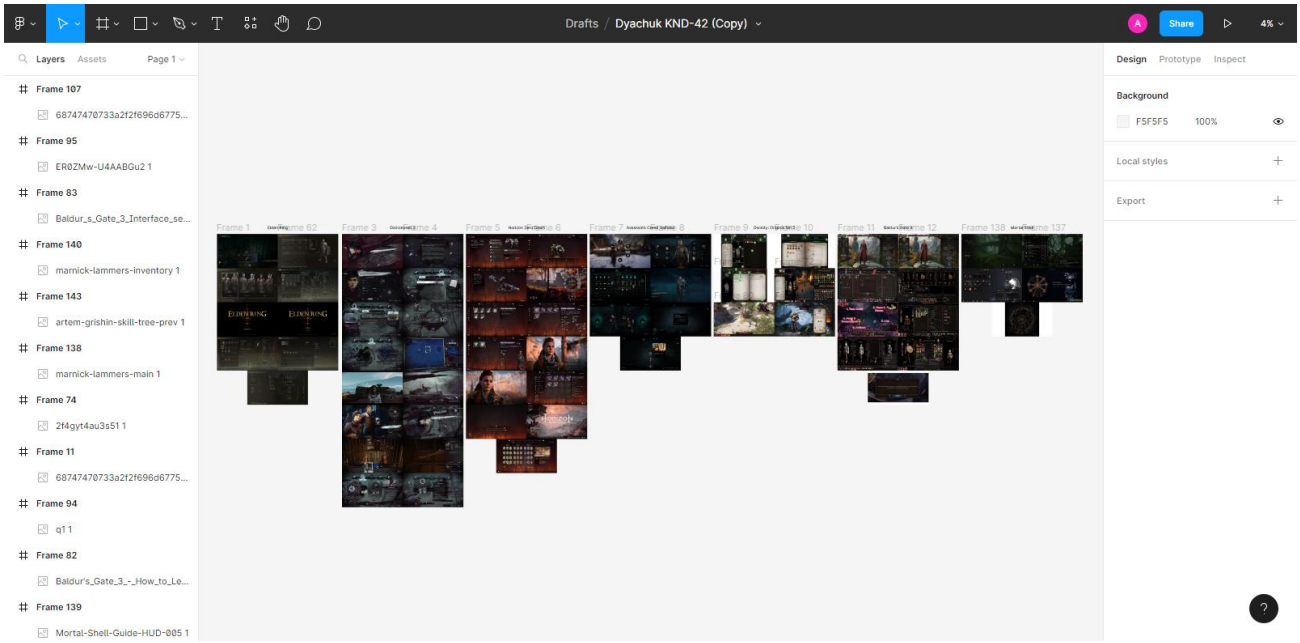


Рисунок 4.10 - Візуальні референси UI у Figma

Скетчі UI / пошук стилю та Полішинг (рис.4.11 - рис.4.12):

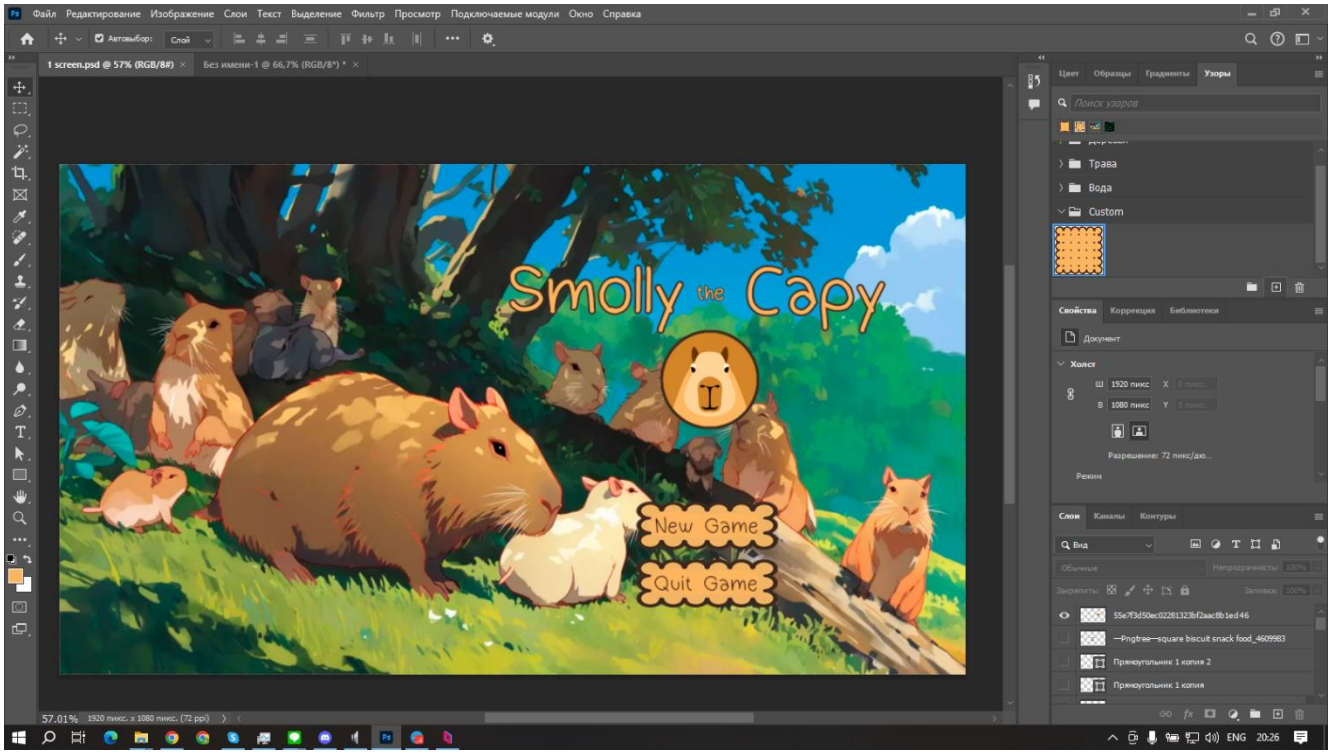


Рисунок 4.11 - Скетч + полішинг головного меню

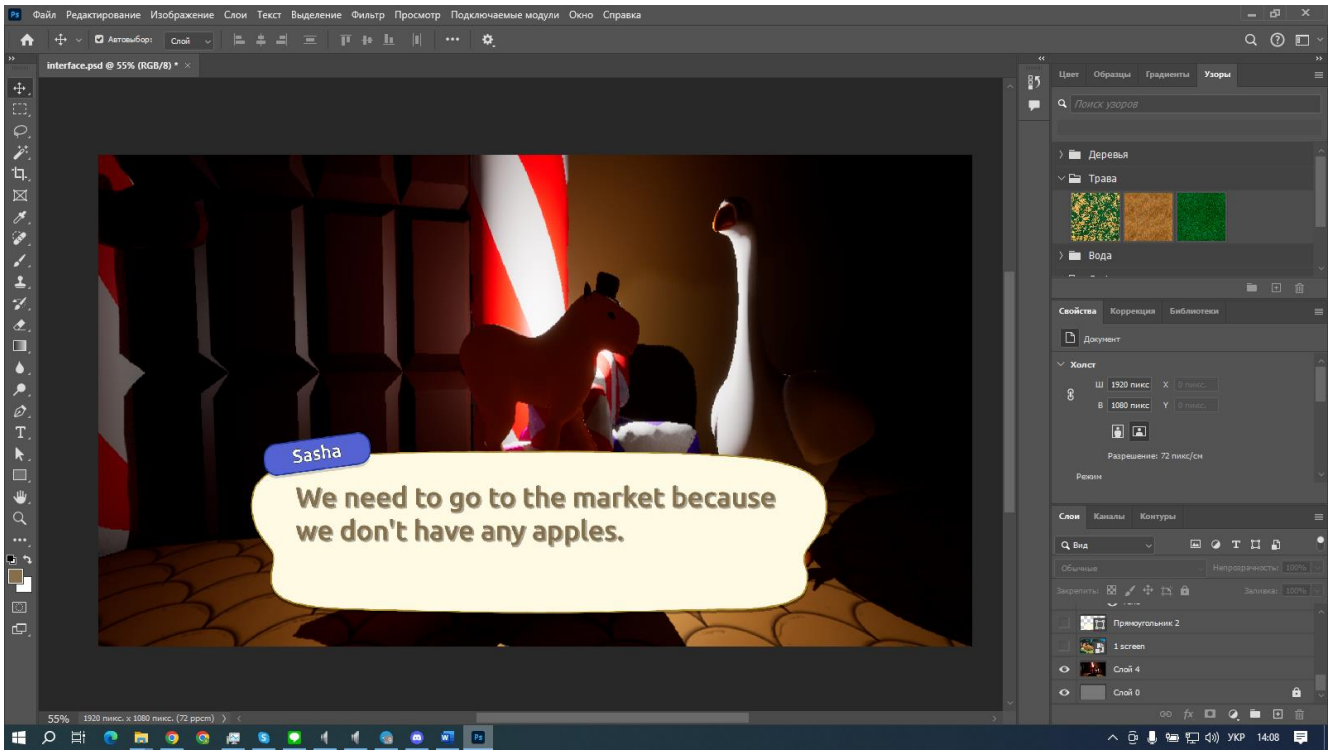


Рисунок 4.12 - Скетч + полішинг меню діалогу

4.3 Етапи роботи з проектому Unreal Engine 5

1. Розробка концепції та планування

Процес почався з генерації ідеї для гри: створення короткої гри про капібару з головоломками, спрямованими на розвиток дітей:

- Розробка концепції гри: визначення жанру, цільової аудиторії (діти), основних механік, головоломок і візуального стилю.
- Створення технічного завдання: визначення необхідних інструментів (Unreal Engine 5, Blender, Adobe Photoshop) та ресурсів (3D-моделі, текстури, звуки).

2. Створення нового проекту в Unreal Engine 5

- У Unreal Engine 5 був створений новий проект, обрано шаблон "Blank Project" для максимальної гнучкості.
- Налаштовано базові параметри сцени: роздільна здатність екрана, частота кадрів, масштаб одиниць.

3. Створення ландшафту (рис.4.13 – рис.4.16)

- Інструмент "Landscape": використаний для створення базової карти з пагорбами, долинами та водоймами.
- Текстурування: через систему матеріалів Unreal Engine додані текстури трави, землі та каменю. Для реалістичності використано технологію Layered Materials.
- Деталі ландшафту: додано дерева, кущі та інші об'єкти, створені в Blender, з акцентом на казковий стиль.

