МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Розробка meta частини для гри на мобільних пристроях»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Тиркусова Н.В**

Студент групи ІН-71/2 — **Безпалий А.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-71 спеціальності "Комп'ютерні науки" денної форми навчання Безпалого Андрія Володимировича

Тема: " Розробка meta частини для гри на мобільних пристроях"

Затверджена наказом по СумДУ

№_____ от _____2021 р.

Зміст пояснювальної записки: 1) огляд проблемної області; 2) анализ аналогічних рішень; 3) постановка задачі; 4)вибір методу розробки проекту 5) реалізація проекту та перевірка отриманого результату роботи.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Тиркусова Н.В.

Завдання приняв до виконання _____ Безпалий А.В..

РЕФЕРАТ

Записка: 54 стор., 28 рис., 0 табл., 1 додаток, 16 джерел.

Об’єкт дослідження — Вивчення процесу взаємодія між розробником та користувачем шляхом додавання мета частини до коргри. Збереження та завантаження досягненого прогрессу на на призначених для користувача пристроях.

Мета роботи — Розробка мета гри, за для збільшення популярності продукту, шляхом додавання нового контенту, як наслідок збільшення реіграбельності та часу, який користувач всередньому проводить за грою

Методи дослідження — Реалізація MVVM моделі, на базі середовища розробки Unity та зберігання и заватаження данних прогресу користувача за рахунок JSON

Результати — Створена метагра, що є частиною когри, на базі середовища розробки Unity, котра надає змогу користувачеві здійснювати апгрейди всередині локації, з візуальним відображенням.

ЧАСТИНА МОБІЛЬНОЇ ГРИ, МУЛЬТИПЛАТФОРМЕНІСТЬ,
MVVM , UNITY, ДОДАТКОВИЙ КОНТЕНТ, C#,.JSON,
ВІЗУАЛІЗАЦІЯ АПГРЕЙДІВ

ЗМІСТ

ВСТУП

На сьогодні мобільний телефон є одним з найважливіших винаходів в житті людства. Прогрес не стоїть на одному місці. Зараз вже дуже складно когось вразити новинками техніки, проте ще кілька десятиліть тому середньостатистична людина навіть і уявити не могла про мобільний телефон. Завдяки мобільному телефону людина завжди на зв'язку, може ділитися своїми новинами з рідними та друзями. Мобільні телефони створюють усі умови за для того щоб наше життя стало комфортніше. Тепер немає необхідності бути прив'язаним до одного и того ж самого місця, якщо є бажання поспілкуватися з іншими людьми або ж з інтересом провести свій вільний час.

На сьогодні мобільний телефон використовує все більша кількість людей, незважаючи на вік. Така популярність обумовлена тим, що потужність та функціонал мобільних пристроїв невпинно росте. Якщо раніше перші мобільні телефони були здатні лише телефонувати, то зараз же потужність телефона в цілому не поступається декім комп'ютерам, а точніше за допомогою телефона можна:

- проглядати фільми;
- прослуховувати музику
- писати повідомлення;
- подорожувати світом;
- обмінюватись даними;
- використовувати мережу інтернет;
- спілкуватися в соціальних мережах;
- використовувати різноманітні додатки;
- грати у ігри.

Зупинимося на останньому пункті більш детально. Одна з перших компаній, яка почала випускати ігри була Nokia, найвідомішою з них з повною впевненісттю можна назвати “Snake”, тобто класична змійка, вже після цього з’явилися й інші ігри, від інших компаній.

Мобільні ігри здебільше необхідні для того, щоб цікаво провести час, наприклад під час поїздки у транспорті, стоянні у черзі чи у вільний час. Поступово ігри еволюціонували, завдяки технічному прогресу.

З кожним днем кількість мобільних ігор зростає. Потужність сучасних телефонів дозволяє реалізовувати найсміливіші проекти, котрі, зазвичай, потребують чималих фінансових ресурсів та час на розробку проекту.

Конкуренція та боротьба за увагу користувача змушує розробників ускладнювати свій майбутній продукт, додаючи до нього унікальний контент та нові можливості, саме те що може зацікавити нового користувача. Але тільки зацікавити на сьогодні недостатньо, набагато важливіше зробити усе можливе щоб утримати користувача на більший час ніж 5-10 хвилин ігрового процесу.

У рамках цієї дипломної описано процес створення Мета частини до мобільної гри, котра збільшить кількість, доступного для користувача, контенту та в свою чергу підвищить рівень зацікавленості проектом , що в свою чергу дозволить збільшити середній час, який користувач у зазвичай проводить у грі за мобільним телефоном.

Це одна з основних задач ігрових проектів, бо чим більше людина проводить час за грою тим більше прибутку приносить, що в свою чергу дозволяє компанії підтримувати та вдосконалювати свій продукт, таким чином збільшуючи збільшити популярність проекту та посприяє його поширенню.

Так як гра поширюється за моделюю Free To Play, збільшення часу, що користувач проведе у грі є однією з ключових здач.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Мета частина: загальне визначення

Метагра (Мета) - в комп'ютерних та мобльних іграх існує поняття, яке описує можливі активності користувача поза основною грою, але, які впливають на її ігровий досвід та процес.

Існує багато трактувань визначення метаігри. В одному випадку може матися на увазі, що це механіки, які в основному своєму розумінні спрямовані на реалізацію цілей розробника, а не збагачення ігрового досвіду користувача. Окрім цього, існує думка що метаігра передбачає введення розробником додаткових можливостей, які, важливо зазеначити, не є частиною основного ігрового процесу. Наприклад:

- Різноманітні блоги;
- Формування вікі-проектів;
- Створення ігрових форумів;
- Можлівість костюмізуваті свого персонажа чи довкілля.

Завдняки додаванню, зазначеному вище, функціоналу відбувається спілкування та обмін досвідом між гравцями, що нерідко сприяє зростанню зацікавленості проектом.

У порівнянні з основним ігровим процесом (Коргра) метагра зазвичай несе другорядне значення, проте її наявність, різноманітність та якість безпосередньо впливають на реіграбельність проекту та кількість часу, який користувач витратить до того моменту як втратить інтерес. Як наслідок користувач скоріше за все видалить гру.

1.2 Огляд проблемної області

На сьогодні метагра наявна мало не в кожному ігровому проекті, тому дуже важко виділити основі проблемні області, проте якщо проаналізувати основні ігрові проекти в окремих жанрах, за різною складністтю та з різним бюджетом, то можна виділити основні проблеми, котрі варто враховувати.

По-перше, перед тим, як створювати мета частину необхідно провести детальний аналіз ринку для того щоб виявити актуальну тему. Наприклад, при нинішніх обставинах, враховуючи пандемію коронавіруса та карантинні обмеження, зросла зацікавленість, з сторони гравців у апакаліптичних темах, зокрема в темах пов’язаних із зомбі чи вірусами. Також необхідно враховувати тривалість тендеції, та робити розрахунки з урахуванням часу, котрий буде витрачено на розробку .

Виявлення ключових інтересів для користувача є одним із ключових моментів, бо чим цікавіша тема тим більша кількість користувачів буде зацікавлена у продукті .

По-друге, створення мети у відриві від коргри, коли метагра не має смислового наслідування від теми основного геймплея. Мета має бути побудована таким чином, щоб у разі, якщо користувачу набридне основний ігровий процес, він мав можливість переключитися на мету и навпаки. Тобто мів двома частинами однієї гри повинен бути логічний взаємозв’язок, як на прикладі вітрового вітряка, де кожна лопать доповнює попередню и працює як єдиний механізм.

По-третє, одним з не менш важливих факторів можна вважати варіативність контенту, необхідно додавати новий контент та видозмінювати існуючий для того, щоб користувач мав змогу отримувати новий досвід, бо інакше швидко втратить інтерес у цілому до проекту.

1.3 Огляд подібних реалізаций метагри

У зв’язку з обраною темою для диплому було проведено дослідження, метою якого було виявлення та аналіз проектів, у котрих реалізована мета частина гри. Отже в результаті аналізу було знайдено та проаналізовано декілька проектів, що мають найбільшу актуальність на сьогодні. Такі як:

1. Gardenscapes;
2. Project Makeover;

Gardenscapes – це мобільна гра, яку випустила компанія Playrix. Гра поєднує в собі одразу дві повноцінні механіки Match-3 і сюжетну лінію. Gardenscapes реалізує в собі історію дворецького по імені Остін, котрий обрав своєю задачею відновити старий фамільний сад.

Рисунок 1.1 - Інтерфейс мета частини «Gardenscapes»

Рисунок 1.2 - Інтерфейс кор частини «Gardenscapes»

Інтерфейс мета частини зображений на Рисунку 1.1, в даному випадку задача мета частини полягає в наданні користувачу витрачати ресурси, що були накопичені під час гри у основний геймплей, котрий зображено на Рисунку 1.2. Під час дослідження продукту значних недоліків визначено не було, проте слід зазначити що, на мою думку, потрібно витрачати занадто багато часу, щоб пройти один повний ігровий цикл, та внести бажаємі зміни до конфігурації саду.

