Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Поповського Всеволода Станіславовича*
(ПІБ)

академічної групи *121-19з-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Створення комп'ютерної гри Voidborn жанру First-Person RPG Dungeon Crawler з використанням рушія Godot*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І. А.*			
Нормоконтролер	*ст.викл. Мартиненко А.А*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19з-1* *Поповського Всеволода Станіславовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Створення комп'ютерної гри Voidborn жанру First-Person RPG dungeon crawler з використанням рушія Godot*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *Доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Поповський В.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 63 с. 12 рис., 3 дод., 5 джерел.
Об’єкт розробки: Комп’ютерна гра
Мета кваліфікаційної роботи: Створити Full-Featured MVP-версію гри Voidborn, жанру First-Person RPG Dungeon Crawler.

У вступі розглядається історія жанру та його визначення.

У першому розділі проводиться аналіз історичного та поточного стану жанра, і ґрунтуючись на цьому аналізу визначається конкретна постановка завдання.

У другому розділі виконано проектування і розробку гри, наведено структуру системи, описано застосовані програмні застосунки, наведено опис роботи гри.

В економічному розділі розраховується трудомісткість та вартість роботи з розробки програми.

Список Ключових Слів: UI, OS, RPG, JRPG, TTRPG, D&D, MVP, AI, PC

ABSTRACT

Explanatory note: 63 pages, 12 pics, 3 apps, 5 sources.
Object of development: Computer game
The purpose of the diploma project: To create a Full-Featured MVP version of the game Voidborn, a First-Person RPG Dungeon Crawler.

The introduction discusses the history of the genre and its definition.

The first chapter analyzes the historical and current state of the genre, and based on this analysis, a specific task statement is defined.

The second section describes the design and development of the game, presents the system structure, describes the software used, and describes the game's operation.

The economic section calculates the labor intensity and cost of program development.

Keyword list: UI, OS, RPG, JRPG, TTRPG, D&D, MVP, AI, PC

ЗМІСТ

4

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

UI - User Interface - Інтерфейс Користувача

OS - Operating System - Операційна Система

RPG - Role Playing Game - Рольова гра

JRPG - Japanese Role Playing Game - Японська рольова гра

TTRPG - TableTop Role Playing Game - Настільна рольова гра

D&D - Dungeons And Dragons

MVP- Minimum Viable Product

AI - Artificial Intelligence - Штучний Інтелект

PC - Personal Computer - Персональний Комп’ютер

ВСТУП

Історія комп’ютерних RPG як жанру починається на столі у 1970-их, з створенною Gary Gygax-ом Chainmail[1]. Chainmail була настільною тактичною військовою грою - кожен ігрок контролював набір мініатюр які репрезентували різних “юнітів”, яких треба було використовувати для того щоб перемагати над юнитами інших ігроків та виконувати цілі сценаріїв (контролювання території, перемога над ключовими ворожими юнітами, рухування своїх юнітів до ключових позицій, ін.).

Одне з розширень для Chainmail включало до себе так званих “Героїв” - унікальних юнітів по типу легендарних воїнів, магів та усіляких розумних магічних створінь. Люди будучи людьми в процесі грі почали вигадувати історії для своїх героїчних юнітів - “відігрувати” їх - перетворюючи їх із фігур з набором характеристик до повноцінних персонажів, які існували в конструйованому світі та борються з іншими героями не просто заради того щоб виграти, а заради досягнення своїх лічних цілей, слідуючи своїм історіям.

Помітивши це Gygax з часом створив Dungeons & Dragons [2] - першу в світі TTRPG. Вона була версією Chainmail в якій кожен ігрок контролював одного

5

героя, якого тепер можно було створювати цілком самому. Один з ігроків не контролював героя, а брав на себе роль так званого “Dungeon Master”-а, який контролював усе окрім персонажів ігроків - створював для них пригоду та відігравав усіх опонентів на шляху героїв. D&D Вибухнула, та досягла такої популярності що навіть зараз, у 2023 році, п’яте видання D&D є найбільш популярною TTRPG-системою у світі.

6

З часом та розвитком комп’ютерних систем була безліч спроб адаптувати D&D - або хоча б частину її концептів, в формат комп'ютерної гри. Адаптувати правила передвижения на мапі, бою, розвитку персонажа - це з часом все реалізували, але з’явилася проблема - ні тоді, ні навіть зараз, немає систем які могли б вести себе як Dungeon Master. Комп’ютерні системи не можуть по ходу гри розуміти абсолютно все що хоче ігрок, та вони звісно не можуть адаптуватися під ці побажання - нереально створювати новий діалог, локації, персонажів, механіки, та сюжетні лінії у комп’ютерних системах на ходу з достатнім рівнем якості.

Тому комп'ютерні ігри які спробували адаптувати D&D - тепер звані RPG - пішли своїм шляхом. Вони сфокувалися на геймплеі - бойових системах, менеджменті ресурсів - та мали лише рудиментарні, статичні сюжетні лінії які існували в першу чергу для того щоб контекстуалізувати геймплей.

Тут і можно знайти розкол між RPG та JRPG - у той час як западні RPG фокусувалися на геймплеі та відидвигали сюжет на другий план, японські “JRPG” пішли у інше направлення - ці ігри в першу чергу були про сюжет - хай він і був статичним як у книзі або фільмі, та геймплей в них існував заради покращення сюжету. Це з часом дійшло до того що японцями був створений жанр Візуальних Новел, які являються іграми на папері але практично просто є книгами з картинками та звуком.

У наш час, в 2023 році, ця різниця вже втратила свій сенс. Є безліч RPG з японії які в першу чергу фокусуються на геймплеі, та не менше западних RPG з довгими та складними сюжетами.

6

Ігри, своєю сутністю, є формою мистецтва, та багато хто може сказати що жанрові рамки зовсім не мають значення - особливо в наш час коли вже не раз з великим успіхом комбінували елементи зовсім різних жанрів - та в якомусь сенсі вони будуть праві, але я досить вважаю що жанрові маркери важливі для тих хто хоче, наприклад, знайти інші ігри схожі на ті що їм сподобалися в минулому, або як максимально скорочені описи.

First-Person Dungeon Crawler RPG є досить вузьким піджанром RPG. З названня можно зрозуміти що ці ігри мають перспективу від першого лиця, та те що основний ігровий цикл заключається в експлорації “данжів” - великих (часто підземних) структур. Цей піджанр взагалі-то є одним із перших піджанрів RPG, з Ultima Underworld 1992 року (рис.1) як один із перших популярних прикладів.

Рис.1. Геймплей Ultima Underworld: The Stygian Abyss

Цей жанр не бачe багато релізів у наш час, вважаючись пережитком минулого - але я так не вважаю. Це може бути особиста думка, але багато моїх любимих комп’ютерних ігор належать саме до цього жанру, наприклад Labyrinth Of Refrain: Coven Of Dusk (рис.2), Lord Of The Seal (рис.3) та недавно випущена Labyrinth Of Galleria: The Moon Society (рис.4).