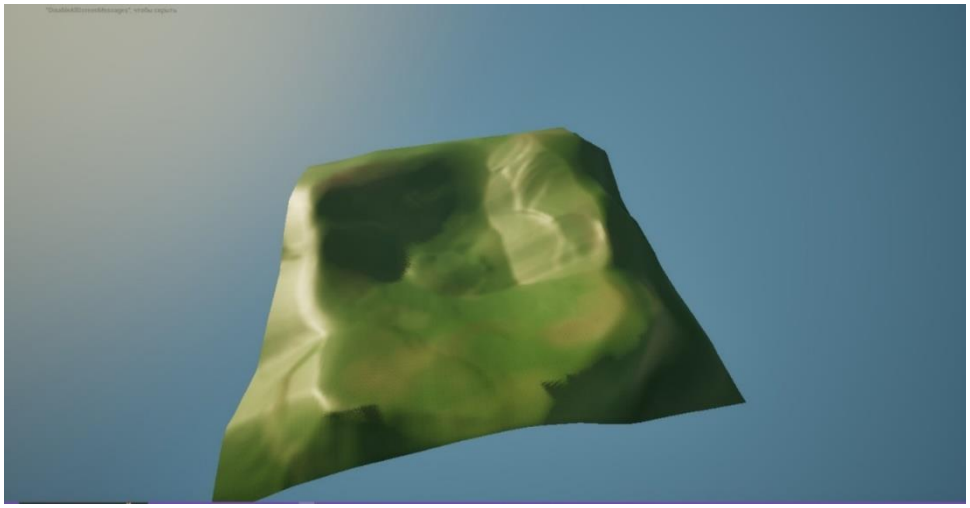


Рисунок 4.13 – Створення ландшафту №1

Рисунок 4.14 – Створення ландшафту №2

Рисунок 4.15 – Створення ландшафту №3

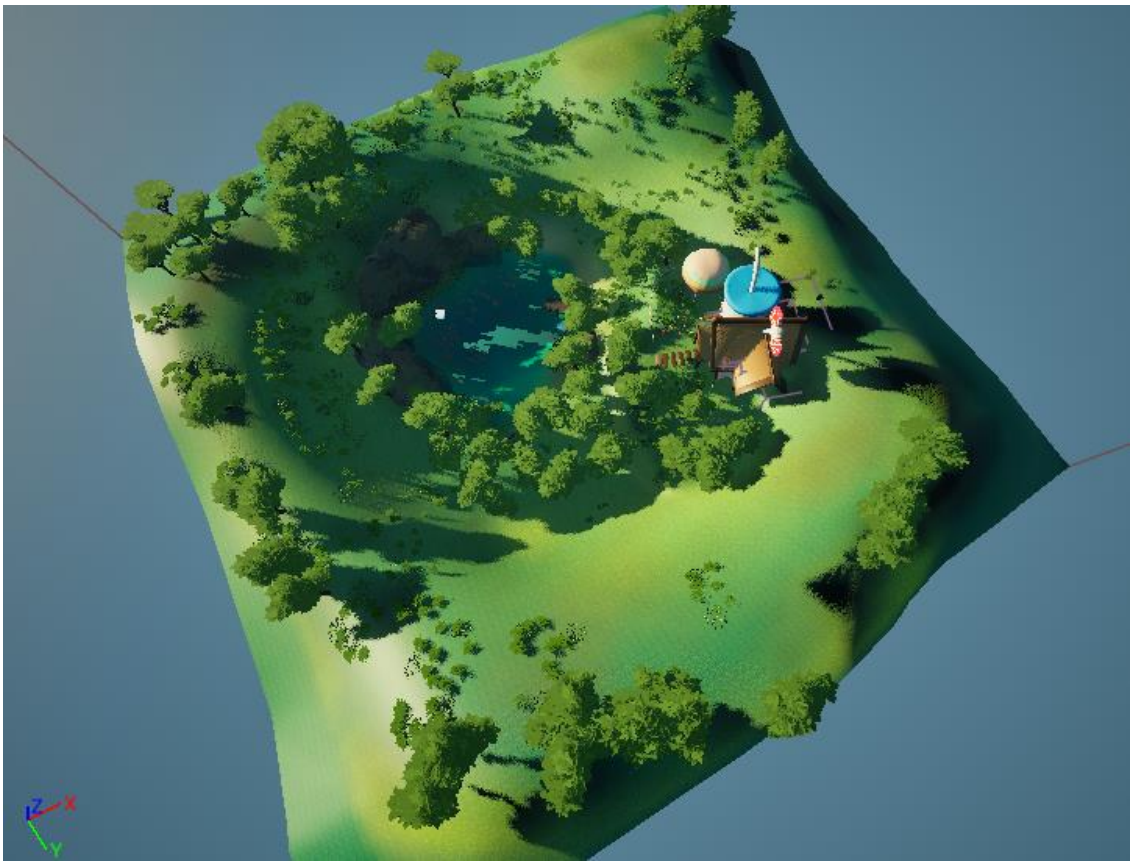


Рисунок 4.16 – Створення ландшафту №4

4. Додавання моделей і розміщення об'єктів (рис.4.17)

- Імпорт моделей: 3D-моделі капібари, об'єктів для головоломок та декорацій створювалися в Blender та імпортувалися у форматі FBX.
- Розміщення об'єктів: за допомогою режиму "Placement Mode" об'єкти розставлені по карті з урахуванням ігрового процесу та зручності гравця.
- Текстури: текстурування виконано в Adobe Photoshop, після чого створені матеріали завантажені в Unreal Engine.

Рисунок 4.17 – Розміщення моделей на карт

5. Налаштування освітлення (рис.4.18)

- Джерела світла: встановлено Directional Light (сонячне світло) та Skylight для рівномірного освітлення.
- Ефекти: використано Volumetric Fog для створення атмосфери, а також додано точкові джерела світла в ключових точках головоломок.
- Налаштування Lumen: забезпечено реалістичне глобальне освітлення.

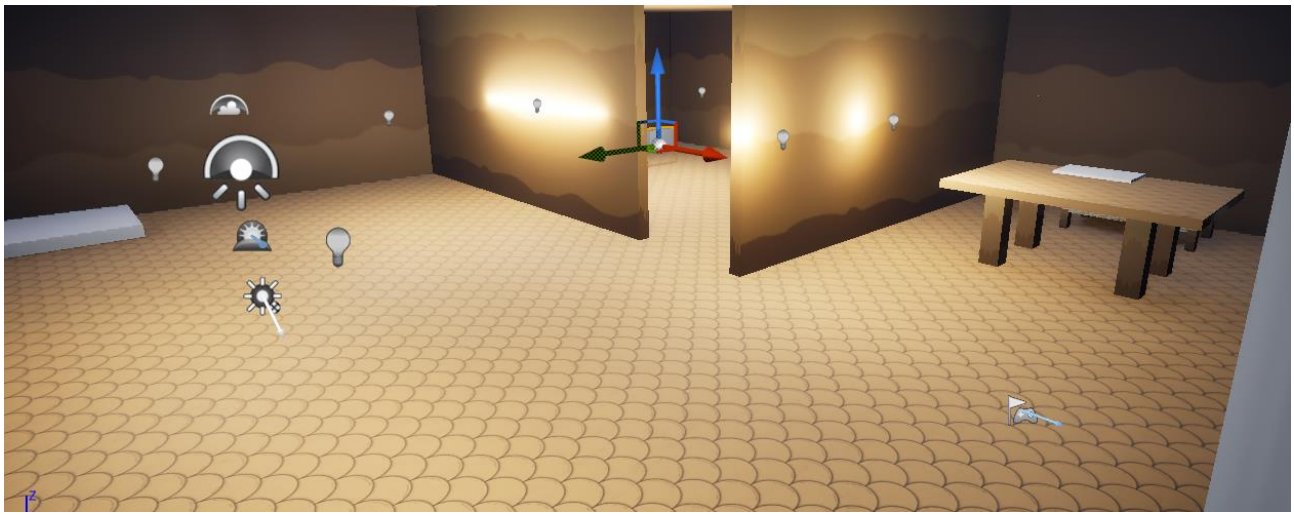


Рисунок 4.18 – Налаштування освітлення

6. Створення головоломок

- Дизайн головоломок: вигадані прості, але цікаві завдання, що сприяють розвитку логіки та пам’яті у дітей.
- Розміщення елементів: головоломки включають інтерактивні об'єкти, такі як двері, важелі, кнопки та кодові замки.

Суть першої загадки у том, що дитина повинна маючи декілька пазлів зображення зібрати повноцінне зображення скриньки, після чого загадка буде виконана і дитина зможе завершити даний рівень (рис.4.19).

Рисунок 4.19 – Головоломка зі збиранням скриньки

Щоб виконати другу головоломку і пройти далі, дитині необхідно правильно розставити зображення на стінах, використовуючи підказки на дверях, тоді двері відчиняться (рис.4.20).

Рисунок 4.20 – Головоломка з підбором картинок для відкривання дверей

Виконуючи третю загадку ми бачимо на стінах наскальні малюнки у вигляді букв ы нам необхідно зібрати з літер слово “Open”, тобто “Відчинити”. ПО таймеру літери змінюються і треба за допомогою кнопки зупинитись на правильній літері на кожній стіні, тоді рівень буде пройдено (рис.4.21).

Рисунок 4.21 – Головоломка з підбором букв для написання слова

У четвертій головоломці у нас є кубики зцифрами і суть у тому що потрібно встановити їх так, щоб сума чисел у кожному з трьох рядів дорівнювала 15, тоді рівень буде пройдено (рис.4.22).

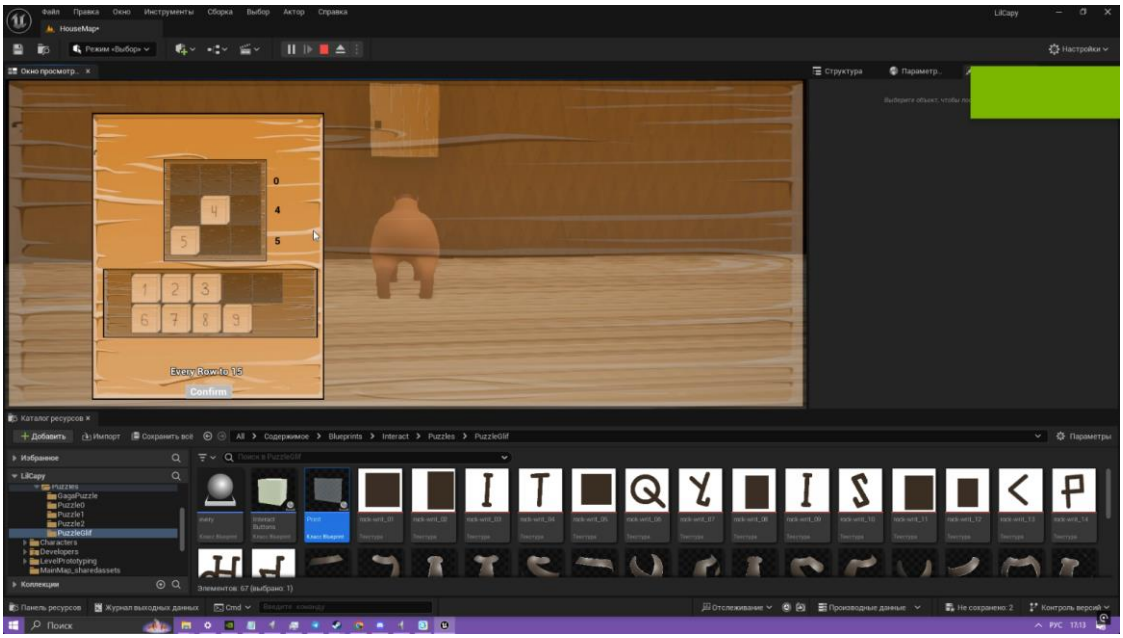


Рисунок 4.22 – Головоломка із кубиків з сумою 15 у ряду

Під час проходження даного рівня, нам необхідно зрозуміти як перейти на інший бік. Для цього необхідно знайти спосіб підняти міст, а саме натиснути кнопку і тримати її, щоб білка змогла перейти на іншу сторону і там вона зможе натиснути на іншу кнопку і знову підняти міст для вас (рис.4.23).

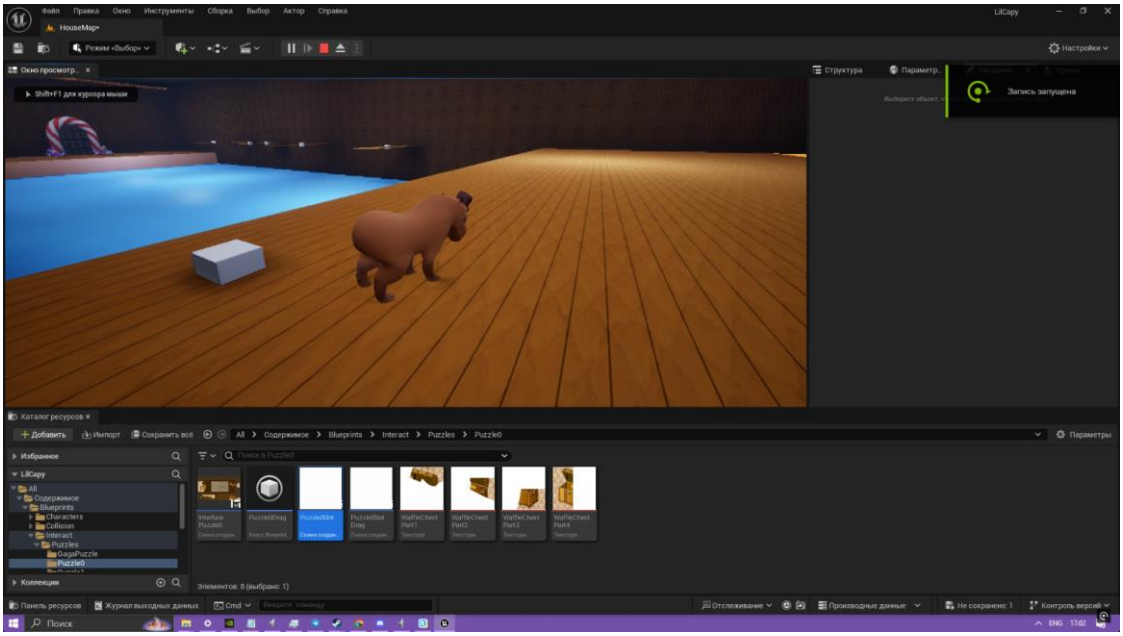


Рисунок 4.23 – Головоломка з допомогою білці пройти на інший берег

7. Блюпринти та ноди

Blueprint Visual Scripting: використано для створення базової логіки гри:

- Відкривання дверей після виконання задачі.
- Анімацію руху персонажа.
- Інтерактивність об'єктів (взаємодія з кнопками та важелями).