Project Makeover- це мобільна гра, котра випущена компанією Magic Tavern. Основна задача гри полягає у вирішенні головоломок в котрих

необхідно створити відповідний образ своєму герою за допомогою зміни одягу, макіяжу чи інтер'єру кімнати.

Рисунок 1.3 - Інтерфейс мета частини «Project Makeover»

Рисунок 1.4 - Інтерфейс мета частини «Project Makeover»

У результаті дослідження даного продукту, виявлено ряд недоліків серед яких: UI, що зазначений на Рисунку 1.3, який дуже важко сприймається оком так як, білі кнопки знаходяться на білому фоні, другий недолік це незакономірне сростання складності ігрових рівнів, від чого втрачається інтерес до проекту.

1.4 Аспекти розробки метагри

Якщо проаналізувати ігри з самого їх зародження, то прослідковується певна закономірність ігри розвивалися насамперед в тому напрямку у котрому розробники отримували гроші за користування своїм. В залежності

від типу монетизації формувався своєрідний набір мета-ігор, що, в свою чергу впливає на сприймання гри в цілому.

Основні типи ігор, котрі впивають на монетизацію:

- аркадні автомати;
- консольні ігри;
- ігри формату “Free to Play”.

Аркадні автомати. Протягом довгого часу ігру монетизувалися за принципом “P2P”(Pay to Play), тобто перед тим як граты необхідно було заплатити. У більшості своїй перші ігри виходили на аркадних автоматах, що и мало безпосередній вплив на формування типу метагри. Розробники ставили собі задачу таким чином, щоб користувач був зацікавлений пограти и через деякий час мав бажання повторити. Однак якщо на самому початку ці задачі реалізовувалися лише за рахунок коргри, то з появою Space Invaders, що вишла у 1978 році, жагу до повернення формувала звичайна таблиця лідерів, Рисунок 1.5. Такий крок одразу збільшив реіграбельність, бо кожен хотів встановити особистий рекорд. Тобто аркадні автомати використовували базову механіку метагри .

Рисунок 1.5 - Інтерфейс мета частини «Space Invaders»

Консольні ігри. Якщо консольні ігри порівнювати з аркданими, то основна відмінність буде полягати в тому, що за консольну гру користувач мав змогу заплатити лише один раз, на відмінно від аркадного автамота, де користувач платив за кожну окрему сесію тому принцип метагри був змінений. То що ж тепер насамперед було необхідно розробникам? Перш за

все, це зацікавити майбутнього гравця у покупці саме свого продукту. По-друге, надати продукту відповідну цінність, щоб користувач залишився задоволеним та надалі купував продукцію виробника. Важливо зазначити, що раніше розробники прагнули робити ігри, котрі користувач міг би проходити 15 , 25, а то и більше 30 годин, проте з часом цей показник зменшився и наразі мінімально допустимою тривалістю вважається гра, на проходження якої, необхідно від 8-ми годин. Збільшити час, проходження гри можна за допомогою додавання частин метагри, наприклад: у грі Red Faction: Guerrilla існує 6 секторів, кожен з яких необхідно відвоювати, але гравець не зможе зробити це у зручній для нього послідовності, а лише у строго обраній розробником послідовності, що в цілому збільшує час, котрий користувач витратить на повне проходження.

У іншому випадку на користувача накладаються обмеження у вигляді лімітованої кількості спроб, що у разі їх закінчення, призведе до повторного проходження. Додавання на локацію різноманітних бонусів чи “секретних” місць, які замасковані и щоб їх знайти необхідно витратити додатковий час.

Існує ще безліч прикладів, але спільне у всіх них одне – це збільшення часу, що необхідне на проходження за рахунок додавання елементів мета гри, які не мають значного впливу на саму гру, а лише роблять її більш цінною.

Останній тип – це відеоігри формату “Free to Play”. Головною ознакою даного формату є те, що користувач може грати безкоштовно. Тож у розробника я декілька основних цілей. По-перше зацікавити користувача встановити продукт. по-друге, як завжди забезпечити реіграбельність продукту, по-третє, створити платний контент, котрий зацікавить користувача у його придбанні.

Для реалізації першого пункту необхідно створити проект із самодостатнім базовим ігровим процесом, що вже сам по собі може зацікавити користувача.

Для вирішення другого пункту, зазвичай використовують систему нагородження бонусами, наприклад, коли користувач заходить у гру кожного дня протягом одного тижня, то отримує приємний бонус, наступний бонус може очікувати його після того як пройде місяць щоденної гри. Однак найдієвішим методом можна вважати продумане обмеження ігрового процесу за допомоги таймерів, наприклад: надання користувачу лише п'ять спроб зіграти ігровий цикл, кожна втрачена одиниця енергії відновлюється одну годину реального часу, в результати користувач витратить усі п'ять спроб за півгодини и тільки через годину зможе зіграти ще однин ігровий цикл. Така техніка може використовуватися для будь-якої механіки, таким чином ми одразу даємо змогу користувачу заплатити реальні гроші, щоб він міг отримати миттєву змогу продовжити гру, прискорити будування будівлі, чи швидкість росту рослин на фермі.

Рисунок 1.6 - Інтерфейс мета частини, очікування нової спроби

Free to Play ігри у своєму базовому ігровому процесі зазвичай мають невелику кількість основних механік, навпаки в основному мають в собі завелику кількість мета ігрових механік.

1.5 Постановка задачі

Мета роботи – розробка мета-частини, функціональні можливості якої збільшать зацікавленість користувача продуктом та збільшить середній час однієї ігрової сесії. етагра являє собою одну велику локацію в яку користувач потрапляє після завантаження гри та після закінчення ігрової сесії. Всередині локації, котра представляє з себе підвал парижского Нотр-Дама. Користувач

заробряє внутрішньоігрові гроші та має можливість витратити їх у реставрацію приміщення, важливо зауважити, що користувач може реставрувати елементи підвалу лише в певному порядку. Після того як користувач підтвердив свій намір проапгрейдить об’єкт.

Етапи створення метагри:

- створення ігрової сцени;
- створення усіх необхідних анімацій;
- створення алгоритму збереження та завантаження локації;
- створення алгоритму, котрий відповідає за апгрейд локації;
- створення необхідних UI елементів.

Створення ігрової сцени передбачає розміщення усіх необхідних спрайтів. Та підготовка їх до подальшої участі у анімаціях.

Створення анімацій необхідне, щоб додати різноманіття та динаміки.

Алгоритму збереження та завантаження локації відповідає за збереження прогресу користувача.

Алгоритм апгрейду локації необхідний для коректного відображення апгрейду, апгрейд відбувається для потрібного об’єкта в правильний момент часу.Створення необхідних UI елементів необхідне для того щоб користувач міг взаємодіяти із грою, наприклад: переглянути апгрейди, чи перейти у коргру.

2. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Середовище розробки Unity

Для реалізації задачі було обрано середовище розробки Unity. Unity являє собою багатоплатформний інструмент, котрий використовується для розробки відеоігор і застосунків. Однією з переваг, що вплинула на вибір саме цього середовища, створені за допомогою Unity програми представляють з себе мультиплатформу, тобто можуть використовуватися на всіх платформах, починаю від комп'ютера, консолі, закінчуючи мобільним телефоном. З використанням двовимірної чи тривимірної-графіки, також нерідко за допомогою Unity створюються продукти с використанням віртуальної або ж доповненої реальності. Також менш важливою перевагою можна вважати наявність чималої бібліотеки асетів та плагінів, що у разі потреби може значно пришвидшити розробку продукту. До Unity Asset Store можна експортувати власні розробки.

Отже можна перерахувати наступні переваги Unity. По-перше використання у роботі компонентно-орієнтованого підхіду, по-друге, мультиплатформеність проекту, по-третє наявність відритої бібліотеки Unity Asset Store. Серед недоліків варто зазначити необхідність додаткової оптимізаціїї пам'яті, котру займає готовий білд, бо навіть нескладна двомірна гра може займати кількасот мегабайтів місця, що забагато для мобільної гри.

Під час розробки проекту створюються окремі сцени, тобто рівні, що являють собою окремі файли, що містять усі сценарії, об'єкти та налаштування. Сцени зазвичай складаються із власних об'єктів, що мають моделі, та об'єктів, що моделі не мають. До об'єкту є можливість додати необхідні, в залежності від задачі, компоненті та скрипти. До кожного об'єкта можна додати свій власний тег, та налаштувати порядок відображення. у сцені.

2.2 Використання патерну MVVM

MVVM – це один із шаблонів проектування архітектури додатку, призначений для того, щоб розділити логіку та інтерфейс користувача.

Якщо за приклад взяти розробку невеликого проекту, то в більшості випадків достатньо прямолінійного підходу, але якщо до функціоналу додаються нові можливості, ускладнюється вже існуюча логіка то все це в кінцевому результаті презвиде до втрати контролю над елементами програми і вже легше та доцільніше буде переробити структуру.

Суть роботи MVVM полягає у двостороннє зв'язуванню Model з View Model та View Model з View, котра зображена на Рисунку 2.1.