7

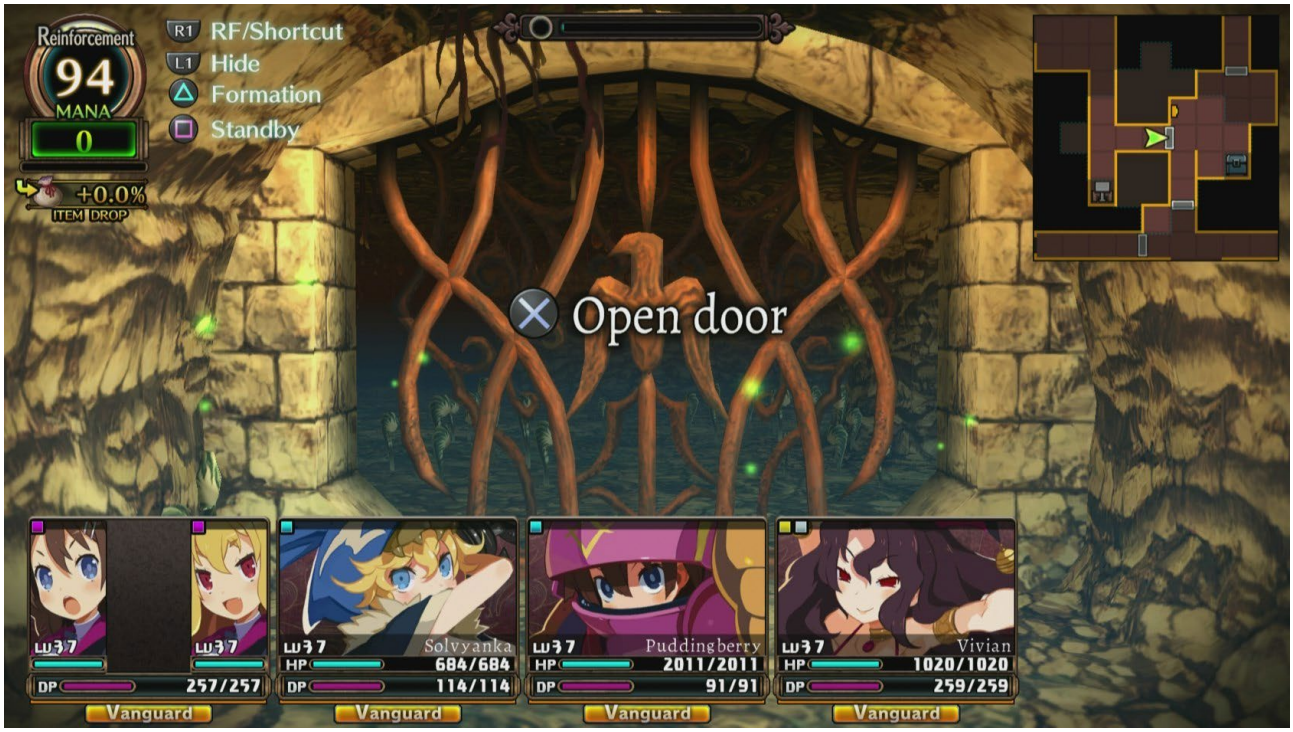

Рис.2. Геймплей Labyrinth Of Refrain: Coven Of Dusk

Рис.3. Геймплей Lord Of The Seal

8

Рис.4. Геймплей Labyrinth Of Galleria: The Moon Society

Саме любов до цього жанру і є основною причиною обрання мети кваліфікаційної роботи: Створення комп'ютерної гри Voidborn жанру First-Person RPG Dungeon Crawler з використанням рушія Godot.

9

РОЗДІЛ 1 - АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 - Загальні відомості з предметної галузі

1.1.1 - Аналіз існуючих аналогів

Однією з найпопулярніших серій ігор у жанрі є Etrian Odyssey. Перша гра в цій серії вийшла на оригінальній Nintendo DS у 2007 році та захватила дуже велику публику своїм геймплеєм та фактом того що вона була на переносній платформі[3].

Ігри у серії концептуально відрізняються слабко - у кожній гравцем збирається команда з 6 персонажів якими потім потрібно досліджувати великий лінійний лабіринт. Ці ігри не занадто комплексні - завдання гравця на кожному рівні полягає або в тому щоб знайти тунель на наступний рівень, або в тому щоб перемогти над потужним противником “боссом” щоб цей тунель відкрити. Декілька ігор у серії експериментували з побічними квестами та сюжетними лініями, але основний геймплейний цикл завжди однаковий.

Рис. 5 Геймплей Etrian Odyssey HD

Головний вибір гравця в грі, таким чином, полягає в виборі команди 6 персонажів з безлічі класів та подальшої спеціалізації цих персонажів з їх прогресією, яка в більшості ігор реалізована в форматі дерев навичок.

10

Бойова система Etrian Odyssey довольно “ванільна” - вона мало відходить від широко поширених принципів які популяризували Final Fantasy та інші старі пошагові RPG. В якомусь сенсі це добре - гравці які вже знають як ці системи працюють зможуть дуже швидко зрозуміти як грати, але з іншої сторони вони не знайдуть в бойовій системі нічого нового.

Інша серія, яка вже згадувалася у вступі, це “Coven And Labyrinth” - вона включає в себе Labyrinth Of Refrain та Labyrinth Of Galleria.

Ці ігри досить унікальні в жанрі, так як мають великий сюжетний фокус - десь 30-40% геймплею це сюжетні сцени з мінімальним геймплеем. Реалізовано це за рахунок того що персонаж гравця не є повноцінним “персонажем” (в першій грі магічна книга, в другій лампа). Цей “аватар” гравця не може говорити та практично виконує чисто геймпейну функцію, у той час як сюжет розкривається персонажами які хочуть аватар гравця використовувати для своїх цілей, які майже завжди зводяться до “піди у лабіринт та зроби там щось”.

Геймпейно ігри у серії досить сильно схожі на Etrian Odyssey. Експлорація данжів все так же йде на квадратній сітці з боями час від часу. Варіація задач гравця тут значно більше ніж в Etrian Odyssey - часто потрібно повертатися на минулі рівні щоб знайти там щось нове, та є декілька місць по ходу гри де можно робити кілька речей в будь-якому порядку. Дизайн данжів, особливо в Galleria, также значно кращий - вони ведуть себе не просто як лабіринти а як повноцінні пазли.

Бойові системи цих ігор нажаль не дуже добрі. Вони мають цікаві ідеї (отримання способностей інших класів через “реінкарнацію” в них, наприклад), але вони всі тонуть в факту того що більшість боїв виграє та сторона яка перша зможе зробити критичну атаку - в цих іграх вони часто можуть вбити персонажа за 1 раз без можливості цього персонажа потім “підняти”.

1.1.2 - Пошук точок інновації

Для того щоб створити успішну гру недостатньо просто створити добру гру - вона не буде існувати в ізоляції. Потрібно шукати місця де можно створити щось своє. Для багатьох ігор це сюжет - навіть якщо геймплей “заїзджений” - мало відрізняється від інший ігор - гравці все одно будуть грати щоб побачити як розвинеться історія. Але є проблема - жанр First-Person Dungeon Crawler-ів історично не дуже добре грається з великим сюжетним фокусом. Нарративно складно розміщувати сюжет в сеттингах, які потрібні для геймплейного функціоналу ігор у жанрі.

Тому треба йти у іншому направленні - інноватуватись у геймпейних системах. У конкретно цьому проекті самою головною точкою інновації є заміна класичноі системи “спорядження” (броня, зброя, аксесуари) на систему “gemcraft”-інгу. Замість статичних предметів спорядження гравець буде отримувати gem-и, які гравець буде повинен вставляти у сітку кожного юніту для того щоб покращувати їх характеристики та давати ім нові пасивні та активні вміння.

Детальний опис цієї системи можно знайти в **2.5.1.2 - Створення системи “gemcraft”**

1.2 - Призначення розробки та галузь застосування

Темою бакалаврської кваліфікаційної роботи є “Створення комп'ютерної гри Voidborn жанру First-Person RPG Dungeon Crawler з використанням рушія Godot”. В якості мови програмування був обран GDScript - встроєна у Godot, обраний рушій, мова програмування - дуже схожа на Python. Головною метою роботи є робочій MVP - Minimum Viable Product - фактично демо-версія яка реалізує всі основні механіки для демонстрації того чим буде повна гра (розробка

повної гри не вписується в кваліфікаційну роботу, так як займе багато років, особливо соло).

13

Гра буде створена для OS Windows та буде підтримувати контроль з клавіатури.