Приклади нодів:

- "Event BeginPlay" для ініціалізації ігрових об'єктів.
- "Branch" для реалізації умовних операторів.
- "Timeline" для анімації руху об'єктів.

Головоломка зі збиранням скриньки:

Ця частини коду віповідають за відображення частин скриньки наєкрані (рис.4.24, рис 4.25).

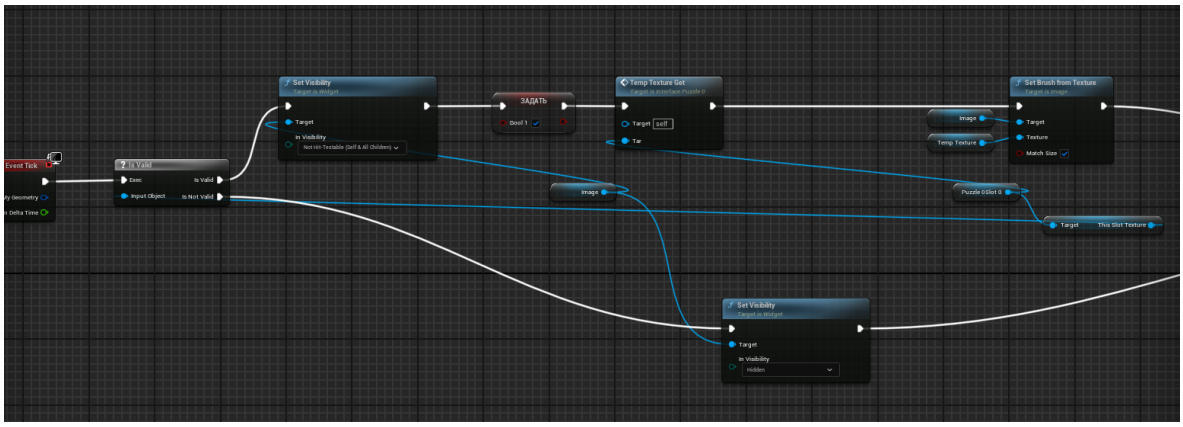


Рисунок 4.24 – Блюпринти головоломки зі збиранням скриньки №1

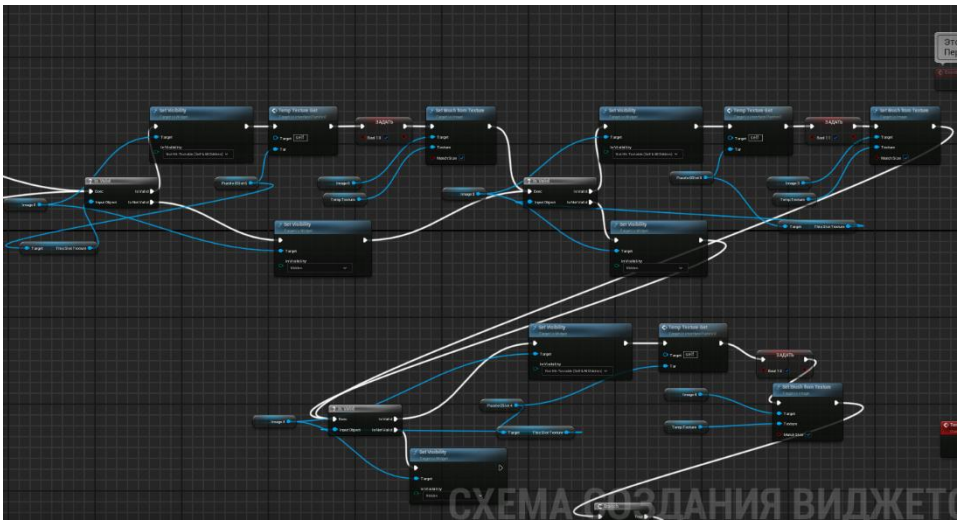


Рисунок 4.25 – Блюпринти головоломки зі збиранням скриньки №2

Цей уривок коду відповідає за отримання текстур за певних умов (рис.4.26).

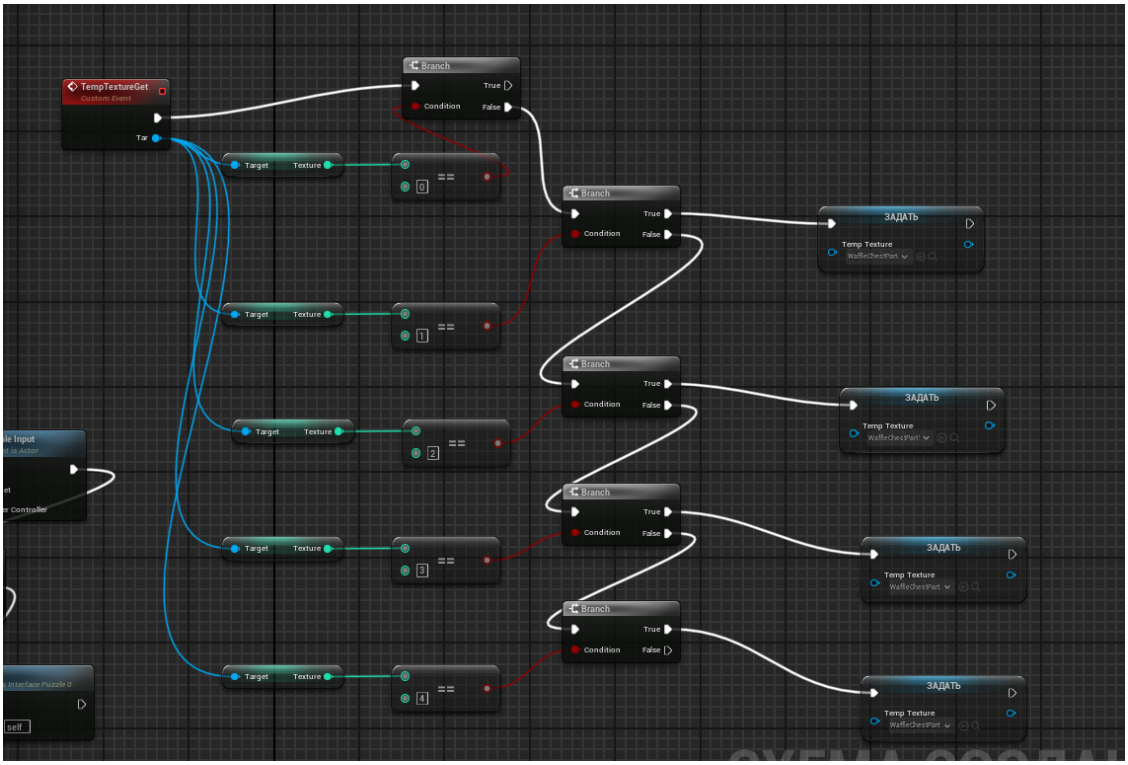


Рисунок 4.26 – Блюпринти головоломки зі збиранням скриньки №3

Нода Branch перевіряє логічну змінну типу Boolean і розгалужує виконання на два шляхи - True (якщо умова виконується) або False (якщо умова не виконується) (рис.4.27).

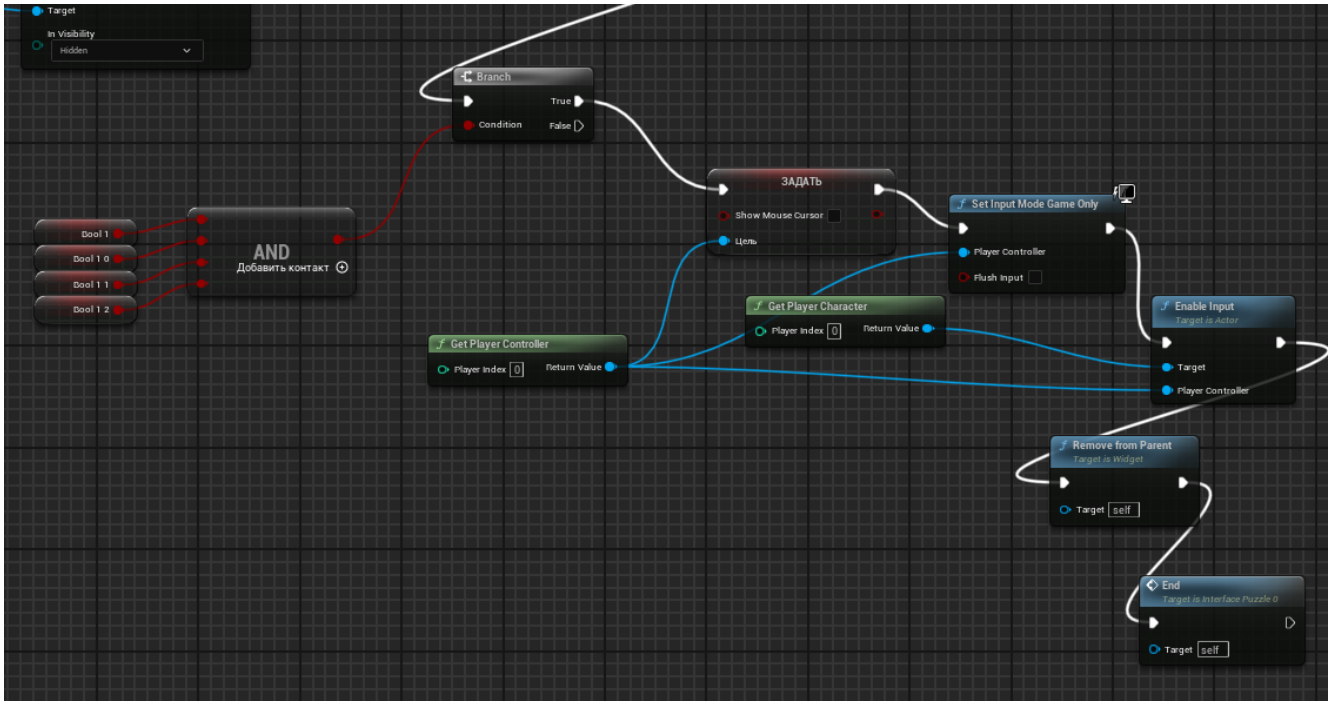


Рисунок 4.27 – Блюпринти головоломки зі збиранням скриньки №4

Завершення коду, створення скринкьки, колізії моделі, та програвання її анімації (рис.4.28):

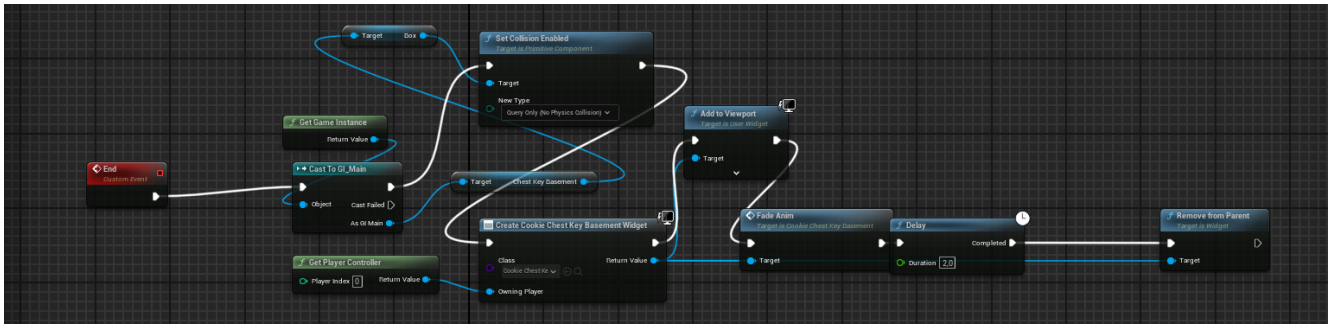


Рисунок 4.28 – Блюпринти головоломки зі збиранням скриньки №5

Event Tick - щоразу в тік перевіряє стан панелей усередині віджету щоб показувати скриню на полі.