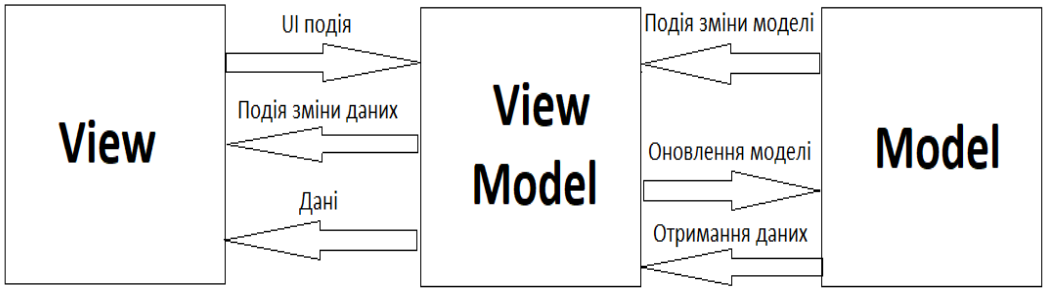

Рисунок 2.1 - Зображення взаємодії між компонентами MVVM

Model представляє з себе логіку роботи з даними та опис основних даних, котрі знадобляться для подяльшої роботи програми. У випадку з моєю задачею Model зберігає у собі дві змінні, типу int та string, котрі зберігають у собі рівень, і назву об'єкту, з яким буде відбуватися апгрейд, відповідно

View Model координує взаємодію управління з Model. За допомогою View Model відбуваються перетворення та маніпуляції з даними для того, щоб використовувати їх в подальшому. Також завдяки View Model нерідко реалізують додаткові властивості, котрі відсутні в Model. У моїй роботі View Model відповідає за перевірку, що визначає, який об'єкт зараз наступний у черзі на апгрейд.

View виконує роль відображення та обробки візуальних елементів сцени, таких як анімацій чи переходів, зазвичай у View не відбувається жодних логічних розрахунків. В моїй роботі за рахунок View відбувається відображення групи відповідних до апгрейду анімацій, та коректне відображення сцени, у разі, коли користувач запускає гру чи повертається до мета-локації після ігрового циклу.

Основною перевагою, яка вплинули на вибір даного патерну це його велика гнучкість, що значно спрощує та пришвидшує розробку.

Існує багаго інших структур, наприклад MVC патерн, основною особливістю даної структури є те, що елементи Controller и View залежать від Model, але в той же час Model не має ніякої залежності від цих двох елементів, як зображено на Рисунку 2.2

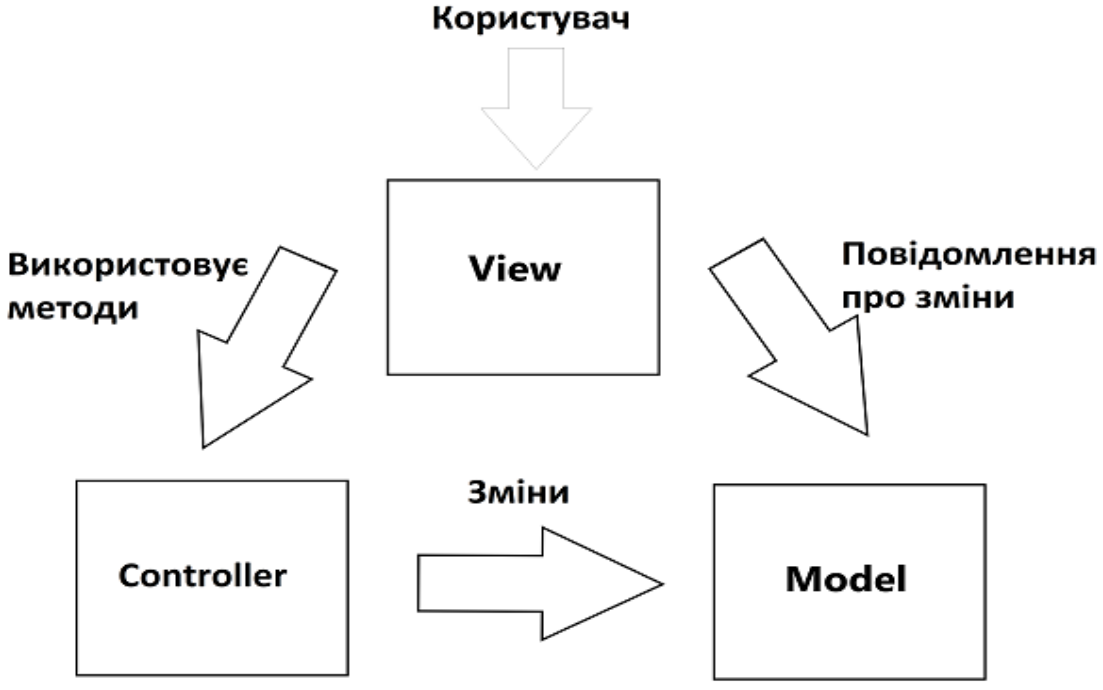

Рисунок 2.2 - Зображення взаємодії між компонентами MVC

Основна складність при роботі зі структрурою MVC – це занадто ускладений процес нарощування функціоналу, та використання чималої

кількості ресурсів, тому що блоки взаємодіють між собою виключно за рахунок передачі даних.

2.3 Збереження даних за допомогою JSON

JSON – це текстовий формат даних, котрий використовується майже у всіх скриптових мовах програмування.

Можна зберегти текстовий файл JSON не змінюючи при цьому його формат, тобто залишаючи його у форматі.json, і він буде відображатися у текстовому форматі. У якокстi значень в JSON можуть використовуватися: запис, неврегульована множина пар ключів, одномірний масив, що представляє з себе вже впорядковану множину пар ключів, ціле чи дійсне число.

На сьогоднішні JSON є одним з основних форматів для обміну даними та представлення складних структур. Тому мова програмування C#, я і багато інші, має вже вбудовану підтримку для роботи з JSON.

Об'єкт формату JSON має починатися і закінчуватися фігурними дужками “{}”. Всередині допускається дві та більше пар ключів/значень, х комою, котра їх розділяє. Для того, щоб відрізнити ключ від значення після кожного ключа має стояти двокрапка “:”.

Основною перевагою у збереженні даних за допомогою JSON у порівнянні з іншими методами можна вважати можливість використовувати складні структури в атрибута. Також в Unity є можливість реалізувати такі методи збереження та загрузки даних, наприклад BinaryFormatter, але серед основних недоліків даного методу це те, що: BinaryFormatter вважається методом, що не може забеспечити безпеку серіалізаці.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Створення ігрової сцени

Перш за все у процесі створення сцени необхідно розмістити усі заготовлені спрайти, приклад спрайтів зображений на Рисунку 3.1 у відповідному місці.

Рисунок 3.1 - Заготовлені для роботи спрайти

Після розміщення усіх спрайтів у сцені, необхідно коректно налаштувати слої між спрайтами, для їх коректного відображення. Наступний етап це формування відповідної ієрархії об'єктів у сцені, тобто створення відповідних дочірніх та материнських груп об'єктів, наприклад, як зображено на Рисунку 3.2.

Рисунок 3.2 - Побудова ієрархії об'єкта

Після проведення усіх необхідних маніпуляцій сцена набуде наступного вигляду, як зображено на Рисунку 3.3.

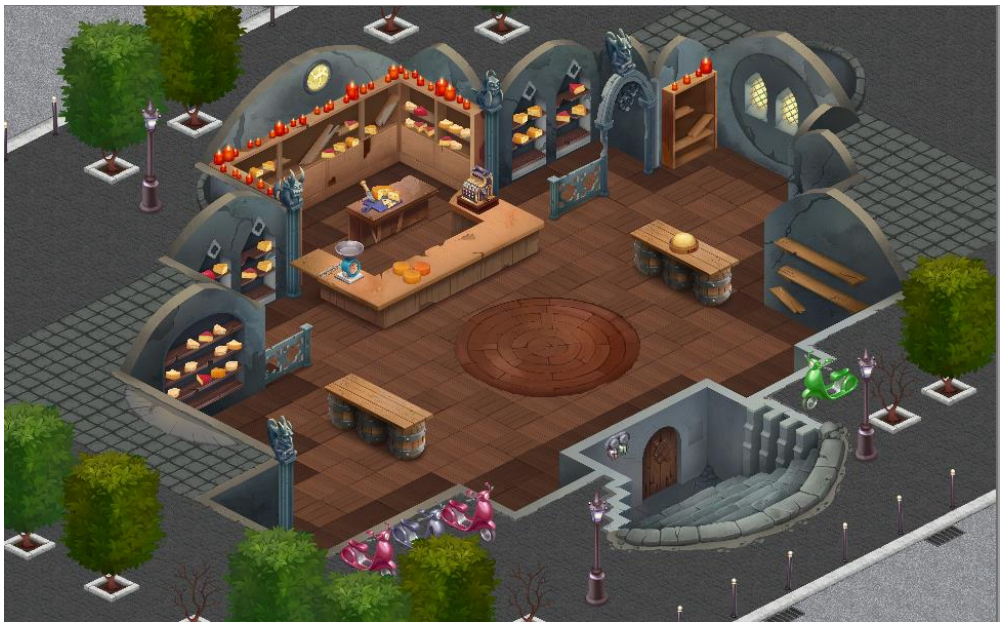

Рисунок 3.3 - Зображення локації у зібраному вигляді

3.2 Створення анімацій для апгрейдів

Для візуалізації апгрейду було використано метод Animation, який зображено на Рисунку 3.4, метод потребує створення анімаційного кліпу у якому можна змінювати інші компоненти об'єкту, таки як Transform, Sprite Renderer.