1.3 - Підстава для розробки

Підставами для розробки є:

- Освітня програма за спеціальністю 121 “Інженерія програмного забезпечення”.
- Графік навчального процесу та навчальний план.
- Наказ ректора Національного технічного університету «Дніпровська політехніка» № 256-с від 11.04.2023 р.
- Завдання на кваліфікаційну роботу на тему “Створення комп'ютерної гри Voidborn жанру First-Person RPG Dungeon Crawler з використанням рушія Godot”.

1.4 - Постановка завдання

Метою роботи є розробка MVP-версії гри. Це повинно включати в себе:

- Механики руху (експлорації) 3D ігрового світу
- Частину ігрового світу
- Пошагову бойову систему
- Систему прогресії та кастомізації персонажів гравця (gemcraft)

1.5 - Вимоги до програми або програмного виробу

1.5.1 - Вимоги до функціональних характеристик

Для досягу функціональних цілей гра повинна:

- Запускатися та стабільно працювати на системах Windows, виключаючи зовсім слабкі та старі системи.
- Не мати ніяких багів, які можуть заважати прогресії

13

- Зручно встановлюватися

1.5.2 - Вимоги до інформаційної безпеки

Для надійної роботи гри потрібно:

- Використовувати лише ліцензійне програмне забезпечення

1.5.3 - Вимоги до складу та параметрів технічних засобів

Вимоги до складу тех. Засобів наступні:

- Монітор з роздільною здатністю хоча б 1600x800px.
- Клавіатура
- ОС: Windows 7/8/10/11 (64-bit)
- CPU: Core i3 3210 або краще
- 4 GB RAM або більше
- Відеокарта: NVIDIA GT 440 1GB або краще
- Пам’ять на диску: 1 GB

1.5.4 - Вимоги до інформаційної та програмної сумісності

Кваліфікаційна робота повинна бути сумісною з ОС Windows.

14

РОЗДІЛ 2 - ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 - Функціональне призначення програми

Після аналізу інших ігор у жанрі та оцінки своїх можливостей з урахуванням відведеного часу, було вирішено створити MVP-версію гри “Voidborn” з використанням рушія Godot. Godot був обраний так як я з ним вже мав досвід. На Godot можно використовувати C# або GDScript для програмування, для цього проекту було обрано GDScript за його простоту та дуже добру інтеграцію в рушій.

Ця версія гри повністю реалізує інтро та перший рівень гри, маючи приблизно 1~1.5 годин геймплею.

Гра включає в себе наступні механіки:

1. Бойова система
    a. AI який обробляє ходи противників
    b. Меню для вибору дії на ходах юнитів гравця
    c. Обробка дій та статусів у ході боя
        i. Вирахування урону атак включаючи хар-ки юнитов які атакують та являються атакованими
        ii. Вирахування взаємодій між різними ефектами, такими як:
            1. MP, який використовується для декількох більш сильних способностей
            2. Бар’єр, який являється другим слоєм здоров’я з іншими правилами обробки урону
            3. Лікування здоров’я
            4. Статус криту
2. Кастомізація персонажів ігрока
    a. Можливість заміни спрайтів персонажів на будь-яку картинку на дисці
    b. Геймплейна кастомізація персонажів через систему “Gemcraft”
        i. Вставка gem-ів на сітки кожного індивідуального персонажа

15

ii. Відкриття нових місць для gem-ів, використовуючи “notch”-і
iii. Посилювання gem-ів, ядра яких торкаються на сітці

3. Експлорація ігрового світу
    a. Рух по ігровому світу
    b. Взаємодія з об’єктами у світі
        i. Сундуки які дають предмети (gem-и, notch-и та ключові предмети)
        ii. Проламування стін з тріщинами після отримання ключового предмету, який дозволяє це робити
        iii. Ініціація бою з противниками, що знаходяться у світі
        iv. Читання тексту на об'єктах у світі
        v. Переходи між рівнями
        vi. Взаємодія з іншими об'єктами (ричаги)

2.2 - Опис застосованих математичних методів

Для обрахування “процентних” характеристик юнітів (характеристики які повинні транслюватися з диапазону $0\sim\infty$ у діапазон 0~100, наприклад захист) була використана наступна формула:

$$r = \left(1 - \frac{1}{\frac{s_v^{1.2} + p_i}{p_i}}\right) \times 100$$

Де p_i - Point Influence (ричаг для балансування), s_v - Stat Value (Значення характеристики $0\sim\infty$), r - результат у діапазоні 0~100.

16

2.3 - Опис використаної архітектури та шаблонів проектування

Код-база проекту побудована на принципах Об’єктно-орієнтованого програмування з функціонально-орієнтованими частинами. З часто використаних патернів програмування виділяється Singleton, який використовується наприклад для того щоб зберігати інформацію про гравця (інвентар, персонажі гравця).

2.4 - Опис використаних технологій та мов програмування

2.4.1 - Рушій Godot

Godot є дуже легковажним рушієм на якому можно розлобляти будь-який жанр ігор для більшості платформ (офіційно підтримуються Android, IOS, Linux, macOS, UWP для Windows, Web та Windows Desktop (.exe-експорт), неофіційно є темплейти експорту на декілька ігрових консолей).

Godot не потребує встановлення та зістовляє з себе лише 1 .exe файл вагою 110мб. Цей файл це все що потрібно для розробки на рушію.

Будучи універсальним рушієм в дусі Unreal або Unity Godot має можливість розробки як 3D так і 2D ігор через два різні набора ігрових компонентів (які, до речі, не прив’язані до “типу проекту”, як наприклад у Unity - 2D та 3D компоненти створені щоб працювати разом без проблем). Але, будучи універсальним рушієм, Godot не має ніяких readymade компонентів які зазвичай використовуються тільки у декількох жанрах. Немає вбудованих компонентів які симулюють рух машин (як наприклад у Source) чи темплейтів для створення пошагових бойових систем (як наприклад є у RPG Engine або WolfEngine) - Godot дає всі інструменти для їх створення, але не більше.

2.4.2 - Мова GDScript

17

GDScript є встроєнною у Godot мовою програмування. Вона є повноцінною мовою з Python-подібним синтаксисом - але не має проблем з швидкістю роботи за рахунок дуже доброї інтеграції у сам рушій.

2.5 - Опис структури програми та алгоритмів її функціонування

2.5.1 - Етапи розробки

Розробку проекту можно разділити на наступні великі частини:

- Створення ігрового світу
- Створення системи “gemcraft”
- Створення бойової системи

2.5.1.1 - Створення ігрового світу

Рушій Godot має графічний інтерфейс та сам ігровий світ, звісно, робиться саме у цьому графічному інтерфейсу. Для MVP-версії гри було створено 3 рівні - стартова кімната (Рис.6), “хаб” (Рис.7) та сам перший рівень - “острівний бастіон” (Рис.8)