Set Visibility встановлює видимість об’єктів (рис.4.29).

Рисунок 4.29 – Set Visibility

Temp Texture Get необхідний щоб можна було міняти об’єкти місцями (рис.4.30).

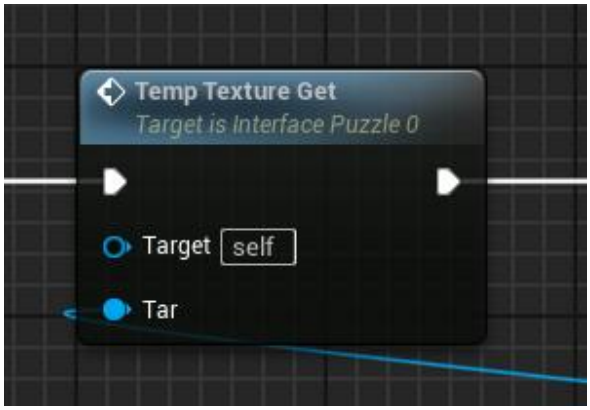


Рисунок 4.30 – Temp Texture Get

Цей івент робить функцію для створення віджету та змінює видимість об'єктів (рис.4.31).

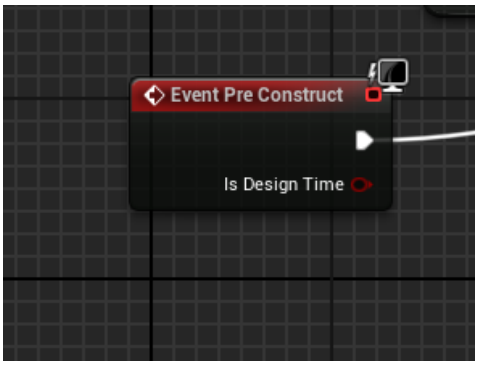


Рисунок 4.31 – Event Pre Construct

Встановлюється Set Visibility у даних Блюпринтах (рис.4.32).

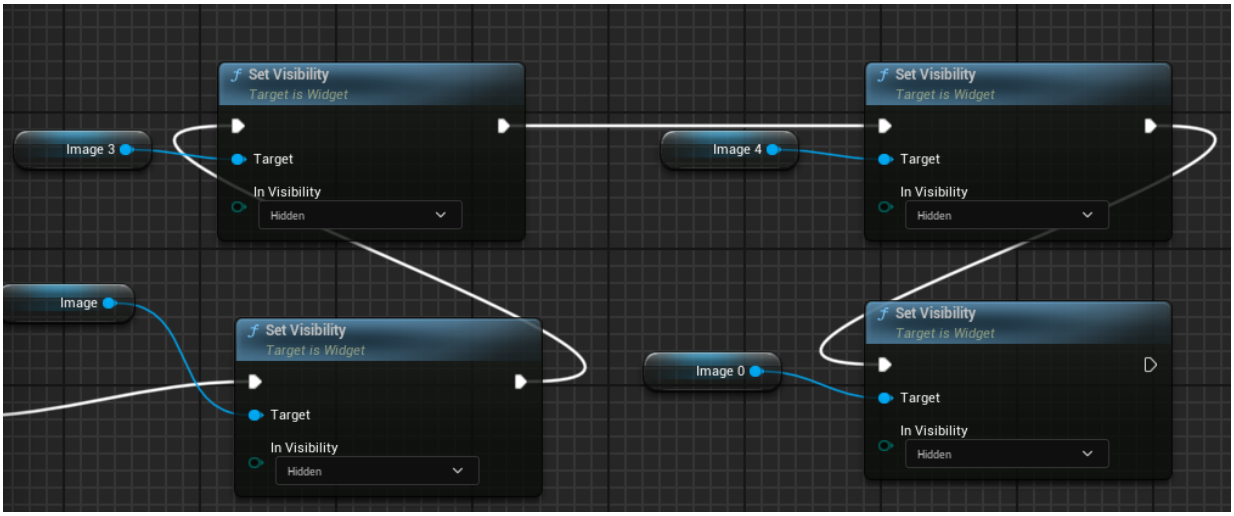


Рисунок 4.32 – Set Visibility

Ця функція отримує тексутру з іншого віджету який є слотом для зображення (рис.4.33).

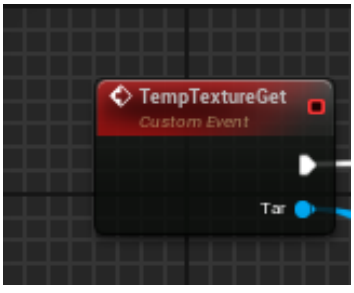


Рисунок 4.33 – Temp Texture Get

Ця функція рахується коли всі картинки розташовані правильно і закінчує функцію виводячи з інетрфейсу і встановлюючи змінну у глобальному блюпринті

Це слот зображення, який можна перетягувати. Цей віджет використовується для відображення графіки в UI (рис.4.34).

Рисунок 4.34 – Віджет зображення

Головоломка з допомогою білці пройти на інший берег (рис.4.35):

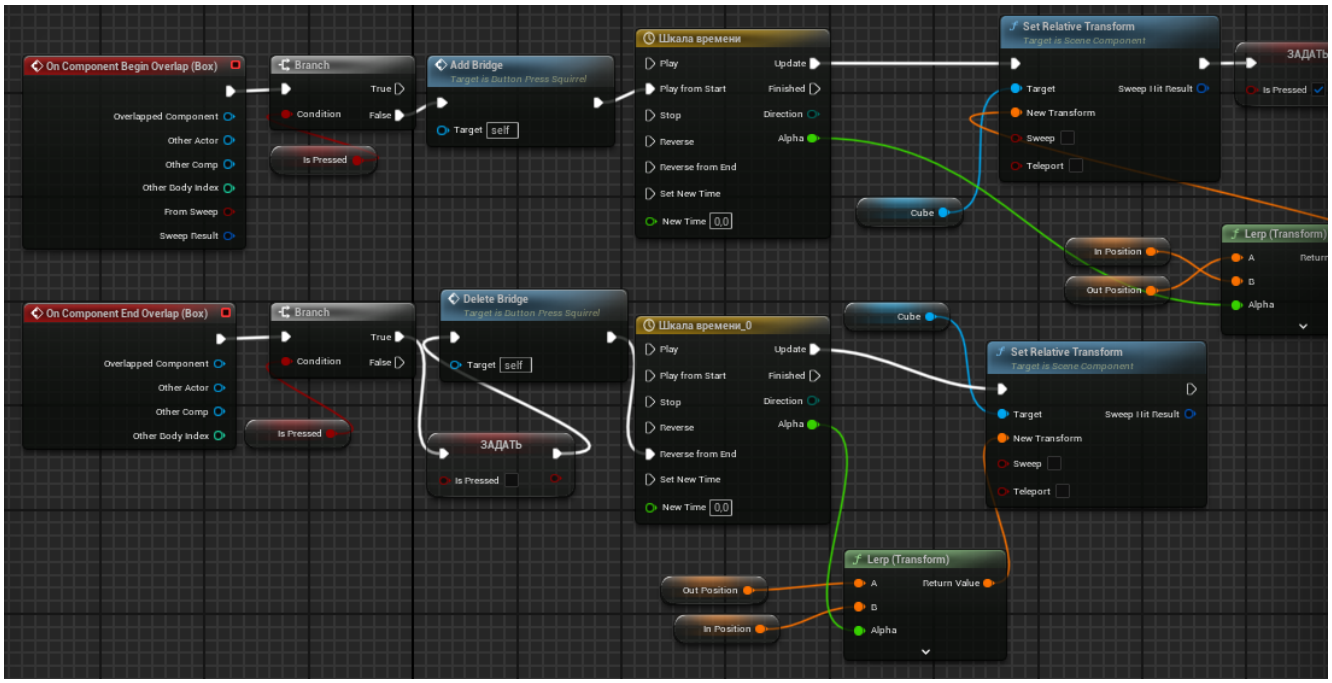


Рисунок 4.35 – Блюпринти даної головоломки

Ці функції реалізують Старт та кінець колізії об'єкта (рис.4.36).

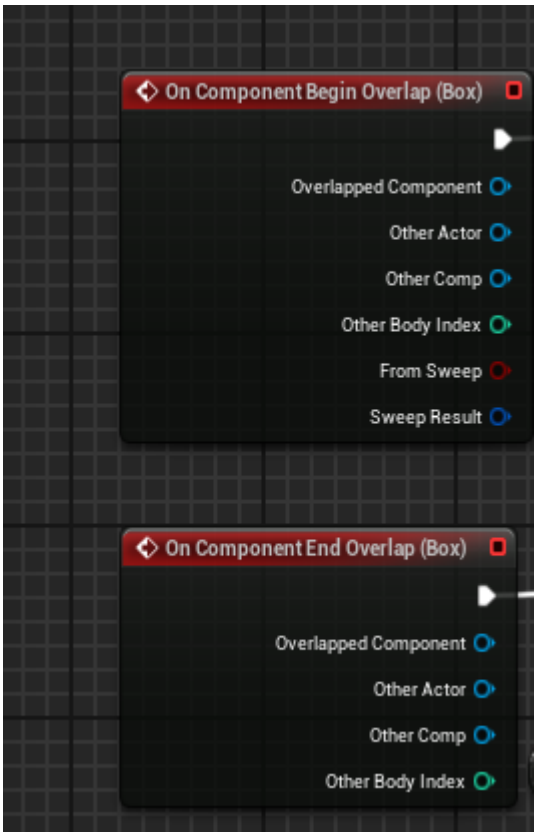


Рисунок 4.36 – Begin and End Overlap

Суть цих блюпринтів у тому щоб виконати такий алгоритм:

Зробити міст - почекати - почекати - поміняти положення кожен тік очікування.

На данних Блипринтах реалізовано віджимання кнопки щою міст зник (рис.4.37).

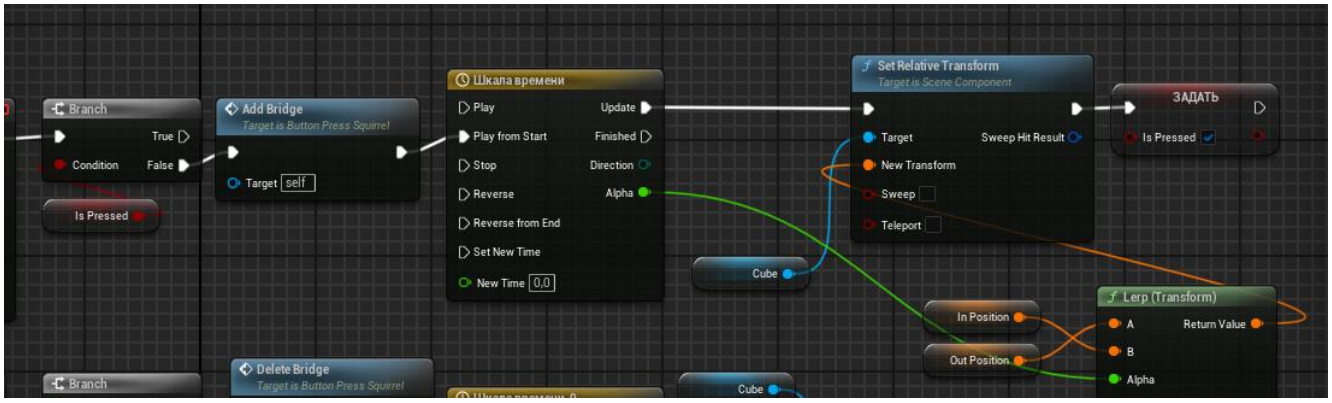


Рисунок 4.37 – Алгоритм віджимання кнопки

Данний алгоритм працює так само, але навпаки, відбувається трансформація, кнопка натискається і з’являється об’єкт міст (рис.4.38).