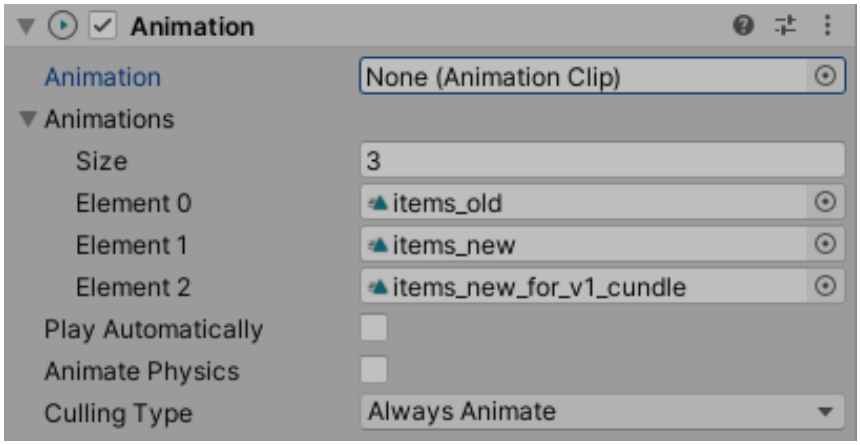

Рисунок 3.4 - Зображення локації у зібраному вигляді

Отже після створення усіх необхідних анімацій для комплексного апгрейду на наступний рівень та має наступний вигляд, зображений на Рисунку 3.5.

Рисунок 3.5 - Зображення процесу апгрейду

3.3 Створення алгоритму збереження та завантаження локації

3.3.1 Збереження даних

Збереження ігрової локації відбувається, при виході користувача зі гри або після переходу у коргру, за рахунок методу, який зображено на Рисунку 3.6.

```
15      public void Save(Model.Location location)
16      {
17          Location = location;
18
19          var path = Path.Combine(Application.persistentDataPath, PATH);
20          path = Path.Combine(path, "upgrade.json");
21
22          var serialized = JsonConvert.SerializeObject(Location, Formatting.Indented);
23          File.WriteAllText(path, serialized);
24      }
25
```

Рисунок 3.6 - Фрагмент коду методу зображення прогресу

Location, що знаходиться на Рисунку 3.7, в подальшому буде зберігається в JSON, та складається з InteriorObjects, це індекс наступного в черзі апгрейду, та масиву об'єктів, котрі потребуватимуть апгрейду.

```
public class Location
{
    public int CurrentUpgradeIndex;

    public InteriorObject[] InteriorObjects;
}
```

Рисунок 3.7 - Фрагмент коду класу Location

Path це змінна, що являє собою шлях, метод Path.Combine відповідає за об'єднаняя кількох строк у шлях, в моєму випадку це об'єднання шляху до індивідуальної папки, за допомогою методу та Application.persistentDataPath , та PATH. Слідом path прирівнюється до кінцевого шляху, де буде зберігатися JSON файл, для подальшого запису до файлу усієї необхідної інформації

3.3.2 Завантаження даних

Завантаження локації відбувається в момент, коли користувач має зайти до неї потрапити, під час заходження до гри чи повернення з коргри.

Цей процес відбувається за рахунок методу, який зображено на Рисунку 3.8

```
26      public void Load()
27      {
28          var path = Path.Combine(Application.persistentDataPath, PATH);
29          path = Path.Combine(path, "upgrade.json");
30          if (File.Exists(path))
31          {
32              var file = File.ReadAllText(path);
33              Location = JsonConvert.DeserializeObject<Model.Location>(file);
34          }
35      }
```

Рисунок 3.8 - Фрагмент коду методу завантаження прогресу

Після перевірки на явність файлу відбувається його зчитування та десеріалізація. В результати чого користувач бачить ту ж саму локацію, котру покидав в останній раз, це продемонстровано на Рисунку 3.9

Рисунок 3.8 – Збереження рівня апгрейду об'єктів

3.4 Створення алгоритму апгрейду локації

3.4.1 Створення Model

Model описує сутність логіки апгрейдів, тобто для того, щоб послідовність апгрейдів та заміна одного об'єкта іншим відбувалися коректно досить “знати” рівень апгрейлу конкретного об'єкта, Level, та його назву, Name, як це зображено на Рисунку 3.9

Послідовність, яка задає порядок пргрейду об'єктів знаходиться в окремому списку

```
public class InteriorObject
{
    public string Name;
    public int Level;
}
```

Рисунок 3.8 - Фрагмент коду, що описує модель

3.4.2 Створення View Model

```
private InteriorObject _model;


public event Action<InteriorObjectVM> Upgrade;
public event Action<InteriorObjectVM> UpgradeComplete;
```

Рисунок 3.9 – створення посилання на model та двох подій

```
public void UpgradeObject()
{
    _model.Level++;
    Upgrade?.Invoke(this);
}
```

Рисунок 3.10 – Фрагмент коду, що описує View Model

В методі, що вказананий на Рисунку 3.10 до нашої моделі додається один(рівень апгрейлу) та запускає подію Upgrade, Рисунок 3.9, котра в свою чергу є елементом View

3.4.3 Створення View

Коли користувач бажає зробити апгрейд, то він натискає на відповідну кнопку на своєму екрані та в цей момент відбувається апгрейл. В моєму випадку, View відповідає за відображення даного апгрейду та коректне відображення сцени в момент її завантаження. Варто зазначити, що

кожен об'єкт в сцені, що має рівні апгрейду має власну View, а отже і методи, що використовуються у кожному з випадків схожі.

На самому початку, коли об'єкт завантажується спрацьовує метод OnBindContext(), зображений на Рисунку 3.11

```
protected override void OnBindContext(InteriorObjectVM context)
{
    base.OnBindContext(context);
    InitView();

    context.Upgrade += Context_Upgrade;
}
```

Рисунок 3.11 – Фрагмент коду, що описує метод InitView

Якщо View Model викличе подію Upgrade, то метод Upgrade виконається. Тобто таким чином ми підписались на подію Upgrade, яку створили у View Model.

На рисунку 3.12 зображено метод InitView, що є елементом скрипта View для даного об'єкту, цей метод відповідає за відображення відповідного спрайта для об'єкта

```
protected override void InitView()
{
    if (Context.Model.Level == 1)
    {
        for (int i = 0; i < ObjectInSceneV1.Length; i++)
        {
            ObjectInSceneV1[i].SetActive(true);
        }

        for (int i = 0; i < AlfaInSeneV1.Length; i++)
        {
            AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
        }
    }
}
```

Рисунок 3.12 - Фрагмент коду, що описує метод InitView

Після того як даний метод спрацьовує відображається відповідний спрайт, в залежності від рівня апрейду конкретного об'єкта.

Якщо користувач вирів зробити апгрейд то відбудиться спрацьвування методу Context_Upgrade, який зображено на Рисунку 3.13

```
protected override void Context_Upgrade(InteriorObjectVM context)
{
    StartCoroutine(upgrade());
}
```

Рисунок 3.13 - Фрагмент коду, що описує метод Context_Upgrade

З урахуванням того, що під час одного апгрейду може відоброжатися одразу кілька десятків анімацій та з метою досягнення максимального згладжування між анімаціями використовується корутина. Тіло корутини зображено на Рисунку 3.14

```
IEnumerator upgrade()
{
    if (Context.Model.Level == 2)...
    else if (Context.Model.Level == 1)
        {
        for (int i = 0; i < Ver2.Length; i++)// ver_2 object on
        {
            Ver2[i].SetActive(true);
        }
        for (int i = 0; i < AnimV1.Length; i++)// v1 off
        {
            AnimV1[i].Play("standard_old");
        }
        for (int i = 0; i < AnimSup1.Length; i++)// items up
        {
            if (Items[i].gameObject.activeSelf)
            {
                AnimSup1[i].Play("items_up");
            }
        }
        yield return new WaitForSeconds(0.3f);//pause for v1-v2
        for (int i = 0; i < AnimV2.Length; i++)// v2 on
        {
            AnimV2[i].Play("standard_new");
        }
        yield return new WaitForSeconds(0.1f);//pause before items down
        for (int i = 0; i < AnimSup1.Length; i++)// items down after upgrade
        {
            if (Items[i].gameObject.activeSelf)...
        }
        for (int i = 0; i < Ver1.Length; i++)// ver_1 object off
        {
            Ver1[i].SetActive(false);
        }
    }
    yield return new WaitForSeconds(1f);
}
```

Рисунок 3.14 - Фрагмент коду, що описує тіло корутини Upgrade

На рисунку 3.14 зображено процес апгрейду об'єкту. А саме програвання анімації старого елементу, ввімкнення нового елементу, слідом за цим виконується анімація для нового об'єкту, після чого старий об'єкт вимикається. Усі апгрейди в сцені відбуваються за наступним принципом: старий об'єкт відіграє свою анімацію, слідом з'являється новий об'єкт, після чого старий об'єкт вимикається. Або ж може буди другий випадок коли старий об'єкт відсутній, в такому випадку одразу вмикається та з'являється новий об'єкт.