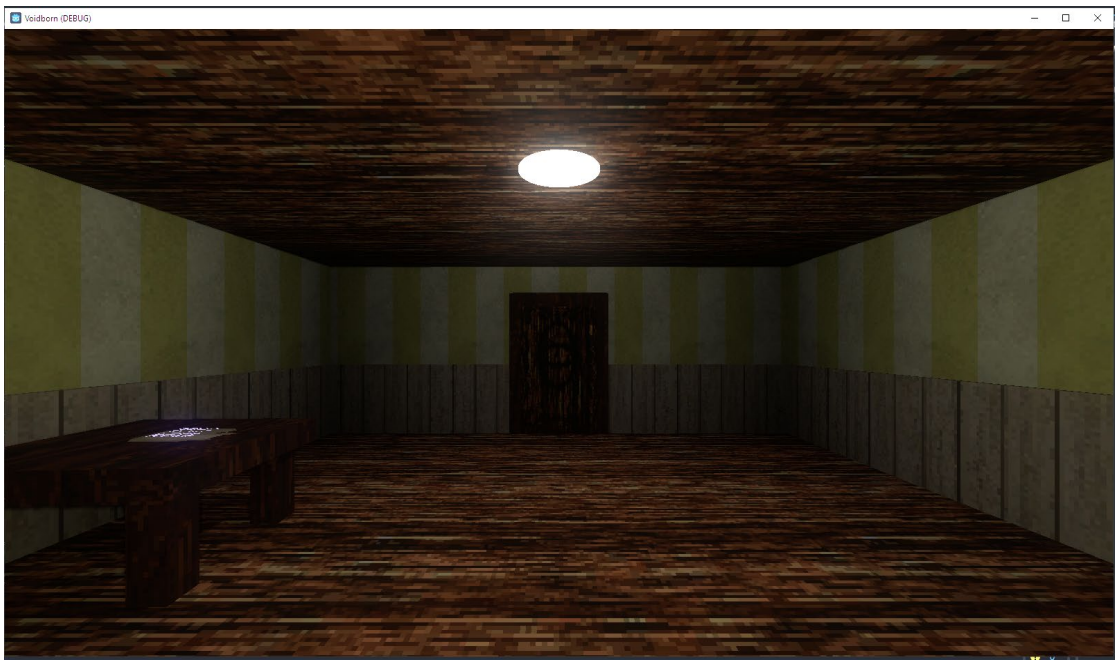

Рис. 6 - Стартова кімната

18

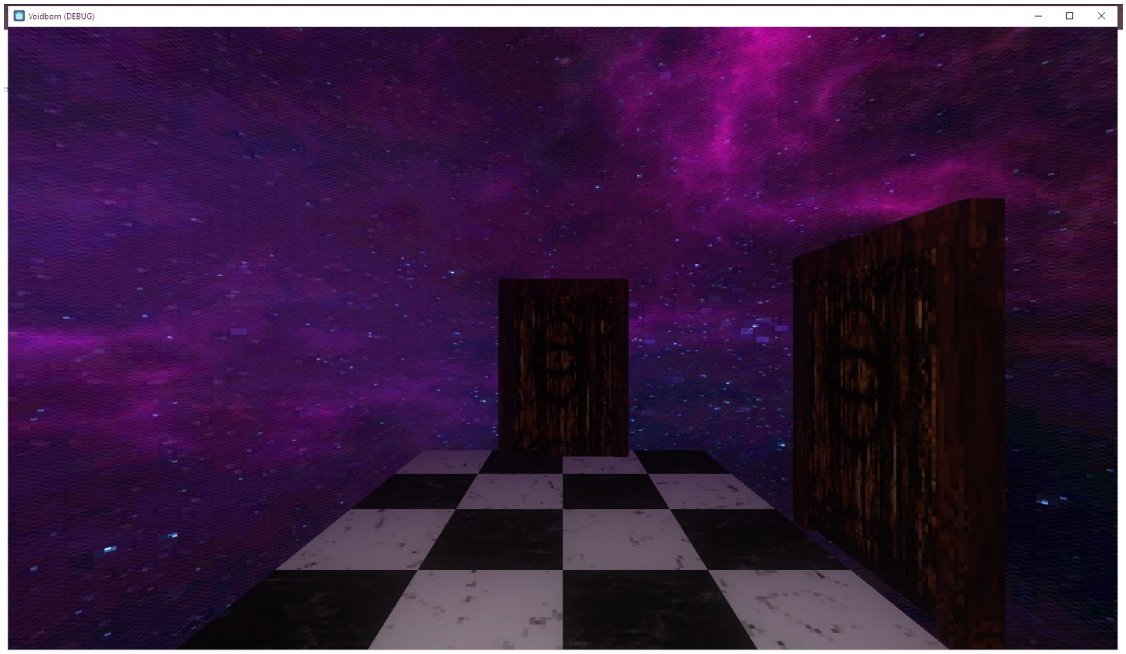

Рис. 7 - Хаб

Рис. 8 - Острівний Бастіон

Рух самого гравця, як класично прийнято у жанрі, проходить на сітці.

2.5.1.2 - Створення системи “gemcraft”

Основним натхненням для цієї системи була система синтезу в Atelier Sophie 2: The Alchemist of the Mysterious Dream (Рис. 9)

19

Рис. 9 - Єкран синтезу в Atelier Sophie 2

В тій грі ця система використовується для створення спорядження - як броні зі зброєю, так і активних предметів (бомби, зілля, ін.). Для створення предмету треба обрати набір компонентів (зліва), кожен з яких має свою форму яку він буде займати у “котлі” (по центру). Завдання гравця - викласти всі компоненти таким чином щоб вони не накладувались один на іншого та їх “сильові точки” (більш яскраві клітки) максимально торкались. Після того як гравець викладає всі компоненти так добре як може вони застосовуються та предмет створюється - тим сильніше чим більше кліток “котла” було заповнено та скільки було торкань “сильових точок”.

В **1.1.2 - Пошук Точок Інновації** вже було сказано що в Voidborn система gemcrafting-у займати місце класичної системи спорядження. Це працює наступним чином:

1. У процесі гри (після боїв та відкриття скриньок) гравець отримує gem-и. Кожен gem має наступні характеристики:
    a. Форма, яку він займає
    b. Одна клітка “ядро”
    c. Звичайний ефект
    d. Посилений ефект

20

2. У гравця є 3 персонажа - кожен з цих персонажів має свою “дошку” на яку можно викладувати gem-и. Всі персонажі мають дошку 3x3 на початку гри, але іі можно розширювати до 5x5 використовуючи “notch”-і які можно знайти у світі, тими же методами що й gem-и.
3. В процесі вставки gem-ів, якщо гравець зможе вставити їх таким чином щоб “ядра” двох gem-ів касались вони стануть “посиленими” та дадуть кращий ефект. Деякі gem-и мають зовсім інший єфект коли вони посилюються, та в залежності від того який ефект гравець хоче іноді треба уникати посилювання.

Рис. 10 - Єкран Gemcraft-інгу. Зліва - список доступних gem-ів, по центру - доска, справа - опис ефекту наведеного gem-у. Клітки с символами - ядра (наразі посилені)

2.5.1.3 - Створення бойової системи

Godot не має темплейту пошагової бойової системи (як наприклад є у RPG Maker або Wolf Engine), тому всю архітектуру треба було робити самому. З однієї

21

сторони це дуже великий об’єм роботи, з іншої у мене є повний креативний контроль над тим як кожна частина системи працює.

22

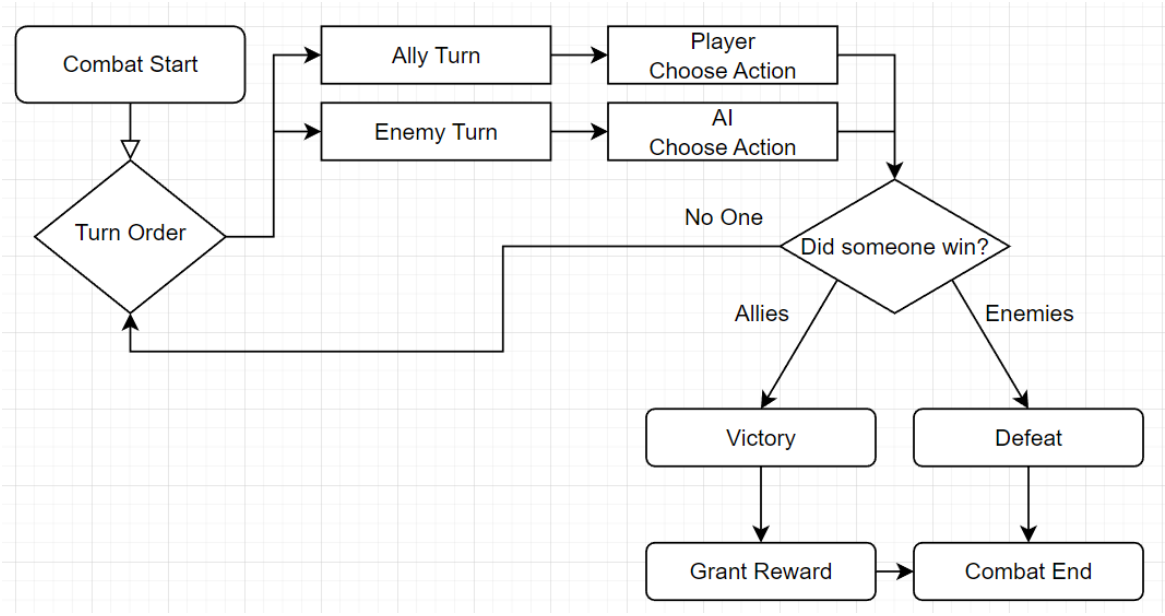

Рис. 11 - Спрощена діаграма логіки бою

Ця діаграма не включає у себе багато мілких речей - логіку переключення між боєм та експлорацією світу, логіку переключення назад, вирахування різних здатностей як союзних так і ворожих юнітів, ін. У цілому бойова система є найбільш складною системою у всій грі з точки зору об’єму та складності коду.