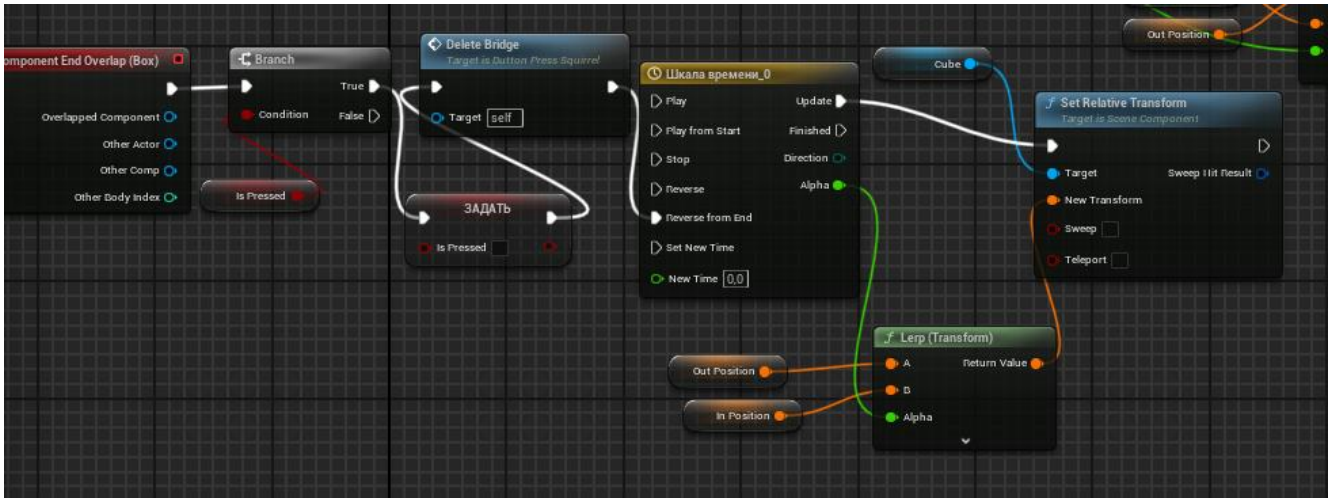


Рисунок 4.38 – Алгоритм натискання кнопки і поява мосту

Блюпринти додавання та видалення мосту (виводить із глобальних змінних інформацію і видаляє звідти міст):

За допомогою цього алгоритму реалізована поява мосту (рис.4.39).

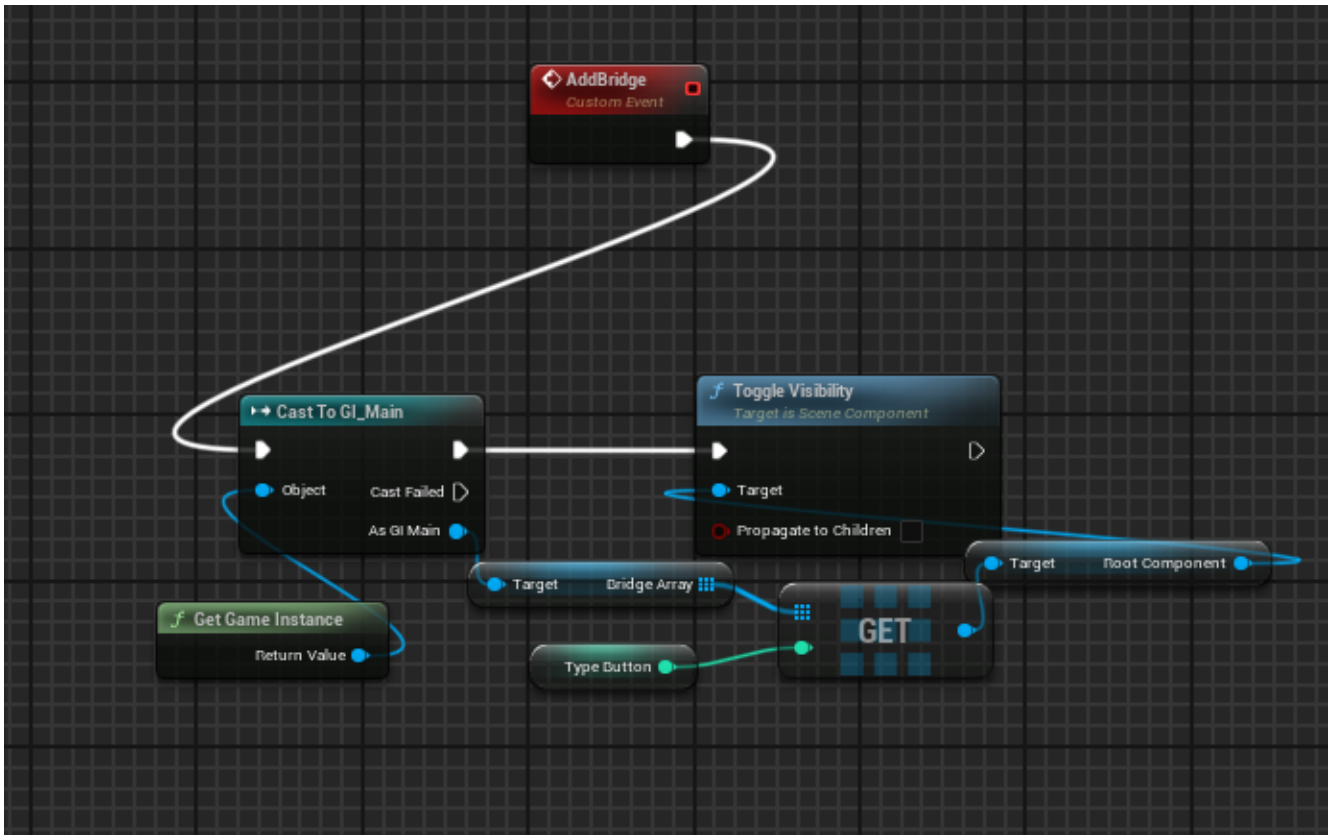


Рисунок 4.39 – Алгоритм появи мосту

Данний алгоритм працює так само, але навпаки, видаляє міст (рис.4.40).

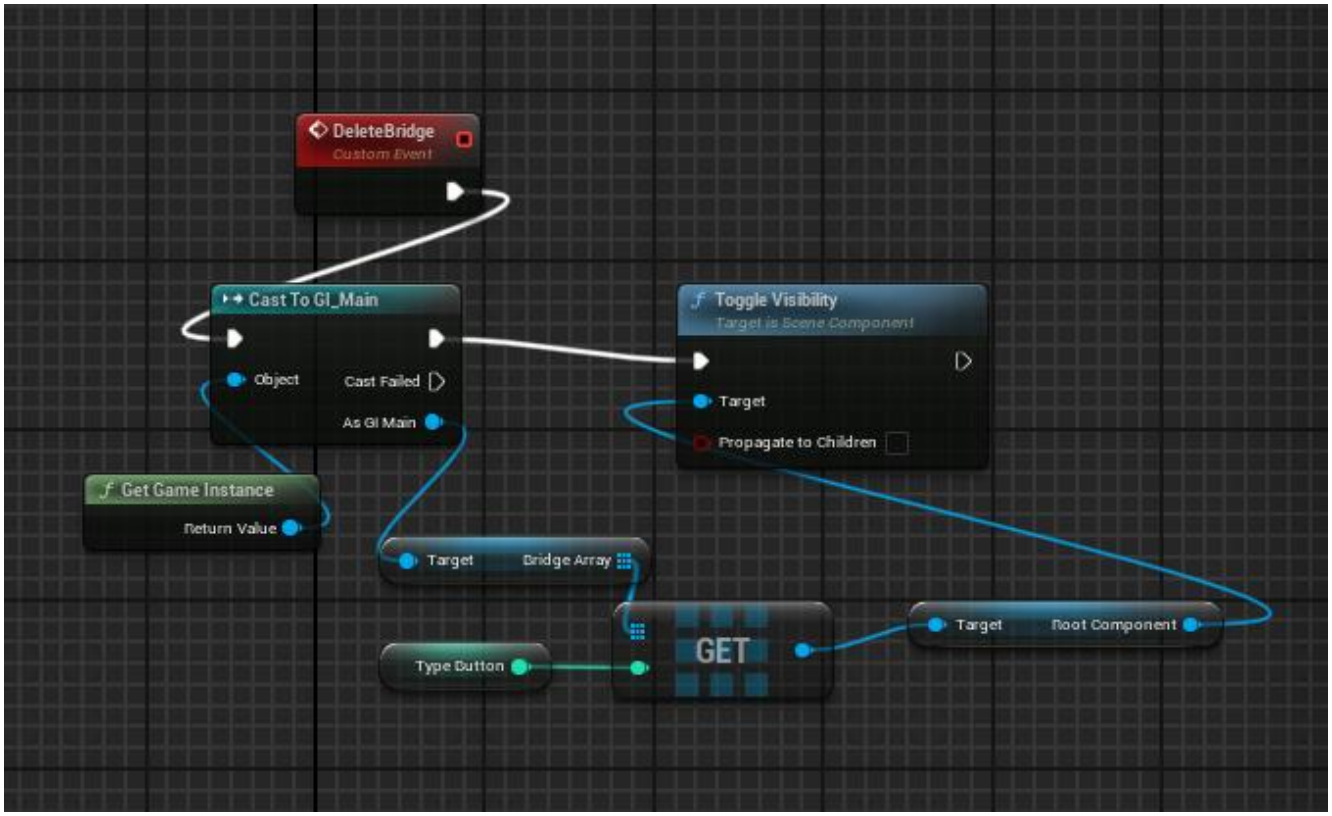


Рисунок 4.40 – Алгоритм видалення мосту

Головоломка з підбором букв для написання слова:

Це код кнопки для зміни букв при натисканні (рис.4.41).

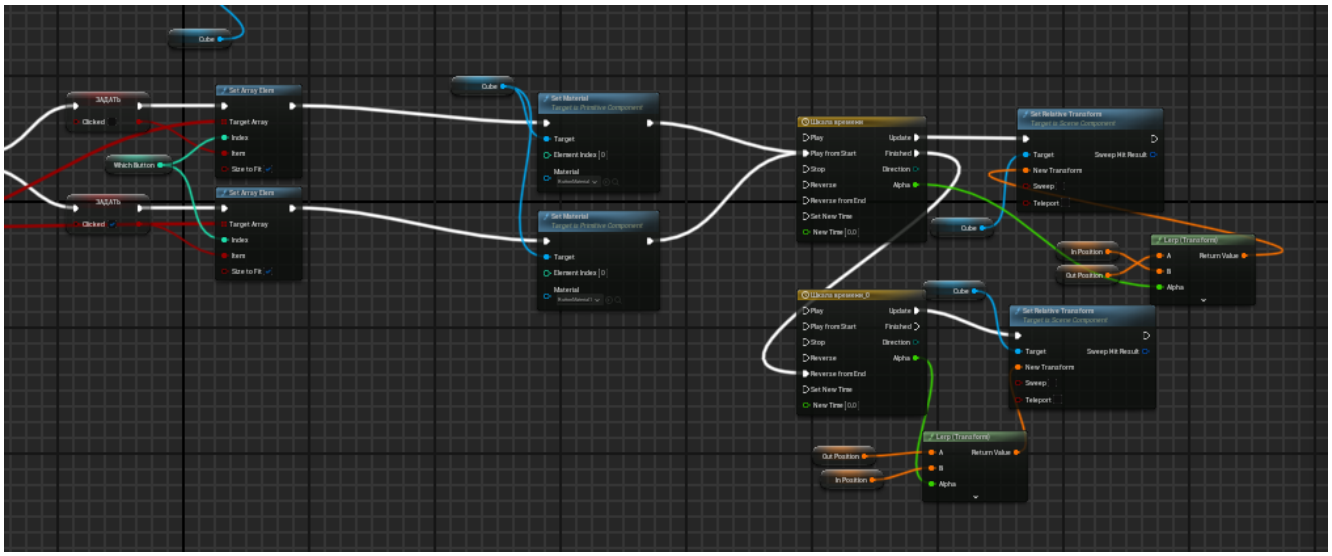


Рисунок 4.41 – Блюпринти даної головоломки

Даний алгоритм перевіряє правильність виконання головоломки (рис.4.42).