3.5 Створення елементів UI

Для того щоб користувач мав можливість у будю-який момент, знаходячисть у сцені, зробити апгрейд необхідно розмістити копку, при натисканні якої, буде відбуватися апгрейд

На Рисунку 3.13 зображено кнопку апгрейду, розміщену у сцені.

Рисунок 3.13 – Зображення UI елементів у сцені

Для того, щоб реалізувати апгрейд при натисканні кнопки необхідго.

До Canvas, який відображає UI елементів додати скрипт UserInterfaceView, в якому знаходиться метод OnUpdate(),що зображений на Рисунку 3.14, та додати спраювання цього методу при натисканні на кнопку.,

```
public void OnUpdate()
{
    Context.UpgradeObject();
}
```

Рисунок 3.13 – Фрагмент коду, що описує метод OnUpdate

В даному строці коду відбувається спрацювання методу UpgradeObjectVM, котрий має наступний вигляд, Рисунок 3.15

```
public void UpgradeObject()
{
    if (CanUpgradeObject())
    {
        var updateObject = GetUpgradeObject();
        updateObject.UpgradeObject();
        Context.Location.CurrentUpgradeIndex++;
    }
}
```

Рисунок 3.13 – Фрагмент коду, що описує метод Upgrade Object

В даному випадку у перший строци ми знаходимо індекс об'єкту, який зараз буде апгрейдиться, у другій строци коду визивається спрацювання скрипту UserInterfaceVM, а саме методу UpgradeObject, Рисунок 3.10, в третій індекс наступного на апгрейд об'єкту збільшується на один, тобно натупного разу відбудиться апгрейд вже для наступного у списку об'єкту.

ВИСНОВКИ

Під час виконання дипломної роботи було проаналізовано таке явище як мета гра, починаючи від самого його зародження, до стану на стану на сьогоднішній день. Зроблено інформаційний огляд та аналіз існуючих продуктів-аналогів, що на даний час є лідерами ринку. Розлянуто ряд основних інструментів, котрі використовуються при розробці сучасної метагри.

В результаті аналізу середовищ, які використовуються для розробки мобільних ігр та додатків, було обрано середу розробки Unity. Компонентно-орієнтованого підхід та мультиплатформеність, а також доступ до Unity Asset Store значно пришвидшують розробку та роблять сам процес білш комфортним для розробника.

У результаті виконаної роботи було створено метагру, до існуючої коргри за використанням мови C#. У метагри реалізовано: збереження та завантаження ігрового прогресу, а саме завантаження локації з урахуванням, виконаних користувачем раніше, апгрейдів, можливість вдосконалювати свою локацію, де кожен апгрейд є унікальним.

Створений продукт є дуже важливою частиною гри в цілому, так як надає змогу розробнику взаємодіяти з користувачем напряму. Як результат користувач проводить більше вільного часу за грою, що дозволить і на далі підтримувати продукт та розвивати його. Важливо зазначит, що мета гра сама по собі не має межі у розширенні. Чим більше продукт на ринку тим більше мета елементів додається.

Мета гра, це половина успіху будь-якого проекту, тому вона не втратить актуальності, а й надалі буде тісно вплетена в ігровий процес користувача, збагачуючи його ігровий досвід та покращуючи якість проекту в цілому.

СПИСОК ЛІТЕРАТУРИ

1.Кристоф Вилле Представляем C# Електронний ресурс Обработка исключений, Конфигуиация и создания версии для устанвки 2019 с.89-с.94 https://books.google.com.ua/books?id=byq5DwAAQBAJ&lpg=PA15&dq=C%23&hl=ru&pg=PA15#v=onepage&q=C%23&f=false

2.Бонд Джереми Гибсон Unity и C#. Геймдев от идеи до реализации. 2-е изд. Електронний ресурс Цифровое прототипирование 2019 с. 299 https://books.google.com.ua/books?id=dOweEAAAQBAJ&lpg=PA316&dq=Unity&hl=ru&pg=PA299#v=onepage&q=Unity&f=false

3.Arnaud Weil Learn WPF MVVM - XAML, C# and the MVVM pattern : Be ready for coding away ... Електронний ресурс 6.3 MVVM 2019 https://books.google.com.ua/books?id=52rKDwAAQBAJ&lpg=PP1&dq=MVVM&hl=ru&pg=PT3#v=onepage&q=MVVM&f=false

4.Майкл Моррисон Создание игр для мобильных телефонов Електронний ресурс Создание мобыльной игры Skeleton Специфика создания мобильных игр 2017 с. 56-с.60 http://surl.li/wmtw

5.Майк Гейг Разработка игр на Unity 2018 Електронний ресурс Порядок отрисовки 2021 с.243-с.251 , http://surl.li/wmuj

6.Сергей Галёнкин Маркетинг игр Електронний ресурс Маркетинговий план 2013 с. http://surl.li/wmuv

7.Бонд Джереми Гибсон Unity и C#. Геймдев от идеи до реализации. 2-е изд. Електронний ресурс Прототипы игр и примеры 2019 с.495 https://books.google.com.ua/books?id=dOweEAAAQBAJ&lpg=PA316&dq=Unity&hl=ru&pg=PA299#v=onepage&q=Unity&f=false

8.Эндрю Троелсен, Филипп Джепикс Язык программирования C# 7 и платформы .NET и .NET Core Електронний ресурс Інтерфейси IEnumerable і IEnumerator 2021 с.336-344 http://surl.li/wmvo

9.BEN SMITH Beginning JSON Електронний ресурс String Manipulation с. 27-с. 30 http://surl.li/wokv

10.Хокинг Джозеф Unity в действии. Мультиплатформенная разработка на C#. 2-е межд. Издание Послесловие 2018 с. 336 http://surl.li/wmwy

11.Семенчук В. Мобильное приложение как инструмент бизнеса Електонний посібник Способи монетизации 2017 Глава 5 http://surl.li/woku

12.Федор Г. Пикус Идиомы и паттерны проектирования в современном C++ Електронний ресурс Наследование и иерархии классов 2020 с. 22 http://surl.li/wmwt

13.Paul Deck Spring MVC: A Tutorial (Second Edition) Електронний посібник Model 2 and the MVC Pattern Model 2 and the MVC Pattern 2016 с. 65

14.Rachel Cordone 4 Unreal Engine 4 Game Development Quick Start Guide: Programming professional ... Електронний посібник Optimization Testing and Packaging 2019 с.170-180. http://surl.li/wmye

15.Любовь Пирская Разработка мобильных приложений в среде Android Studio Електронний посібник Интерфейс мобильного приложения(взаимодействие с пользователем) 2019 с.170-с. 174 http://surl.li/wmyb

16.World Intellectual Property Organization Mastering the Game: Business . and Legal Issues for Video Game Developers ... Електронний ресурс Процесс разработки консольной игры 2014 с.30 – с.33 http://surl.li/wmyq

ДОДАТОК А

Скрипт InteriorObjectRepository

```
using Game.Meta.Location.Model;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.IO;
using UnityEngine;

namespace Game.Meta.Location
{
    public class InteriorObjectRepository
    {
        private const string PATH = "Upgrade";

        public Model.Location Location;

        public void Save(Model.Location location)
        {
            Location = location;

            var path = Path.Combine(Application.persistentDataPath, PATH);
            path = Path.Combine(path, "upgrade.json");

            var serialized = JsonConvert.SerializeObject(Location, Formatting.Indented);
            File.WriteAllText(path, serialized);
        }

        public void Load()
        {
            var path = Path.Combine(Application.persistentDataPath, PATH);
            path = Path.Combine(path, "upgrade.json");
            if (File.Exists(path))
            {
                var file = File.ReadAllText(path);
                Location = JsonConvert.DeserializeObject<Model.Location>(file);
            }
        }

        private static DirectoryInfo RequireDirectory()
```

```
        {
            var dataPath = Path.Combine(Application.persistentDataPath, PATH);
            var directory = new DirectoryInfo(dataPath);
            if (!directory.Exists)
            {
                directory.Create();
            }
            return directory;
        }

#if UNITY_EDITOR
        [UnityEditor.MenuItem("Assets/Open location upgrade path")]
        public static void OpenLevelFile()
        {
            var dataPath = Path.Combine(Application.persistentDataPath, PATH);
            Utils.OpenInFileBrowser.Open(dataPath);
        }
#endif
    }
}
```