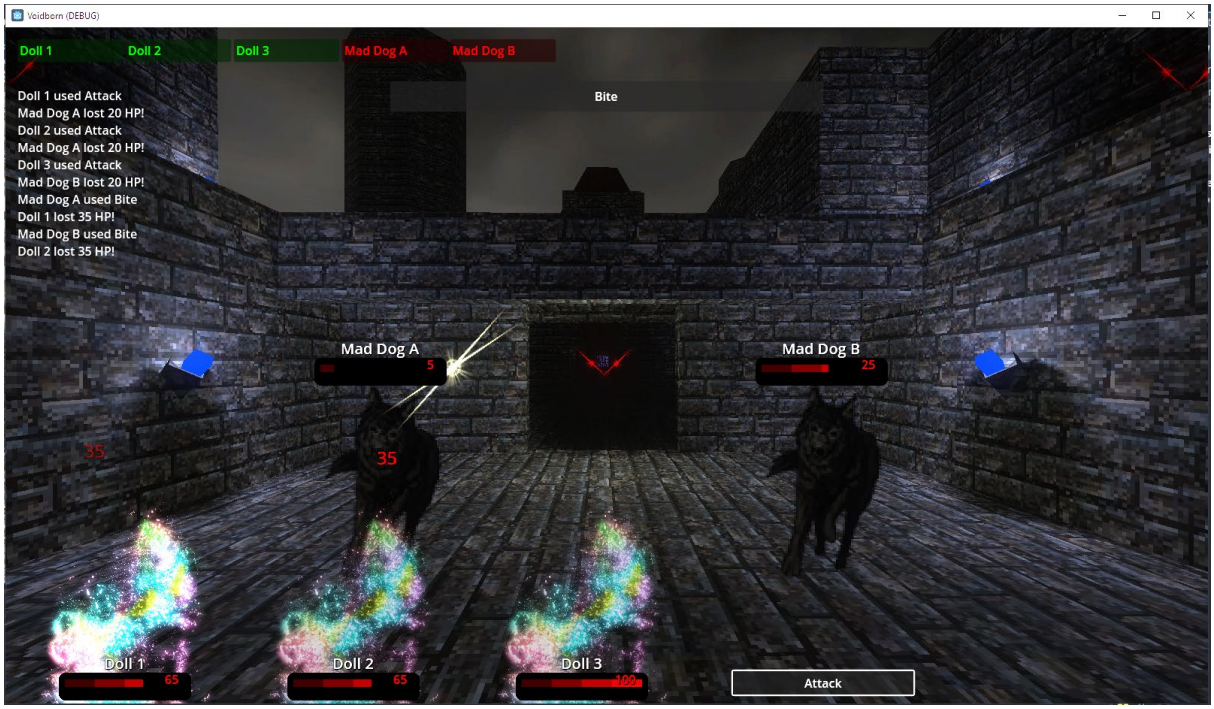

Рис. 12 - Бій в Voidborn

22

2.6 - Обгрунтування та організація вхідних та вихідних даних програми

Вхідні дані - дії та вибори гравця.

Вихідні дані - реакція ігрових систем та світу на дії гравця.

2.7 - Опис роботи розробленого програмного продукту

2.7.1 - Використані технічні засоби

Для розробки програми використовувалися наступні технічні засоби:

- Стаціонарний ПК (ОЗУ 16гб, процесор Intel i7-6700K, відеокарта GTX 970 4GB)

2.7.2 - Використані програмні засоби

Для розробки програми використовувалися наступні програмні засоби:

- Godot Game Engine, як ігровий рушій
- Aseprite для редагування текстур та інших картинок

2.7.3 - Виклик та завантаження програми

Гра не потребує встановлення та запускається як звичайний .exe-файл у будь-якому місці на диску.

2.7.4 - Опис інтерфейсу користувача

Гра повністю контролюється використовуючи клавіатуру. Екрани бою та gemcraft-ингу можно побачити на Рис.12 та Рис.10.

23

РОЗДІЛ 3 - ЕКОНОМІЧНА ЧАСТИНА

Під час розробки програмного забезпечення важливими етапами є визначення трудомісткості розробки і розрахунок витрат на створення програмного продукту.

3.1 - Визначення трудомісткості розробки програмного забезпечення

Задані дані:

1. передбачуване число операторів (підпрограм) – 2100;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,1;
4. годинна заробітна плата програміста – 4.6$/год [5];
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.2;
7. вартість машино-години ЕОМ, грн/год – 0.57 грн/год. (400 kw/h, 0.4 kwh [6]).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.2)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \quad (3.3)$$

24

де q - передбачуване число операторів;
C - коефіцієнт складності програми;
p - коефіцієнт корекції програми в ході її розробки.

$$Q = 2100 * 1.6 * (1 + 0.1) = 3696.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75..85) * k}, \text{людино} - \text{годин}. \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;
k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{3696 * 1{,}2}{80 * 1.2} = \frac{4435}{96} = 46, \text{людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) * k}, \text{людино} - \text{годин}, \quad (3.5)$$

$$t_a = \frac{3696}{25 * 1.2} = 123, \text{людино} - \text{годин}.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25) * k}, \text{людино} - \text{годин}, \quad (3.6)$$

$$t_n = \frac{3696}{25 * 1.2} = 123, \text{людино} - \text{годин},$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) * k}, \text{людино} - \text{годин}, \quad (3.7)$$

25

$$t_{\text{отл}} = \frac{3696}{5 * 1.2} = 616, \text{людино} - \text{годин};$$

- за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 * t_{\text{отл}}, \text{людино-годин}, \quad (3.8)$$

$$t_{\text{отл}}^{k} = 1{,}5 * 616 = 924, \text{людино-годин}.$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}, \text{людино-годин}, \quad (3.9)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{15..20 * k}, \text{людино} - \text{годин}, \quad (3.10)$$

$$t_{\text{др}} = \frac{3696}{20 * 1.2} = 148, \text{людино} - \text{годин}.$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \text{людино-годин}, \quad (3.11)$$

$$t_{\text{до}} = 0{,}75 * 148 = 111, \text{людино-година},$$

$$t_{\text{д}} = 148 + 111 = 259 \text{ людино-годин}.$$

Тепер розрахуємо трудомісткість ПЗ:

$t = 40 + 46 + 123 + 123 + 616 + 256 = 1204$, людино-годин.