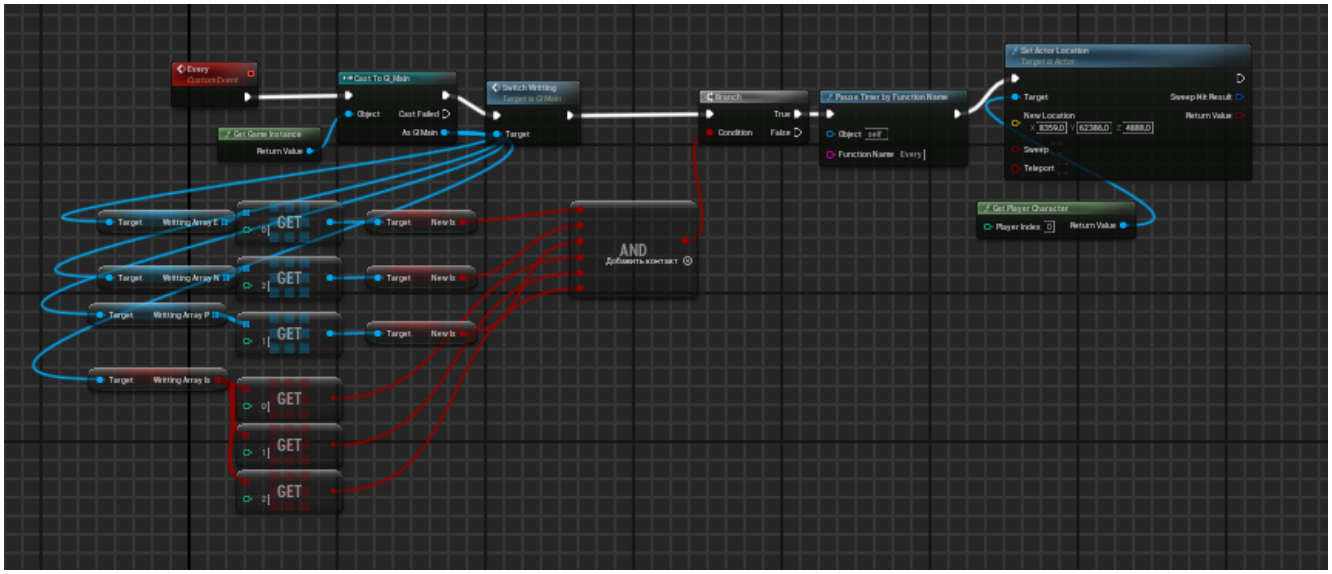


Рисунок 4.42 – Алгоритм перевірки умов

Дані функції змінюють літери за таймером у даній головоломці (рис.4.43).

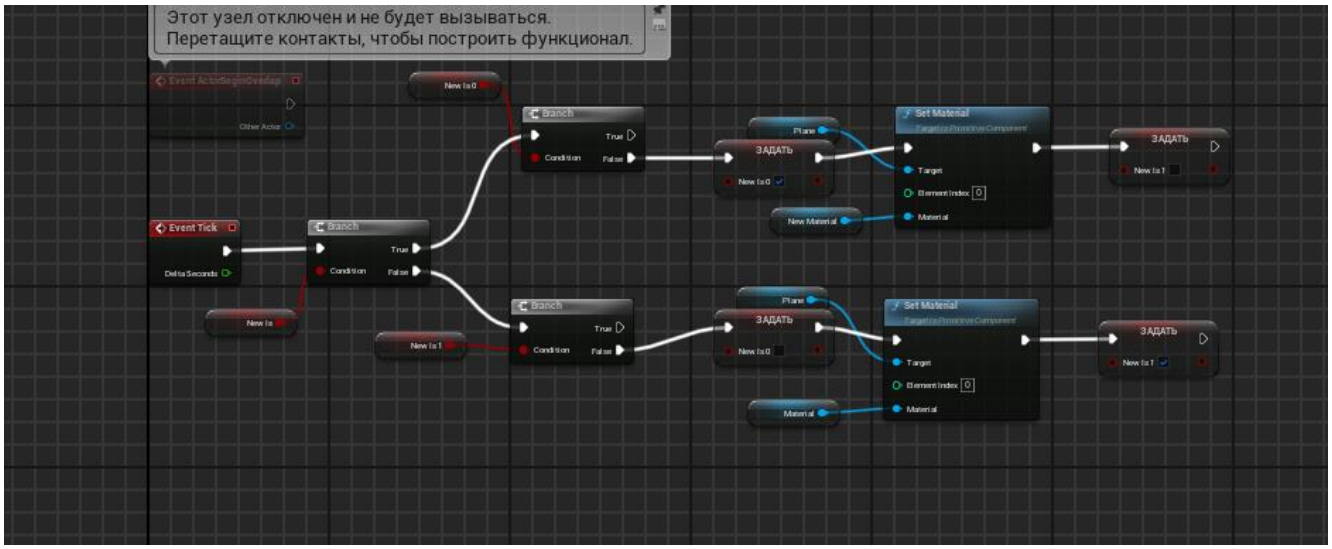


Рисунок 4.43 – Зміна маеріалів/літер

Головоломка із кубиків з сумою 15 у ряду (рис.4.44):

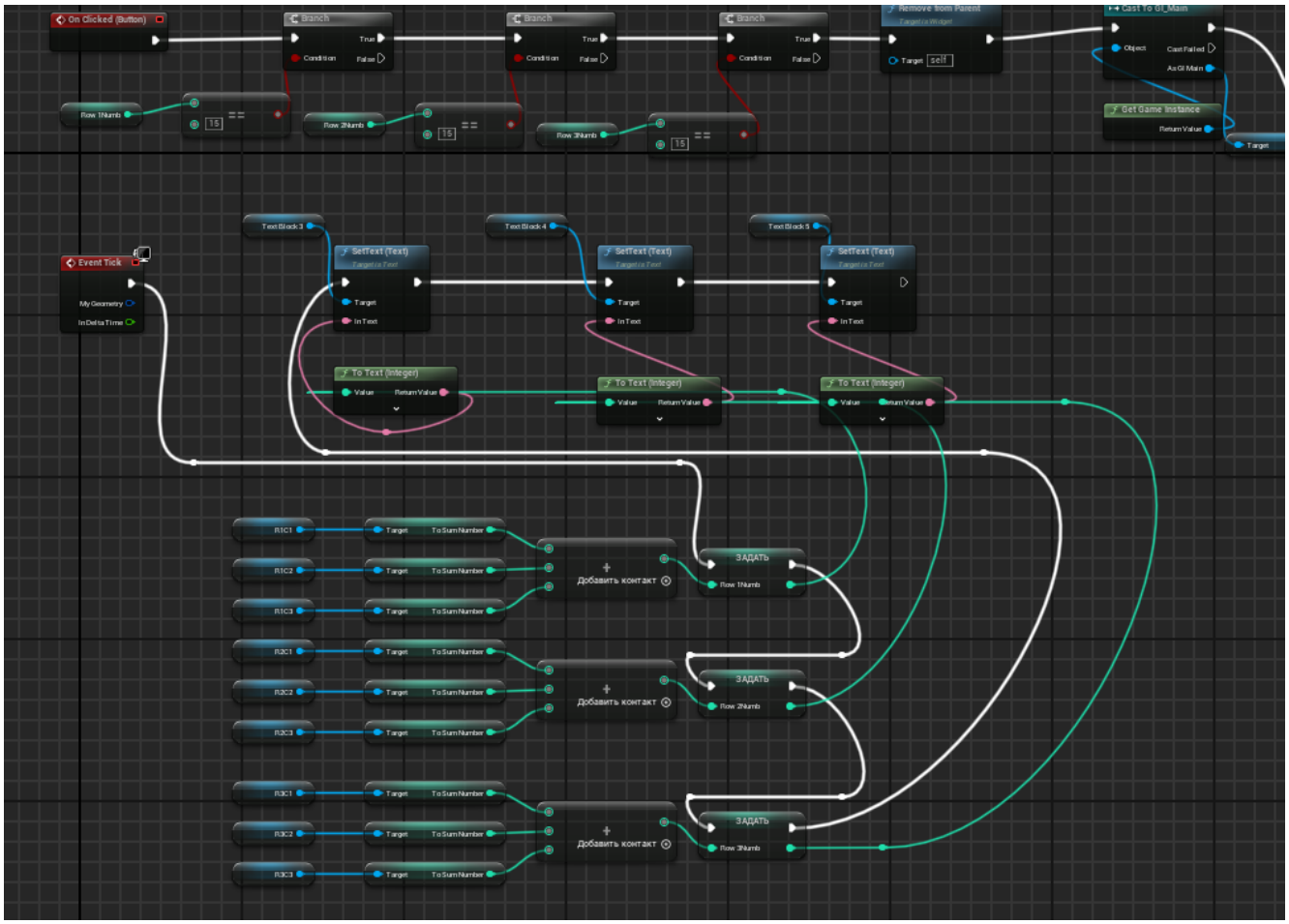


Рисунок 4.44 – Блюпринт даної головоломки

За допомогою цього алгоритму відбувається перевірка математики. Якщо під час перевірки всі умови виконано правильно, то виконується цей скрипт і рівень пройдено (рис.4.45).

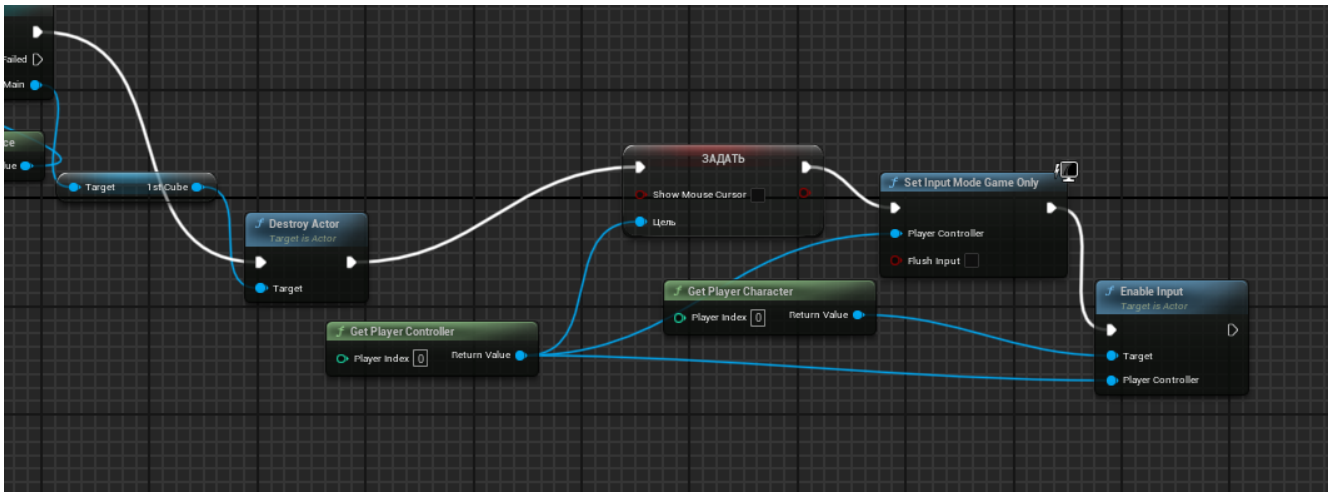


Рисунок 4.45 – Перевірка математики

8. Кодинг та скрипти

- Реалізація логіки: деякі складні механіки (наприклад, генерація випадкових кодів для головоломок) реалізовано на C++.
- Створення функцій: написано функції для перевірки статусу виконаних завдань, управління часом і геймплеєм.