Скрипт Location

```
namespace Game.Meta.Location.Model
{
    public class Location
    {
        public int CurrentUpgradeIndex;

        public InteriorObject[] InteriorObjects;

        public Location(InteriorObject[] interiorObjects)
        {
            InteriorObjects = interiorObjects;
            CurrentUpgradeIndex = 0;
        }
    }
}
```

Скрипт InteriorObject

```
namespace Game.Meta.Location.Model
{
    public class InteriorObject
    {
        public string Name;
        public int Level;
    }}
```

Скрипт InteriorObjectVM

```
using Game.Meta.Location.Model;
using System;

namespace Game.Meta.Location.ViewModel
{
    public class InteriorObjectVM : ViewModelBase
    {
        private InteriorObject _model;

        public event Action<InteriorObjectVM> Upgrade;
        public event Action<InteriorObjectVM> UpgradeComplete;
        public event Action<InteriorObjectVM> NextToUpgrade;

        public  InteriorObjectVM(InteriorObject  model,  Context  context)  :
base(context)
        {
            _model = model;
        }

        public  InteriorObjectVM(string  name,  int  lvl,  int  profit,  Context
context) : base(context)
        {
            _model = new InteriorObject() { Name = name, Level = lvl};
        }

        public InteriorObject Model
        {
            get
```

```
            ............{
            ................return·_model;
............}
........}

........public·void·UpgradeObject(int·newProfit)
........{
............_model.Level++;···········
............Upgrade?.Invoke(this);
........}

........public·void·UpgradeObjectComplete()
........{
............UpgradeComplete?.Invoke(this);
........}

........public·void·ShowNextUpgradeObject()
........{
............NextToUpgrade?.Invoke(this);
........}
····}
}
```

Скпипт·UserInterfaceVM

```
using·Game.Meta.Location.View;
using·System;
using·System.Collections.Generic;

namespace·Game.Meta.Location.ViewModel
{
····public·class·UserInterfaceVM·:·ViewModelBase
····{
........public·event·Action<UserInterfaceVM>·UpdateScore;
........public·event·Action<UserInterfaceVM>·CantUpgradePopup;
........public·event·Action<InteriorObjectVM>·ObjectUpgrade;

........public·UserInterfaceVM(Context·context)·:·base(context)
........{
............SetNextUpgradeObject();
........}

........private·bool·CanUpgradeObject()
........{
............if·(Context.Location.CurrentUpgradeIndex·>=·Context.UpgradeList.Count)
............{
................return·false;
............}
............var·item·=·Context.UpgradeList[Context.Location.CurrentUpgradeIndex];
............var·currScore·=·Context.PlayerData.Model.Score;

............if·(item.Cost·<=·currScore)
............{
................return·true;
............}

............return·false;
........}

........private·InteriorObjectVM·GetUpgradeObject()
........{
............if·(Context.Location.CurrentUpgradeIndex·>=·Context.UpgradeList.Count)
............{
................return·null;
```

```
            }
            var item = Context.UpgradeList[Context.Location.CurrentUpgradeIndex];
            var path = GetPath(item.ObjectPath);

            var obj = Context.Location.InteriorObjects.Find(x => x.Model.Name == path);

            return obj;
        }

        private static string GetPath(InteriorObjectView view)
        {
            var stack = new Stack<string>();

            stack.Push(view.gameObject.name);

            var parent = view.gameObject.transform.parent;

            while (parent != null)
            {
                stack.Push(parent.name);
                parent = parent.parent;
            }

            return string.Join("/", stack.ToArray());
        }

        public void UpgradeObject()
        {
            if (CanUpgradeObject())
            {
                var item = Context.UpgradeList[Context.Location.CurrentUpgradeIndex];
                Context.PlayerData.Model.Score -= item.Cost;
                UpdateScore?.Invoke(this);

                var updateObject = GetUpgradeObject();
                updateObject.UpgradeObject(item.Profit);
                ObjectUpgrade?.Invoke(updateObject);

                Context.Location.CurrentUpgradeIndex++;

                SetNextUpgradeObject();
            }
            else
            {
                CantUpgradePopup.Invoke(this);
            }
        }

        private void SetNextUpgradeObject()
        {
            var nextToUpgrade = GetUpgradeObject();
            nextToUpgrade?.ShowNextUpgradeObject();
        }
    }
}
```