3.2 - Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{\text{по}} = З_{\text{зп}} + З_{\text{мв}}, \text{грн}. \quad (3.12)$$

Заробітна плата виконавців визначається за формулою:

26

$$З_{зп} = t * С_{пр}, \text{грн}, \quad (3.13)$$

де t - загальна трудомісткість, людино-годин;
$C_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 1204 * 4.6 = 5538 \$ (202535 \text{ uah}).$$

Вартість машинного часу, необхідного для налагодження програми:

$$З_{мв} = t_{отл} * С_{мч}, \text{грн}, \quad (3.14)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год,
$C_{мч}$ - вартість машино-години ЕОМ, грн/год,
$C_{мч}$ = 14, грн/год.

$$З_{мв} = 616 * 0.57 = 351, \text{грн}.$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення ПЗ:

$$К_{по} = 202535 + 351 = 202886, \text{грн}.$$

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_k * F_p}, \text{міс.}, \quad (3.15)$$

де B_k - число виконавців (приймається 1),
F_p - місячний фонд робочого часу (40 годин на тиждень F_p = 176 годин).

$$Т = \frac{1204}{1 * 176} = 6.8, \text{міс}.$$

Висновок:

Вартість ПЗ становить 203 тис. грн. і не вимагає додаткових витрат. Очікуваний час розробки становить 6.8 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування, налагодження програми і підготовку документації.

27

28

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гра Chainmail URL: https://en.wikipedia.org/wiki/Chainmail_(game)
2. Гра D&D URL: https://en.wikipedia.org/wiki/Dungeons_%26_Dragons
3. Статистика продажів ігор на Nintendo DS URL: https://www.neogaf.com/threads/japan-ds-psp-sales-charts.177723/
4. ЗП Програмістів URL: https://www.unian.ua/society/skilki-zaroblyaye-programist-realni-cifri-novini-ukrajini-11538661.html
5. Ціни за світло URL: https://24tv.ua/economy/tarif-elektroenergiyu-kvitni-2023-bude-bez-zmin-skilki-zaraz_n2283031

29

ДОДАТОК А - КОД ПРОГРАМИ

FILE: \Assets\SFX\Materials\BasicSFX.gd
```
extends Node2D

signal finished()
@export var xy_random = Vector2(32, 32)

func _ready():
    $AnimatedSprite2D.animation_finished.connect(on_finished)
    $AnimatedSprite2D.position = Vector2(randi()%int(xy_random.x*2)-xy_random.x, randi()%int(xy_random.y*2)-xy_random.y)
    $AnimatedSprite2D.play()

func on_finished():
    finished.emit()
    queue_free()
```

FILE: \BreakableWall.gd
```
extends Node3D

@export var wall_name = "wall"
@export var required_flag = ""

func _ready():
    if not wall_name in Player.statefuls.keys():
        Player.statefuls[wall_name] = 0
    upd()

func interacted():
    if required_flag == "" or required_flag in Player.flags.keys():
        Player.statefuls[wall_name] = 1
        upd()

func upd():
    Util.set_collision_of_descendants($states, 0)
    for i in $states.get_children():
        i.visible = false
    $states.get_children()[Player.statefuls[wall_name]].visible = true
    Util.set_collision_of_descendants($states.get_children()[Player.statefuls[wall_name]], 1)
```

FILE: \ChangeLevelInteractible.gd
```
extends WorldInteractactible

@export var level: String
@export var spawn = 0

func interact():
    Util.root_node.change_level(level, spawn)
```

FILE: \Chest.gd
```
extends Node3D

@export var chest_name = "TEST CHEST"
@export var rewards = []
@export var set_flags = {"power_gloves": true}
```


30

```
@export var notches = 0
@export_multiline var popup = "Got power gloves!"

func _ready():
    if not chest_name in Player.statefuls.keys():
        Player.statefuls[chest_name] = 0
    upd()

func interacted():
    if Player.statefuls[chest_name] == 0:
        for i in rewards:
            Player.add_gem(i)
        for i in set_flags.keys():
            Player.flags[i] = set_flags[i]
        if popup != "":
            open_popup()
        Player.statefuls[chest_name] = 1
        Player.add_notches(notches)
        upd()

func open_popup():
    Util.world_player.disable()
    var msg = AcceptDialog.new()
    add_child(msg)
    msg.dialog_text = popup
    msg.exclusive = true
    msg.popup_centered()

    msg.confirmed.connect(popup_closed)
    msg.confirmed.connect(msg.queue_free)

    msg.canceled.connect(popup_closed)
    msg.canceled.connect(msg.queue_free)

func popup_closed():
    Util.world_player.enable()

func upd():
    for i in $lid.get_children():
        i.visible = false
    $lid.get_children()[Player.statefuls[chest_name]].visible = true
```

FILE: \CombatInteractibleRoot.gd

```
extends MeshInstance3D

@export var first_encounter: String
# Called when the node enters the scene tree for the first time.
func _ready():
    $Collision.interaction_finished.connect(upd)
    upd()

func upd():
    if first_encounter in Player.flags["completed_encounters"]:
        $Smoke.process_material = preload("res://Materials/FinishedParticles.tres")
        $Sprite3D.modulate = Color(0.7, 0.1, 0.1, 0.3)
    else:
        $Smoke.process_material = preload("res://Materials/UnfinishedParticles.tres")
        $Sprite3D.modulate = Color(1, 0, 0, 1)
```

FILE: \Interactible.gd
```
extends StaticBody3D
class_name WorldInteractactible

signal interaction_finished

func interact():
    finish_interact()

func finish_interact():
    emit_signal("interaction_finished")
```

FILE: \LevelRoot.gd
```
extends Node3D

func spawn_player(spawn):
    var player = preload("res://Scenes/player.tscn").instantiate()
    player.position = $Spawns.get_child(spawn).position
    player.rotation = $Spawns.get_child(spawn).rotation
    add_child(player)
```

FILE: \loot_panel.gd
```
extends Panel

func on_anim_finish(_anim_state):
    self.queue_free()
```

FILE: \note_root.gd
```
extends Node3D

@export_multiline var note_text = "Note text"
```

FILE: \PlayerController.gd
```
extends Node3D

@export var move_speed = 0.25
@export var look_deg = 25
@export var look_speed = 0.25
@export var height_adj_speed = 2
@export var height = 0.4

var cam_tween
var height_tween
var moving = false
var falling = false

var disable_stacks = 0

func _ready():
    $Camroot/Camera3d.current = true

func enable():
    disable_stacks -= 1
    if disable_stacks == 0:
```


32

```
        set_process(true)

func disable():
    disable_stacks += 1
    if disable_stacks > 0:
        set_process(false)

func _process(_delta):
    if Input.is_action_just_pressed("cancel") and Util.active_input(self):
        Util.player_interface.open_pause()
    elif Input.is_action_just_pressed("interact") and Util.active_input(self):
        var obj = $Rays/CastInteract.get_collider()
        if obj != null and obj is WorldInteractactible:
            disable()
            obj.connect("interaction_finished", enable, CONNECT_ONE_SHOT)
            obj.call("interact")
            return

    handle_cam()

    var distance_to_ground =
$Rays/CastDown.global_transform.origin.distance_to($Rays/CastDown.get_collision_point())
    var target_height = position.y
    falling = false
    if distance_to_ground != height:
        target_height -= distance_to_ground-height

    if target_height != position.y:
        if height_tween != null:
            height_tween.stop()
        position.y = target_height

    if (Input.is_action_pressed("movement", true) and not falling) and Util.active_input(self):
        handle_move()

func handle_cam():
    var look_vector = Vector3.ZERO
    if Input.is_action_pressed("cam_up") and Util.active_input(self):
        look_vector += Vector3(deg_to_rad(look_deg), 0, 0)

    if Input.is_action_pressed("cam_down") and Util.active_input(self):
        look_vector += Vector3(deg_to_rad(-look_deg), 0, 0)

    if Input.is_action_pressed("cam_left") and Util.active_input(self):
        look_vector += Vector3(0, deg_to_rad(look_deg), 0)

    if Input.is_action_pressed("cam_right") and Util.active_input(self):
        look_vector += Vector3(0, deg_to_rad(-look_deg), 0)

    if cam_tween != null:
        cam_tween.stop()
    cam_tween = create_tween()
    cam_tween.tween_property($Camroot/Camera3d, "rotation", look_vector, look_speed)

func handle_move():
    if not moving:
        var move_target = position
        var rotation_target = rotation

        if Input.is_action_pressed("move_forward") and !$Rays/CastForward.is_colliding() and Util.active_input(self):
            move_target = position + global_transform.basis.z*-1
        elif Input.is_action_pressed("move_back") and !$Rays/CastBack.is_colliding() and Util.active_input(self):
            move_target = position + global_transform.basis.z
```