9. Постобробка

- Постефекти: додано Bloom, Color Grading і Ambient Occlusion для покращення візуального вигляду.
- Шейдери: використано кастомні матеріали для створення унікального стилю гри.
- Остаточна оптимізація: зменшення ваги текстур, налаштування Level of Detail (LOD) для моделей.

10. Тестування та фіналізація

- Проведено тестування гри на різних пристроях для виявлення помилок.
- Додано навчання для користувача та фінальний рівень з бонусами.

ВИСНОВКИ

У роботі проведено дослідження та реалізацію інтерактивної, розвивальної гри, створеної з використанням сучасних технологій Unreal Engine, Blender та Adobe Photoshop. Було проаналізовано ключові аспекти проектування та розробки таких ігор, які спрямовані на поєднання навчального та розвивального контенту з ігровою формою.

У рамках дослідження було створено набір головоломок, які сприяють покращенню когнітивних здібностей дітей, водночас забезпечуючи високий рівень занурення завдяки інноваційним технологіям та сучасному дизайну.

Було враховано основні принципи створення інтуїтивно зрозумілого інтерфейсу, включаючи:

-Перцепцію: забезпечення комфортного сприйняття візуальних елементів;

-Інтерактивність: підтримка зручної та логічної взаємодії користувача з грою;

-Зрозумілість: доступність контенту для різних вікових груп;

-Естетичність: привабливий дизайн, що сприяє мотивації до навчання.

Додатково були досліджені можливості інклюзивного підходу у дизайні, що дозволяє врахувати потреби дітей із різними особливостями сприйняття. Впровадження таких елементів, як адаптивні кольори, спеціальні шрифти та чіткі підказки, забезпечує доступність контенту для широкої аудиторії.

Результати роботи показали, що поєднання ігрових та навчальних елементів у єдиній інтерактивній системі дозволяє досягти значного прогресу у розвитку дітей, відкриваючи нові горизонти для впровадження інноваційних методів освіти.

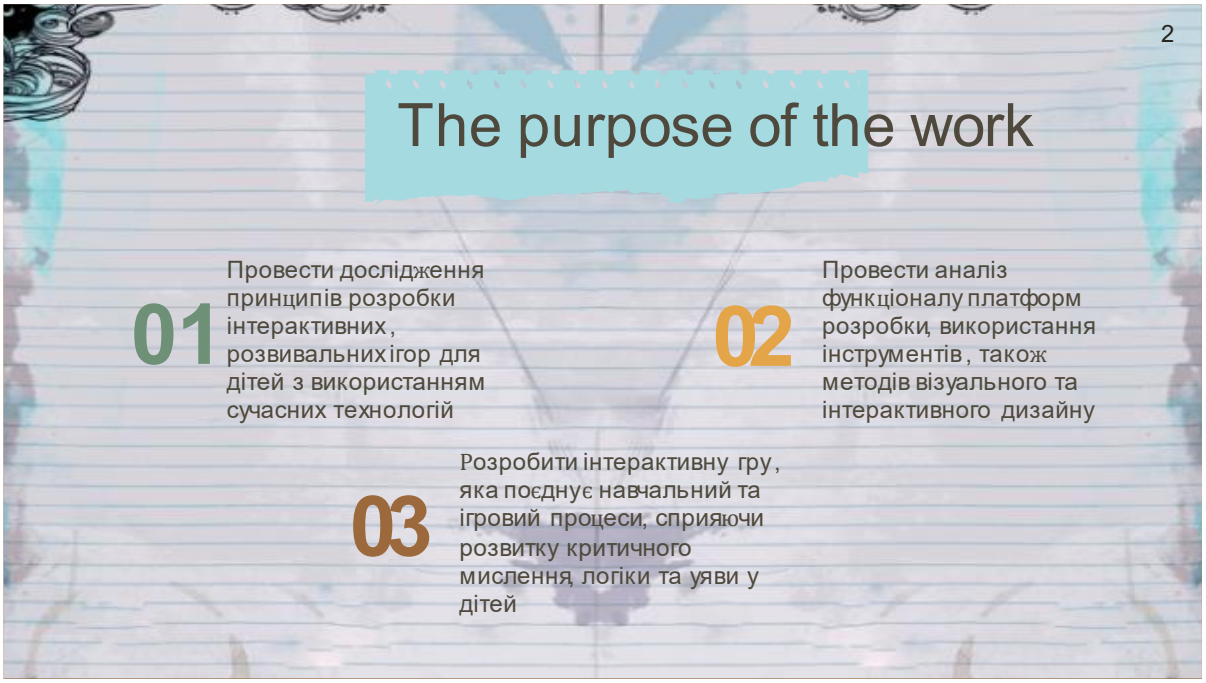
ПЕРЕЛІК ПОСИЛАНЬ

1. Джессі Шелл, The Art of Game Design: A Book of Lenses, Second Edition), Morgan Kaufmann, 2014
2. Schell, J. (2020). The Art of Game Design: A Book of Lenses. Fourth Edition. CRC Press,
3. Norman, D. (2013). The Design of Everyday Things - книга про принципи UX-дизайну і створення зручних інтерфейсів користувача.
4. Дослідження з освітніх ігор: статті про вплив інтерактивних ігор на процес навчання та розвиток когнітивних навичок у дітей:
5. Ю. В. Євдокимова, "Педагогічні ігри в освіті: від теорії до практики", Журнал педагогічних інновацій, 2021.
6. Биков В.Ю., Литвинова С.Г. "Інноваційні технології в освіті: теорія, практика, перспективи розвитку".
7. Kukulska-Hulme, A., & Traxler, J. (Eds.). "Mobile Learning: The Next Generation". Routledge, 2016.
8. Reimers, F. M., & Schleicher, A. "A Framework to Guide an Education Response to the COVID-19 Pandemic of 2020". OECD, 2020.
9. Thomas, M., Reinders, H., & Warschauer, M. "Contemporary Computer-Assisted Language Learning". Bloomsbury Academic, 2013.
10. Gee, J. P. (2007). What Video Games Have to Teach Us About Learning and Literacy. Palgrave Macmillan.
11. Unreal Engine Documentation [Електронний ресурс] : [Веб-сайт].- Електронні данні.- Посібник з розробки на Unreal Engine та використання Blueprint для створення логіки ігор - Режим доступа: https://docs.unrealengine.com
12. Blender Foundation [Електронний ресурс] : [Веб-сайт].- Електронні данні.- офіційні посібники з 3D-моделювання та текстурування у Blender - Режим доступа: https://www.blender.org
13. Чек-листи з UI/UX-дизайну [Електронний ресурс] : [Веб-сайт].- Електронні данні.- матеріали з проектування ігрових інтерфейсів - https://uxdesign.cc
14. interfaceingame.com [Електронний ресурс] : [Веб-сайт].- Електронні данні.- [Game interfaces for designers].- Режим доступа: https://interfaceingame.com/games/
15. gameuidatabase.com [Електронний ресурс] : [Веб-сайт].- Електронні данні.- [Game interfaces for designers].- Режим доступа: https://www.gameuidatabase.com/

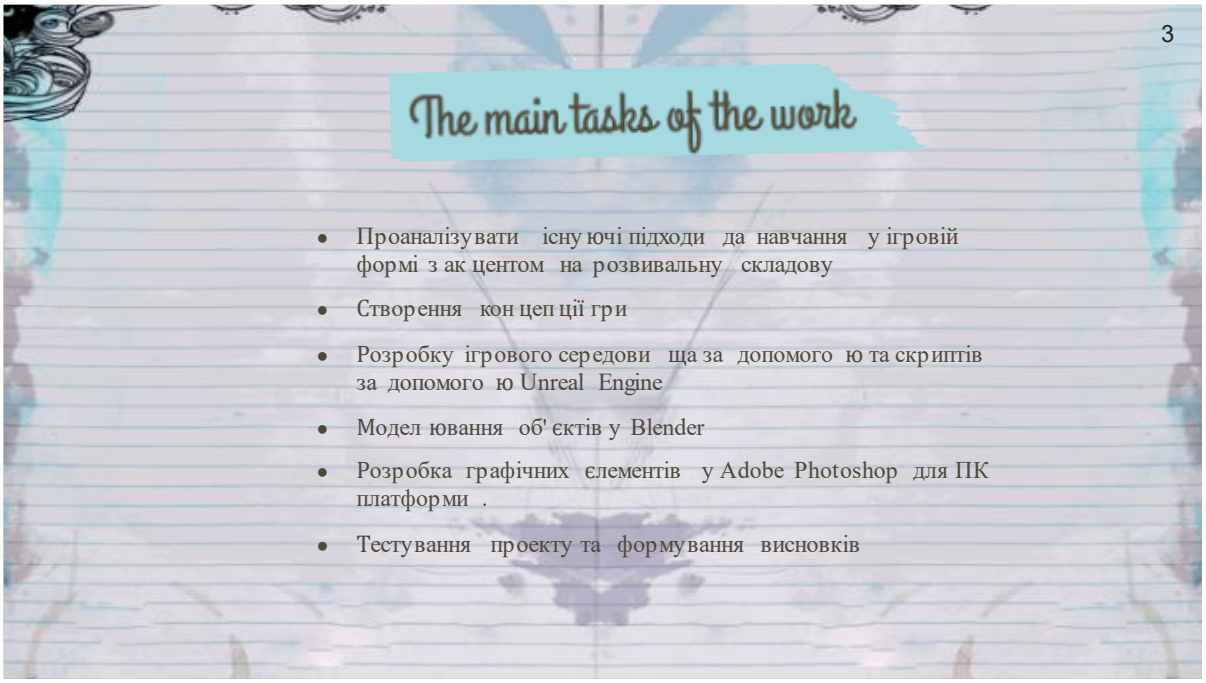
Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