Скрипт View для InteriorObject

```
using·Game.Meta.Location.ViewModel;
using·System.Collections;
using·UnityEngine;

namespace·Game.Meta.Location.View
{
····public·class·AlarmsGreen·:·InteriorObjectView
····{
········//[SerializeField]·private·SpriteRenderer·_sprite;
········[SerializeField]·private·GameObject[]·_upgrades;

········public·GameObject[]·ObjectInSceneV1,·ObjectInSceneV2;
········public·SpriteRenderer[]·AlfaInSeneV1,·AlfaInSeneV2;

········public·GameObject[]·Ver1,Ver2;
········public·Animation[]·AnimSup1,·AnimSup2;

········public·GameObject·Flag;

········public·AudioSource·UpgradeSound;

········public·Animation·FlagAnim;

········public·Animation[]·PanelAnim,·ClickButtonAnim,·GradientAnim,·AvatarAnim,·
BubbleAnim,·TextAnim;
········public·GameObject[]·Panel,·ClickButton,·Gradient,·Avatar,·Bubble,·Text;

········protected·override·void·OnBindContext(InteriorObjectVM·context)
········{
············base.OnBindContext(context);
············InitView();

············context.Upgrade·+=·Context_Upgrade;
········}

········protected·override·void·OnUnbindContext(InteriorObjectVM·context)
········{
············base.OnUnbindContext(context);

············context.Upgrade·-=·Context_Upgrade;
········}

········protected·override·void·InitView()
········{
············if·(Context.Model.Level·==·0)
············{
················for·(int·i·=·0;·i·<·ObjectInSceneV1.Length;·i++)
················{
····················ObjectInSceneV1[i].SetActive(true);
················}

················for·(int·i·=·0;·i·<·AlfaInSeneV1.Length;·i++)
················{
····················AlfaInSeneV1[i].color·=·new·Color(1,·1,·1,·1);
················}
············}

············if·(Context.Model.Level·==·1)
············{
················for·(int·i·=·0;·i·<·ObjectInSceneV2.Length;·i++)
················{
····················ObjectInSceneV2[i].SetActive(true);
```

```
            }

            for (int i = 0; i < AlfaInSeneV2.Length; i++)
            {
                AlfaInSeneV2[i].color = new Color(1, 1, 1, 1);
            }
        }

    }

    protected override void Context_Upgrade(InteriorObjectVM context)
    {
        StartCoroutine(CoroutineCoin());

        UpgradeSound.Play();

        StartCoroutine(Coroutine());
    }

    IEnumerator Coroutine()
    {
        FlagAnim.Play("flag_off");

        yield return new WaitForSeconds(0.5f);

        Flag.SetActive(false);

        if (Context.Model.Level == 1)
        {
            for (int i = 0; i < Ver1.Length; i++)
            {
                Ver2[i].SetActive(true);
            }

            for (int i = 0; i < AnimSup1.Length; i++)
            {
                AnimSup1[i].Play("alarm_sprite_on");
                AnimSup2[i].Play("alarm_fx_on");
            }

            yield return new WaitForSeconds(1f);

            for (int i = 0; i < Ver1.Length; i++)
            {
                Ver1[i].SetActive(false);
            }
        }

        yield return new WaitForSeconds(1f);

        Context.UpgradeObjectComplete();

        {
            yield return new WaitForSeconds(2f);

            ClickButton[0].SetActive(true);

            Gradient[5].SetActive(true);
            Avatar[5].SetActive(true);
            Bubble[8].SetActive(true);
            Text[8].SetActive(true);

            Panel[0].SetActive(true);
```

```
                PanelAnim[0].Play("panel_on");

                AvatarAnim[5].Play("avatar_right_all_long_on");

                GradientAnim[5].Play("gradient_on");

                BubbleAnim[8].Play("bubble_on");

                TextAnim[8].Play("text_on");

                yield return new WaitForSeconds(3f);

                ClickButtonAnim[0].Play("button_on");

                GameContext.Current.PlayerInfo.Dialogue08 = false;

                GameContext.Current.PlayerInfo.Dialogue07 = true;
            }
        }

        IEnumerator CoroutineCoin()
        {
            yield return new WaitForSeconds(3);
        }
    }
}
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class Arc : InteriorObjectView
    {
        //[SerializeField] private SpriteRenderer _sprite;
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInSceneV1, ObjectInSceneV2;
        public SpriteRenderer[] AlfaInSeneV1, AlfaInSeneV2;


        public GameObject[] Ver1, Ver2;
        public Animation[] AnimV1, AnimV2;
        public Animation[] AnimSup1;
        public GameObject[] gargoyle;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;

        protected override void OnBindContext(InteriorObjectVM context)
        {
```

```
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);

            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
        {
            if (Context.Model.Level == 0)
            {
                for (int i = 0; i < ObjectInSceneV1.Length; i++)
                {
                    ObjectInSceneV1[i].SetActive(true);

                    AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
                }
            }
            else if (Context.Model.Level == 1)
            {
                for (int i = 0; i < ObjectInSceneV2.Length; i++)
                {
                    ObjectInSceneV2[i].SetActive(true);

                    AlfaInSeneV2[i].color = new Color(1, 1, 1, 1);
                }
            }
        }

        protected override void Context_Upgrade(InteriorObjectVM context)
        {
            UpgradeSound.Play();

            StartCoroutine(Coroutine());
        }

        IEnumerator Coroutine()
        {
            FlagAnim.Play("flag_off");

            yield return new WaitForSeconds(0.5f);

            Flag.SetActive(false);

            if (Context.Model.Level == 1)
            {
                for (int i = 0; i < Ver2.Length; i++)// ver_2 object on
                {
                    Ver2[i].SetActive(true);
                }

                for (int i = 0; i < AnimV1.Length; i++)// v1 off
                {
                    AnimV1[i].Play("standard_old");
                }

                for (int i = 0; i < AnimSup1.Length; i++)// items up
                {
```

```
                    if (gargoyle[i].gameObject.activeSelf)
                    {
                        AnimSup1[i].Play("items_up");
                    }
                }

                yield return new WaitForSeconds(0.3f);//pause for v1-v2

                for (int i = 0; i < AnimV2.Length; i++)// v2 on
                {
                    AnimV2[i].Play("standard_new");
                }

                yield return new WaitForSeconds(0.1f);//pause before items down

                for (int i = 0; i < AnimSup1.Length; i++)// items down after upgrade
                {
                    if (gargoyle[i].gameObject.activeSelf)
                    {
                        AnimSup1[i].Play("items_down");

                        yield return new WaitForSeconds(0.05f);// pause after items down
before new item down
                    }
                }

                for (int i = 0; i < Ver1.Length; i++)// ver_1 object off
                {
                    Ver1[i].SetActive(false);
                }
            }

            yield return new WaitForSeconds(1f);

            Context.UpgradeObjectComplete();
        }....
    }
}
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class Box : InteriorObjectView
    {
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInSceneV1;
        public SpriteRenderer[] AlfaInSeneV1;

        public GameObject[] Ver1;
        public Animation[] AnimSup1, AnimSup2;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;
```

```
        protected override void OnBindContext(InteriorObjectVM context)
        {
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);

            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
        {
            if (Context.Model.Level == 1)
            {
                for (int i = 0; i < ObjectInSceneV1.Length; i++)
                {
                    ObjectInSceneV1[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV1.Length; i++)
                {
                    AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
                }
            }
        }

    .....

        protected override void Context_Upgrade(InteriorObjectVM context)
        {
            UpgradeSound.Play();

            StartCoroutine(Coroutine());
        }

        IEnumerator Coroutine()
        {
            FlagAnim.Play("flag_off");

            yield return new WaitForSeconds(0.5f);

            Flag.SetActive(false);

            if (Context.Model.Level == 1)
            {
                for (int i = 0; i < Ver1.Length; i++)
                {
                    Ver1[i].SetActive(true);
                }

                for (int i = 0; i < AnimSup1.Length; i++)
                {
                    AnimSup1[i].Play("basket_spaun");
                    AnimSup2[i].Play("basket_sprite_on");

                    yield return new WaitForSeconds(0.25f);
                }
            }
```

```
            yield return new WaitForSeconds(1f);

            Context.UpgradeObjectComplete();
        }
    }
}
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class NicheBig : InteriorObjectView
    {
        //[SerializeField] private SpriteRenderer _sprite;
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInSceneV1, ObjectInSceneV2;
        public SpriteRenderer[] AlfaInSeneV1, AlfaInSeneV2;

        public GameObject[] Ver1, Ver2;
        public Animation[] AnimV1, AnimV2;
        public Animation[] AnimSup1;

        public SpriteRenderer[] Cheese1;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;

        protected override void OnBindContext(InteriorObjectVM context)
        {
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);

            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
```

```
        {
            if (Context.Model.Level == 0)
            {
                for (int i = 0; i < ObjectInSceneV1.Length; i++)
                {
                    ObjectInSceneV1[i].SetActive(true);

                    AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
                }
            }
            else if (Context.Model.Level == 1)
            {
                for (int i = 0; i < ObjectInSceneV2.Length; i++)
                {
                    ObjectInSceneV2[i].SetActive(true);

                    AlfaInSeneV2[i].color = new Color(1, 1, 1, 1);
                }
            }
        }

        protected override void Context_Upgrade(InteriorObjectVM context)
        {
            UpgradeSound.Play();

            StartCoroutine(Coroutine());
        }

        IEnumerator Coroutine()
        {
            FlagAnim.Play("flag_off");

            yield return new WaitForSeconds(0.5f);

            Flag.SetActive(false);

            if (Context.Model.Level == 1)
            {
                for (int i = 0; i < Ver2.Length; i++)// ver 2 object on
                {
                    Ver2[i].SetActive(true);
                }

                for (int i = 0; i < AnimSup1.Length; i++)
                {
                    if (Cheese1[i].sprite != null)
                    {
                        AnimSup1[i].Play("SE_first");
                    }
                }//cheese frist

                yield return new WaitForSeconds(0.2f);

                for (int i = 0; i < AnimV1.Length; i++)
                {
                    AnimV1[i].Play("niche_old");
                    AnimV2[i].Play("niche_new");
                }//v1-v2

                yield return new WaitForSeconds(1f);

                for (int i = 0; i < AnimSup1.Length; i++)
                {
                    if (Cheese1[i].sprite != null)
```

```
                    {
                        AnimSup1[i].Play("SE_second");

                        yield return new WaitForSeconds(0.065f);
                    }
                }

                for (int i = 0; i < Ver1.Length; i++)
                {
                    Ver1[i].SetActive(false);
                }
            }

            yield return new WaitForSeconds(1f);

            Context.UpgradeObjectComplete();
        }
    }
        }
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class Roze : InteriorObjectView
    {
        //[SerializeField] private SpriteRenderer _sprite;
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInSceneV1, ObjectInSceneV2, ObjectInSceneV3;
        public SpriteRenderer[] AlfaInSeneV1, AlfaInSeneV2, AlfaInSeneV3;

        public GameObject[] Ver1, Ver2, Ver3;
        public Animation[] AnimV1, AnimV2, AnimV3;
        public Animation[] AnimSup1, AnimSup2;
        public GameObject Statue;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;

        protected override void OnBindContext(InteriorObjectVM context)
        {
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);
```

```
            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
        {
            if (Context.Model.Level == 0)
            {
                for (int i = 0; i < ObjectInSceneV1.Length; i++)
                {
                    ObjectInSceneV1[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV1.Length; i++)
                {
                    AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
                }
            }
            else if (Context.Model.Level == 1)
            {
                for (int i = 0; i < ObjectInSceneV2.Length; i++)
                {
                    ObjectInSceneV2[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV2.Length; i++)
                {
                    AlfaInSeneV2[i].color = new Color(1, 1, 1, 1);
                }
            }
            else if (Context.Model.Level == 2)
            {
                for (int i = 0; i < ObjectInSceneV3.Length; i++)
                {
                    ObjectInSceneV3[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV3.Length; i++)
                {
                    AlfaInSeneV3[i].color = new Color(1, 1, 1, 1);
                }
            }

        }......

        protected override void Context_Upgrade(InteriorObjectVM context)
        {
            UpgradeSound.Play();

            StartCoroutine(Coroutine());
        }

        IEnumerator Coroutine()
        {
            FlagAnim.Play("flag_off");

            yield return new WaitForSeconds(0.5f);

            Flag.SetActive(false);

            yield return new WaitForSeconds(0.1f);
            {
                if (Context.Model.Level == 1)
                {
                    for (int i = 0; i < Ver2.Length; i++)
```

```
                {
                    Ver2[i].SetActive(true);
                }

                for (int i = 0; i < AnimV1.Length; i++)
                {
                    AnimV1[i].Play("standard_old");
                    AnimV2[i].Play("standard_new");
                }

                if (Statue.gameObject.activeSelf)
                {
                    AnimSup2[0].Play("statue_fx_oldnew");

                    AnimSup1[0].Play("items_up");

                    yield return new WaitForSeconds(0.33f);

                    AnimSup1[0].Play("items_down");
                }

                yield return new WaitForSeconds(1f);

                for (int i = 0; i < Ver1.Length; i++)
                {
                    Ver1[i].SetActive(false);
                }
            }
            else if (Context.Model.Level == 2)
            {
                for (int i = 0; i < Ver3.Length; i++)
                {
                    Ver3[i].SetActive(true);
                }

                for (int i = 0; i < AnimV2.Length; i++)
                {
                    AnimV2[i].Play("standard_old");
                    AnimV3[i].Play("standard_new");
                }

                if (Statue.gameObject.activeSelf)
                {
                    AnimSup2[0].Play("statue_fx_oldnew");

                    AnimSup1[0].Play("items_up");

                    yield return new WaitForSeconds(0.33f);

                    AnimSup1[0].Play("items_down");
                }


                yield return new WaitForSeconds(1f);

                for (int i = 0; i < Ver2.Length; i++)
                {
                    Ver2[i].SetActive(false);
                }
            }

            yield return new WaitForSeconds(1f);

            Context.UpgradeObjectComplete();
```

```
            for (int i = 0; i < MaskV1.Length; i++)
            }
        }
    }
}
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class Stande1 : InteriorObjectView
    {
        //[SerializeField] private SpriteRenderer _sprite;
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInScene;
        public SpriteRenderer[] AlfaInSene;
        public GameObject[] MaskV1, MaskV2, MaskV3;

        public GameObject[] Ver1, Ver2, Ver3;
        public Animation[] AnimV1, AnimV2, AnimV3;
        public Animation[] AnimSup1, AnimSup2, AnimSup3, AnimSup4, AnimSup5;
        public GameObject[] Cundle;
        public SpriteRenderer[] Cheese1, Cheese2;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;

        protected override void OnBindContext(InteriorObjectVM context)
        {
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);

            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
        {
            if (Context.Model.Level == 0)
            {
                ObjectInScene[0].SetActive(true);

                AlfaInSene[0].color = new Color(1, 1, 1, 1);
```

```
                }
            }//выезд сыров

            yield return new WaitForSeconds(0.2f);

            for (int i = 0; i < MaskV2.Length; i++)
            {
                MaskV2[i].SetActive(false);
            }// отключения масок в2

            for (int i = 0; i < AnimV2.Length; i++)
            {
                AnimV2[i].Play("standard_old");
            }//уход в2 ....

            for (int i = 0; i < AnimSup5.Length; i++)
            {
                if (AnimSup5[i] != null && Cundle[i].gameObject.activeSelf)
                    AnimSup5[i].Play("items_up");
            }//взлет свечек

            yield return new WaitForSeconds(0.2f);

            for (int i = 0; i < AnimV3.Length; i++)
            {
                AnimV3[i].Play("standard_new");
            }//появление в3

            for (int i = 0; i < MaskV3.Length; i++)
            {
                MaskV3[i].SetActive(true);
            }//включение масок в3

            yield return new WaitForSeconds(0.5f);

            for (int i = 0; i < AnimSup1.Length; i++)
            {
                if (i < AnimSup1.Length && AnimSup1[i] != null && Cheese1[i].sprite
!= null)
                {
                    AnimSup1[i].Play("SE_second");
                }

                if (i < AnimSup2.Length && AnimSup2[i] != null && Cheese2[i].sprite
!= null)
                {
                    AnimSup2[i].Play("SW_second");
                }

                if (i < AnimSup3.Length && AnimSup3[i] != null &&
Cundle[i].gameObject.activeSelf)
                {
                    AnimSup3[i].Play("items_down");
                }

                if (i < AnimSup4.Length && AnimSup4[i] != null &&
Cundle[i].gameObject.activeSelf)
                {
                    AnimSup4[i].Play("items_down");
                }

                yield return new WaitForSeconds(0.025f);

            }//заезд сыров и свечек
```

```
              for (int i = 0; i < Ver2.Length; i++)
              {
                  Ver2[i].SetActive(false);
              }//выключаем v2
          }
          else if (Context.Model.Level == 1)
          {
              for (int i = 0; i < Ver2.Length; i++)
              {
                  Ver2[i].SetActive(true);
              }//включеми v2

              for (int i = 0; i < AnimSup1.Length; i++)//выезд сыров
              {
                  if (AnimSup1[i] != null && Cheese1[i].sprite != null)
                  {
                      AnimSup1[i].Play("SE_first");
                  }
              }//выезд сыров

              for (int i = 0; i < AnimSup2.Length; i++)
              {
                  if (AnimSup2[i] != null && Cheese2[i].sprite != null)
                  {
                      AnimSup2[i].Play("SW_first");
                  }
              }//выезд сыров

              yield return new WaitForSeconds(0.2f);

              for (int i = 0; i < MaskV1.Length; i++)
              {
                  MaskV1[i].SetActive(false);
              }// отключения масок в1

              for (int i = 0; i < AnimV1.Length; i++)
              {
                  AnimV1[i].Play("standard_old");
              }//уход в1....

              for (int i = 0; i < AnimSup5.Length; i++)
              {
                  if (AnimSup5[i] != null && Cundle[i].gameObject.activeSelf)
                      AnimSup5[i].Play("items_up");
              }//взлет свечек

              yield return new WaitForSeconds(0.2f);

              for (int i = 0; i < AnimV2.Length; i++)
              {
                  AnimV2[i].Play("standard_new");
              }//появление в2

              for (int i = 0; i < MaskV2.Length; i++)
              {
                  MaskV2[i].SetActive(true);
              }//включение масок в2

              yield return new WaitForSeconds(0.5f);

              for (int i = 0; i < AnimSup1.Length; i++)
              {
```

```
                    if (i < AnimSup1.Length && AnimSup1[i] != null && Cheese1[i].sprite
!= null)
                    {
                        AnimSup1[i].Play("SE_second");
                    }

                    if (i < AnimSup2.Length && AnimSup2[i] != null && Cheese2[i].sprite
!= null)
                    {
                        AnimSup2[i].Play("SW_second");
                    }

                    if (i < AnimSup3.Length && AnimSup3[i] != null &&
Cundle[i].gameObject.activeSelf)
                    {
                        AnimSup3[i].Play("items_down");
                    }

                    if (i < AnimSup4.Length && AnimSup4[i] != null &&
Cundle[i].gameObject.activeSelf)
                    {
                        AnimSup4[i].Play("items_down");
                    }

                    yield return new WaitForSeconds(0.025f);

                }//заезд сыров и свечек

                for (int i = 0; i < Ver1.Length; i++)
                {
                    Ver1[i].SetActive(false);
                }//выключаем v1
            }

            yield return new WaitForSeconds(1f);

            Context.UpgradeObjectComplete();
        }
    }
}
```