33

```
        elif Input.is_action_pressed("move_left") and Util.active_input(self):
            rotation_target = rotation+Vector3(0,deg_to_rad(90),0)
        elif Input.is_action_pressed("move_right") and Util.active_input(self):
            rotation_target = rotation+Vector3(0,deg_to_rad(-90),0)

        if move_target != position or rotation_target != rotation:
            moving = true
            if move_target != position:
                var move_tween_x = create_tween()
                move_tween_x.tween_property(self, "position:x", move_target.x,
move_speed).set_trans(Tween.TRANS_SINE)
                move_tween_x.tween_callback(ended_move)

                var move_tween_z = create_tween()
                move_tween_z.tween_property(self, "position:z", move_target.z,
move_speed).set_trans(Tween.TRANS_SINE)
                move_tween_z.tween_callback(ended_move)
            else:
                var rotate_tween = create_tween()
                rotate_tween.tween_property(self, "rotation", rotation_target, move_speed).set_trans(Tween.TRANS_CIRC)
                rotate_tween.tween_callback(ended_move)

func ended_move():
    moving = false
```

FILE: \RootNode.gd

```
extends Node
class_name RootNode

var curr_level

func _ready():
    curr_level = find_child("3D")
    curr_level.spawn_player(0)
    Util.reload_node_references()
    Util.active_input_object = Util.world_player

func change_level(level, spawn=0):
    var level_instance = load(level).instantiate()
    remove_child(curr_level)
    curr_level.queue_free()
    add_child(level_instance)
    curr_level = level_instance
    curr_level.spawn_player(spawn)
    Util.reload_node_references()
```

FILE: \StatefulChangeInteractible.gd

```
extends WorldInteractactible

@export var stateful_name = "stateful"
@export var new_value = 0
@export var toggle_mode = false
@export var toggle_val = 1

func interact():
    if not toggle_mode:
        Player.upd_state(stateful_name, new_value)
    else:
        if Player.statefuls[stateful_name] == new_value:
```


34

```
            Player.upd_state(stateful_name, toggle_val)
        else:
            Player.upd_state(stateful_name, new_value)

    finish_interact()
```

FILE: \statefuls.gd

```
extends Node3D


# Called when the node enters the scene tree for the first time.
func _ready():
    for i in get_children():
        if not i.name in Player.statefuls.keys():
            Player.statefuls[i.name] = 0
    Player.statefuls_changed.connect(upd)
    upd()

func upd():
    Util.set_collision_of_descendants(self, 0)
    for i in get_children():
        for j in i.get_children():
            j.visible = false
        i.get_child(Player.statefuls[i.name]).visible = true
        Util.set_collision_of_descendants(i.get_child(Player.statefuls[i.name]), 1)
```

FILE: \AutoLoad\EncounterRewards.gd

```
extends Node

var rewards = {
    "res://Scenes/Encounters/test_enemies.tscn": [],
    "res://Scenes/Encounters/tutorial/first_fight.tscn":
        [
            "res://Scripts/Resources/Gems/Andradite.tres",
            "res://Scripts/Resources/Gems/Garnet.tres",
        ],
    "res://Scenes/Encounters/tutorial/boss.tscn":
        [

        ],
    "res://Scenes/Encounters/tutorial/dogs_dogs_dogs.tscn":
        [
            "res://Scripts/Resources/Gems/CutGarnet.tres"
        ],
    "res://Scenes/Encounters/tutorial/outside_elite.tscn":
        [
            "res://Scripts/Resources/Gems/SharpTopaz.tres"
        ],
    "res://Scenes/Encounters/tutorial/pre_boss.tscn":
        [
            "res://Scripts/Resources/Gems/Topaz.tres"
        ],
    "res://Scenes/Encounters/tutorial/rampart_chest.tscn":
        [
            "res://Scripts/Resources/Gems/Ruby.tres",
        ],
    "res://Scenes/Encounters/tutorial/rampart_lever.tscn":
        [
            "res://Scripts/Resources/Gems/RoseRuby.tres",
```


35

```
    ],
}
```

FILE: \AutoLoad\Player.gd

```
extends Node

@export var notches = 0

@export var gems = []

@export var flags = {"completed_encounters":[]}

@export var statefuls = {}

signal statefuls_changed
func upd_state(state_name, new_state):
    statefuls[state_name] = new_state
    statefuls_changed.emit()

func add_gem(path):
    var gem = load(path)
    gems.append(gem)
    sort_player_gems()
    var loot_panel = preload("res://Scenes/2D/loot_panel.tscn").instantiate()
    loot_panel.find_child("Label").text = gem.gem_name
    Util.player_interface.find_child("Loot").add_child(loot_panel)

func sort_player_gems():
    gems.sort_custom(func(a, b): return a.gem_name < b.gem_name)

func add_notches(amount):
    if amount <= 0:
        return

    var loot_panel = preload("res://Scenes/2D/loot_panel.tscn").instantiate()
    loot_panel.find_child("Label").text = str(amount) + " notches"
    Util.player_interface.find_child("Loot").add_child(loot_panel)
    notches += amount
```

FILE: \AutoLoad\SFX.gd

```
extends Node

func damage_sfx(node, text, color):
    var sfx = preload("res://Scenes/2D/damage_SFX.tscn").instantiate()
    sfx.upd(text, color, Vector2((randi() % 100) - 50, (randi() % 100) - 50))
    node.add_child(sfx)
```

FILE: \AutoLoad\Util.gd

```
extends Node

var root_node: RootNode
var world_player: Node
var combat_scene: Node
var player_interface: Node
var combat_interface_root: Node
var doll_root: Node
var active_input_object: Node
```


36

```
signal node_references_reloaded

func _ready():
	randomize()

func annihilate_children(node):
	var innocents = node.get_children()
	for i in innocents:
		node.remove_child(i)
		i.queue_free()

func stat_to_percent(stat_val, point_influence = 130):
	return 1.0-(1.0/((pow(stat_val, 1.2)+point_influence)/point_influence))

func get_hated_doll():
	var dolls = []
	for i in doll_root.get_children():
		if i.cHP > 0:
			for j in i.hate:
				dolls.append(i)
	return dolls.pick_random()

func reload_node_references():
	root_node = $"../ROOT"
	world_player = $"../ROOT/3D/Player"
	combat_scene = $"../ROOT/3D/Combat"
	player_interface = $"../ROOT/2D"
	combat_interface_root = $"../ROOT/2D/Combat"
	doll_root = $"../ROOT/2D/Dolls"
	node_references_reloaded.emit()

func active_input(node: Node):
	if node == active_input_object:
		return true
	if node == null:
		return false
	return active_input(node.get_parent())

func combat_log(txt):
	if combat_interface_root == null:
		return
	combat_interface_root.c_log(txt)

func set_collision_of_descendants(node, val):
	if node is CollisionObject3D:
		node.collision_layer = val

	for i in node.get_children():
		set_collision_of_descendants(i, val)
```

FILE: \Combat\AllyUnit.gd

```
extends UnitBase
class_name AllyUnit

var gem_board = GemBoard.new()
var sprite_folder_path = ""
var hate = 10

func get_sfx_node():
	return $SFXRoot
```