Слайд 1



Слайд 2



Слайд 3

Слайд 4

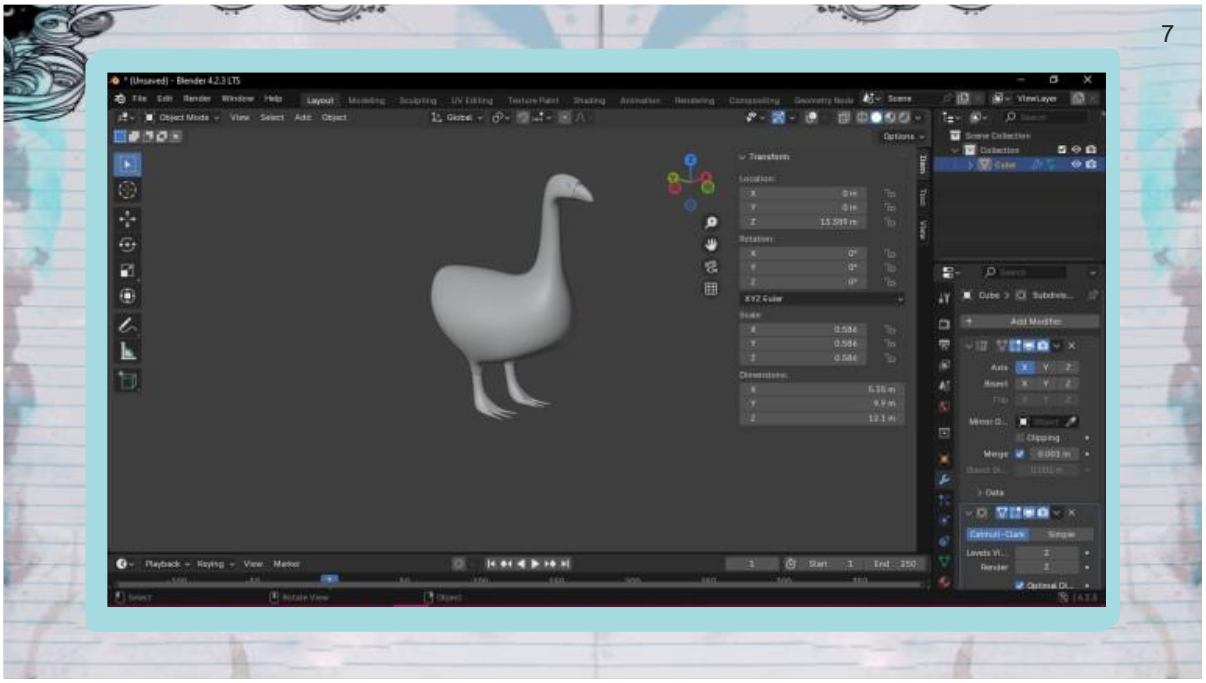
Problem solving

01 Розуміння принципів розробки інтерактивних, розвивальних ігор для дітей

02 Розуміння та правильне використання законів і правил геймдизайну

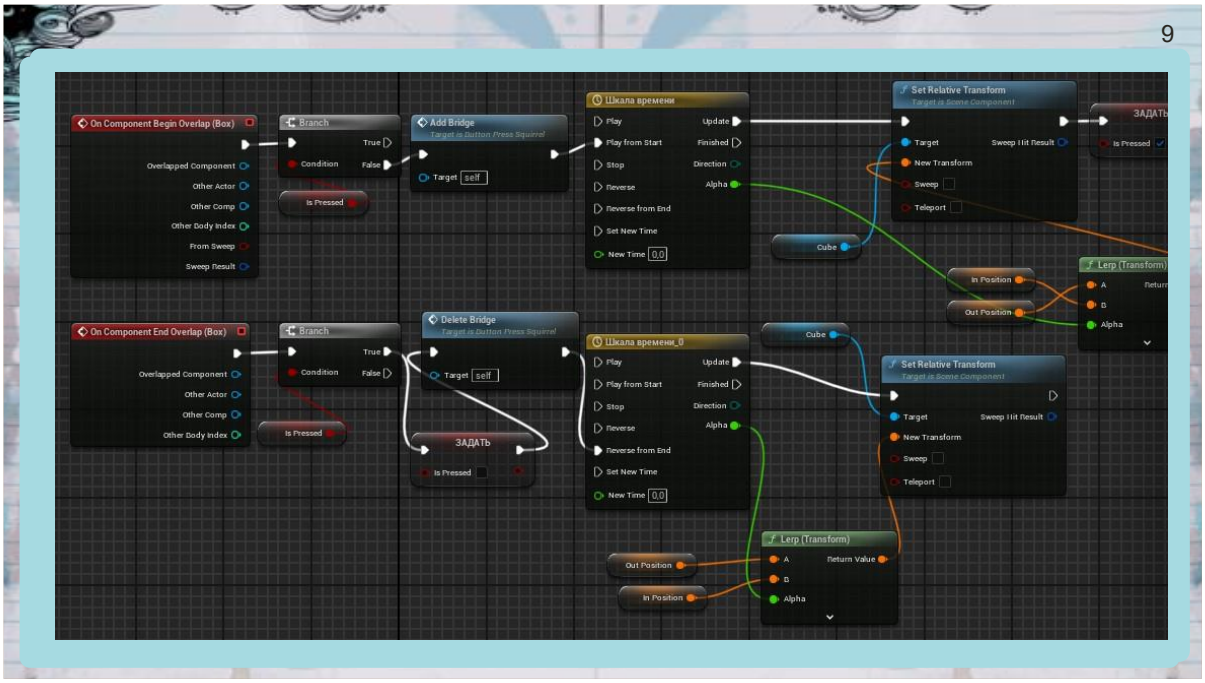
Слайд 5

Слайд 6

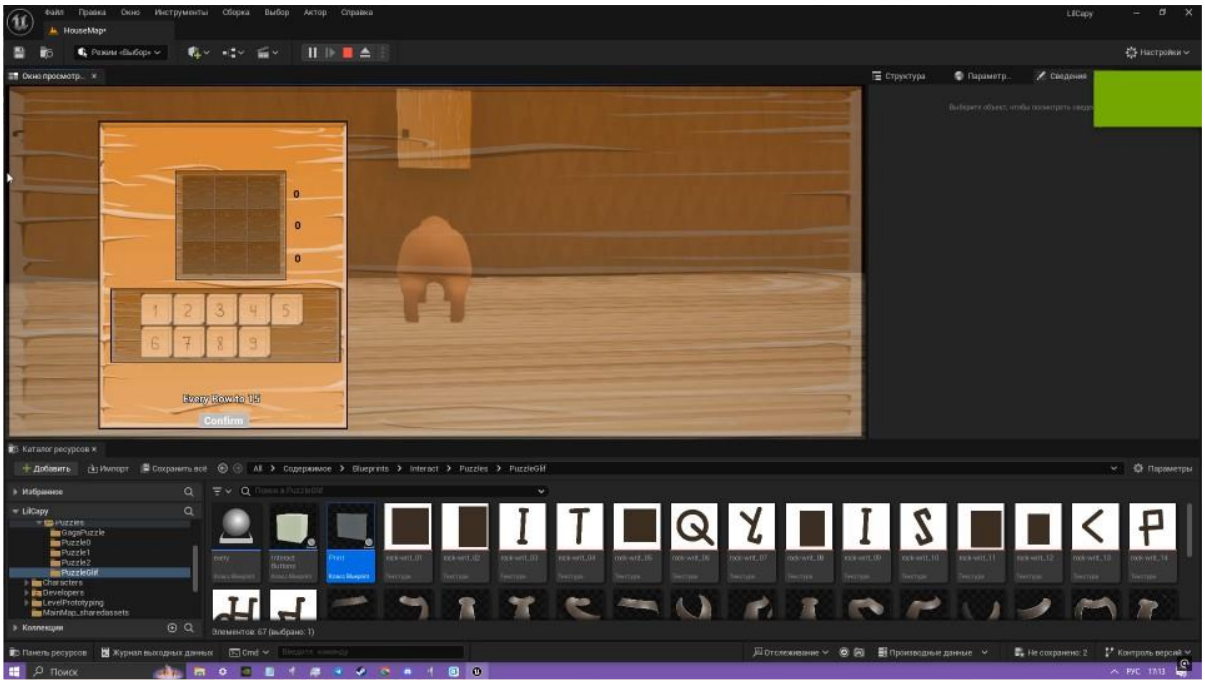


Слайд 7

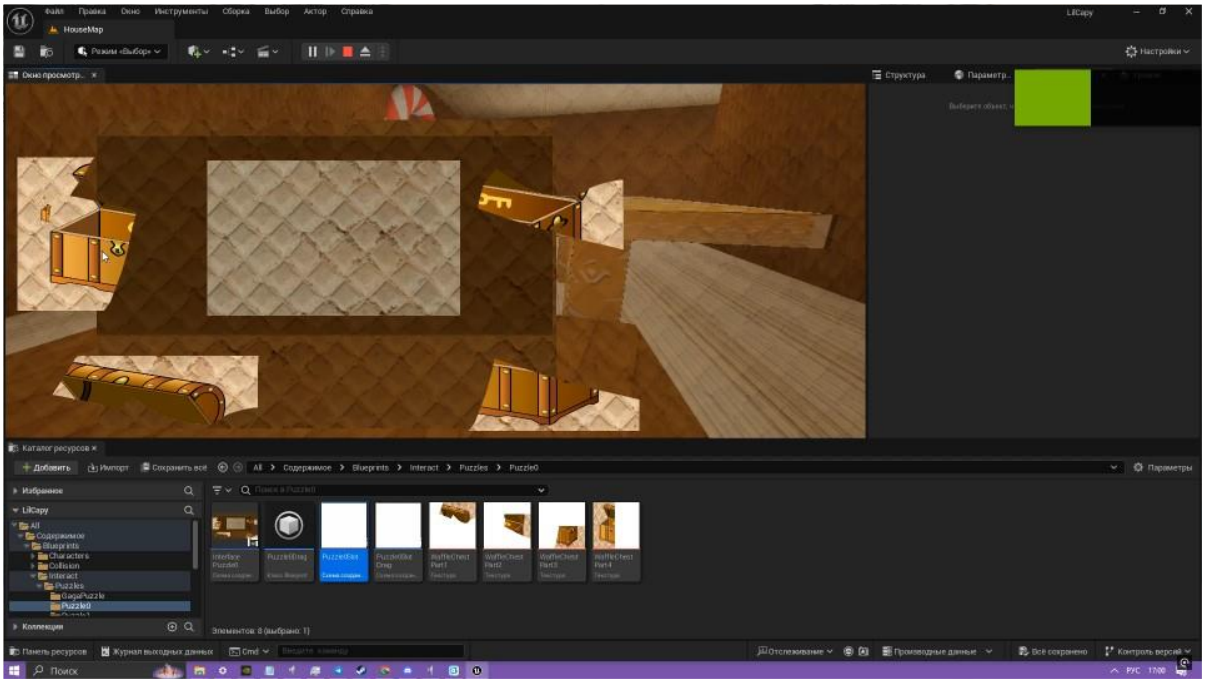
Слайд 8



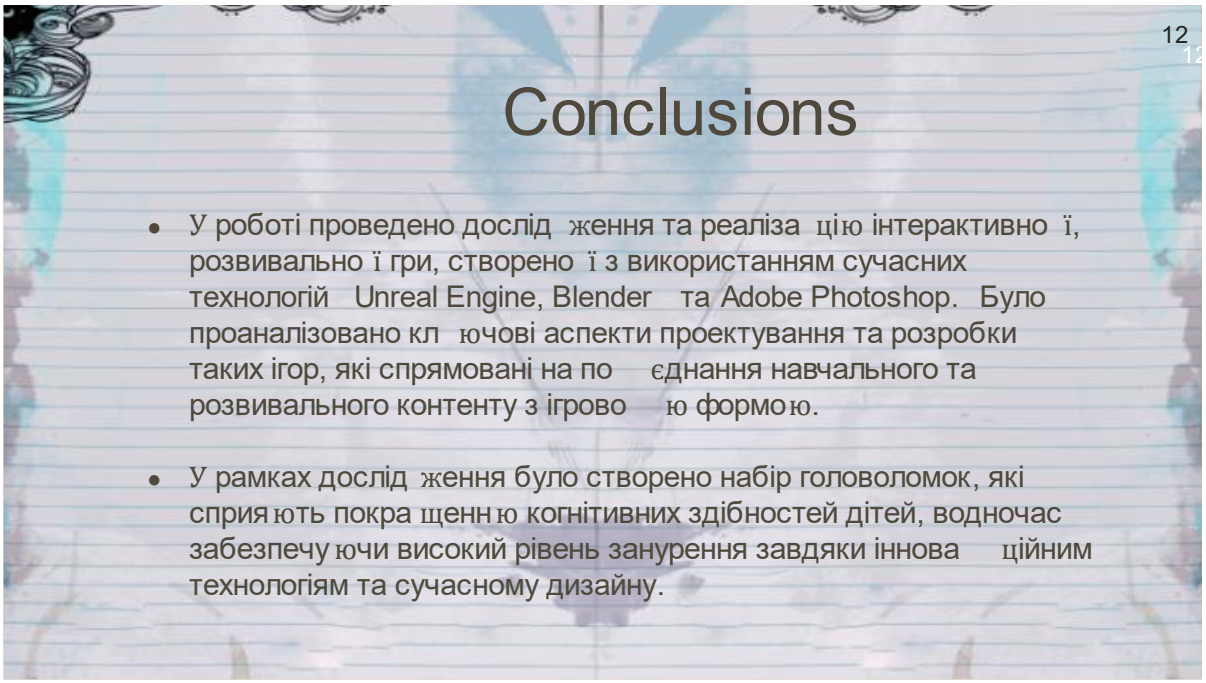
Слайд 9



Слайд 10



Слайд 11



Слайд 12