Скрипт View для InteriorObject

```
using Game.Meta.Location.ViewModel;
using System.Collections;
using UnityEngine;

namespace Game.Meta.Location.View
{
    public class Statue : InteriorObjectView
    {
        //[SerializeField] private SpriteRenderer _sprite;
        [SerializeField] private GameObject[] _upgrades;

        public GameObject[] ObjectInSceneV1, ObjectInSceneV2;
        public SpriteRenderer[] AlfaInSeneV1, AlfaInSeneV2;

        public GameObject[] Ver1, Ver2;
```

```
        public Animation[] AnimV1, AnimV2;
        public Animation[] AnimSup1;

        public GameObject Flag;

        public AudioSource UpgradeSound;

        public Animation FlagAnim;

        protected override void OnBindContext(InteriorObjectVM context)
        {
            base.OnBindContext(context);
            InitView();

            context.Upgrade += Context_Upgrade;
        }

        protected override void OnUnbindContext(InteriorObjectVM context)
        {
            base.OnUnbindContext(context);

            context.Upgrade -= Context_Upgrade;
        }

        protected override void InitView()
        {
            if (Context.Model.Level == 1)
            {
                for (int i = 0; i < ObjectInSceneV1.Length; i++)
                {
                    ObjectInSceneV1[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV1.Length; i++)
                {
                    AlfaInSeneV1[i].color = new Color(1, 1, 1, 1);
                }
            }
            else if (Context.Model.Level == 2)
            {
                for (int i = 0; i < ObjectInSceneV2.Length; i++)
                {
                    ObjectInSceneV2[i].SetActive(true);
                }

                for (int i = 0; i < AlfaInSeneV2.Length; i++)
                {
                    AlfaInSeneV2[i].color = new Color(1, 1, 1, 1);
                }
            }
        }

        protected override void Context_Upgrade(InteriorObjectVM context)
        {
            UpgradeSound.Play();

            StartCoroutine(Coroutine());
        }

        IEnumerator Coroutine()
        {
            FlagAnim.Play("flag_off");

            yield return new WaitForSeconds(0.5f);
```

```
            {
                if (Context.Model.Level == 2)
                {
                    for (int i = 0; i < Ver2.Length; i++)
                    {
                        Ver2[i].SetActive(true);
                    }

                    AnimSup1[0].Play("statue_fx_oldnew");
                    AnimV1[0].Play("standard_old");

                    yield return new WaitForSeconds(0.33f);

                    AnimV2[0].Play("standard_new");

                    Ver1[0].SetActive(false);

                }
                else if (Context.Model.Level == 1)
                {
                    for (int i = 0; i < Ver1.Length; i++)
                    {
                        Ver1[i].SetActive(true);
                    }

                    for (int i = 0; i < AnimV1.Length; i++)
                    {
                        AnimV1[i].Play("standard_new");
                        AnimSup1[i].Play("statue_fx_new");
                    }
                }

                yield return new WaitForSeconds(2f);

                Context.UpgradeObjectComplete();
            }
        }......
    }
}
```