37

```
func start_turn():
    set_opaque()
    super.start_turn()

func end_turn():
    set_transparent()
    super.end_turn()

func set_opaque():
    $Bust.modulate = Color(1, 1, 1, 1)

func set_transparent():
    $Bust.modulate = Color(1, 1, 1, 0.5)

func load_sprite(path):
    if FileAccess.file_exists(path + "/portrait.json"):
        var str_json = FileAccess.get_file_as_string(path + "/portrait.json")
        var json_content = JSON.parse_string(str_json)
        var image = Image.new()
        var error = image.load(path + json_content["base"])
        if error != OK:
            return
        $Bust.texture = ImageTexture.new().create_from_image(image)
        $Bust.scale = Vector2(json_content["zoom"] + 1, json_content["zoom"] + 1)
        $Bust.offset = Vector2(json_content["offset_x"], json_content["offset_y"])
        sprite_folder_path = path

func get_stat_val(stat):
    var mod = 0
    for i in gem_board.inserted_gems:
        if i[3]: # Gem is amped
            if stat in i[0].amp_stat_adjustments:
                mod += i[0].amp_stat_adjustments[stat]
        else:
            if stat in i[0].stat_adjustments:
                mod += i[0].stat_adjustments[stat]
    return mod + super.get_stat_val(stat)

func update_skills():
    super.update_skills()
    hate = 10
    for i in gem_board.inserted_gems:
        hate += i[0].hate_mod
        if i[3]:
            for j in i[0].amp_skills:
                skills.append(load(j).new())
            for j in i[0].amp_tags:
                tags.append(j)
        else:
            for j in i[0].skills:
                skills.append(load(j).new())
            for j in i[0].tags:
                tags.append(j)



FILE: \Combat\CombatRoot.gd
extends Node3D

signal combat_started
signal combat_ended
signal turn_order_changed
```


38

```
var look_deg = 25
var look_speed = 0.25
var cam_tween
var curr_encounter: String
var turn_order = []

func _process(delta):
    if $CamRoot/Camera.current:
        handle_cam(delta)

func start_combat(encounter):
    curr_encounter = encounter
    add_child(load(encounter).instantiate())
    turn_order = []
    for i in get_everyone():
        order_insert(i, (1000-i.get_stat_val(UnitBase.SPD))/10)

    for i in get_enemies():
        i.tree_exited.connect(check_end_combat)
        i.turn_started.connect(glance.bind(i))

    for i in get_allies():
        i.set_transparent()
        i.died.connect(order_remove.bind(i))
        i.died.connect(check_end_combat)
        i.revived.connect(order_insert.bind(i, 200))

    $CamRoot/Camera.set_current(true)
    Util.doll_root.visible = true
    Util.combat_interface_root.visible = true
    emit_signal("combat_started")
    next_turn()
    Util.combat_interface_root.find_child("CombatLog").visible = true
    Util.combat_interface_root.find_child("CombatLog").text = ""

func end_combat():
    Util.combat_interface_root.find_child("CombatLog").visible = false
    $CamRoot/Camera.set_current(false)
    Util.doll_root.visible = false
    Util.combat_interface_root.visible = false
    for i in Util.doll_root.get_children():
        i.reset()
    $Enemies.queue_free()
    emit_signal("combat_ended")

func check_end_combat():
    if get_node_or_null("Enemies") == null:
        return true

    if get_enemies().is_empty():
        win()
        return true

    var won = true
    for i in get_enemies():
        if i.cHP > 0:
            won = false
            break

    if won:
        win()
        return true
```


39

```
    var lost = true
    for i in Util.doll_root.get_children():
        if i.cHP > 0:
            lost = false
            break

    if lost:
        end_combat()
        return true
    return false

func win():
    if not curr_encounter in Player.flags["completed_encounters"]:
        Player.flags["completed_encounters"].append(curr_encounter)
        for i in EncounterRewards.rewards[curr_encounter]:
            Player.add_gem(i)
    end_combat()

func next_turn():
    await get_tree().process_frame
    var next = next_in_order()
    next.connect("turn_ended", next_turn, CONNECT_ONE_SHOT)
    next.start_turn()

func order_insert(object, delay):
    for i in turn_order:
        if i[0] == delay:
            return order_insert(object, delay+1)
    turn_order.append([delay, object])
    turn_order.sort_custom(func(a, b): return a[0] < b[0])
    emit_signal("turn_order_changed")

func order_remove(object):
    for i in turn_order:
        if i[1] == object:
            turn_order.erase(i)
            emit_signal("turn_order_changed")
            return

func next_in_order():
    var lowest_nonzero = [1000, null]
    var zero = []
    for i in turn_order:
        if i[0] == 0:
            zero = i
        elif i[0] < lowest_nonzero[0]:
            lowest_nonzero = i

    var adj = lowest_nonzero[0]
    turn_order.erase(zero)

    for i in turn_order:
        i[0] -= adj

    emit_signal("turn_order_changed")
    return lowest_nonzero[1]

func get_enemies():
    return $Enemies.get_children()

func get_allies():
    return Util.doll_root.get_children()
```


40

```
func get_everyone():
    return get_allies() + get_enemies()

var look_target = Vector3.ZERO
var override_time = 0.0

func glance(enemy):
    var prev_rot = $CamRoot/Camera.rotation
    $CamRoot/Camera.look_at(enemy.get_camera_focus())
    look_target = $CamRoot/Camera.rotation
    $CamRoot/Camera.rotation = prev_rot
    override_time = 1.0

func handle_cam(delta):
    var look_vector = look_target

    override_time -= delta
    if override_time < 0:
        look_vector = Vector3.ZERO

    if Input.is_action_pressed("cam_up"):
        look_vector += Vector3(deg_to_rad(look_deg), 0, 0)

    if Input.is_action_pressed("cam_down"):
        look_vector += Vector3(deg_to_rad(-look_deg), 0, 0)

    if Input.is_action_pressed("cam_left"):
        look_vector += Vector3(0, deg_to_rad(look_deg), 0)

    if Input.is_action_pressed("cam_right"):
        look_vector += Vector3(0, deg_to_rad(-look_deg), 0)

    if cam_tween != null:
        cam_tween.stop()
    cam_tween = create_tween()
    cam_tween.tween_property($CamRoot/Camera, "rotation", look_vector, look_speed)
```

FILE: \Combat\EnemyUnit.gd

```
extends UnitBase
class_name EnemyUnit

func start_turn():
    super.start_turn()
    var cont = not Util.combat_scene.check_end_combat()
    if cont:
        return true
    else:
        return false

func set_cHP(val):
    super.set_cHP(val)
    if self.cHP <= 0:
        Util.combat_scene.order_remove(self)
        self.queue_free()

func get_sfx_node():
    return $"2dAnchor/2dRoot/SFXRoot"

func get_camera_focus():
    return $Sprite3D.global_position
```


41

FILE: \Combat\UnitBase.gd

```
extends Node
class_name UnitBase

const ATK = "ATK"
const SPD = "SPD"
const DEF = "DEF"
const RES = "RES"
const HP = "HP"
const HPReg = "HPReg"
const MP = "MP"
const MPReg = "MPReg"
const CritChance = "CritChance"
const CritPow = "CritPow"
const StunRes = "StunRes"
const HealPow = "HealPow"
const IncHealMult = "HealMult"
const Luck = "LUK"

const PhysDMG = "PHYS"
const MagDMG = "MAG"
const TrueDMG = "TRUE"

const stats = [ATK, SPD, DEF, RES, HP, HPReg, MP, MPReg, CritChance, CritPow, StunRes, HealPow, IncHealMult,
Luck]

const percentile_stats = [
    DEF,
    RES,
    CritChance,
    StunRes,
]

@export var baseStats = {
    ATK: 20,
    SPD: 0,
    DEF: 0,
    RES: 0,
    HP: 100,
    HPReg: 0,
    MP: 50,
    MPReg: 0,
    CritChance: 5,
    CritPow: 50,
    StunRes: 0,
    HealPow: 0,
    IncHealMult: 0,
    Luck: 0,
}

@export var baseSkills = [
    "res://Scripts/Combat/Skills/BasicAttack.gd",
]

var skills = []

@export var baseTags = []
var tags = []

@export var unitName = "Base Unit"
```


42

```
signal died
signal revived
var dead = false

var cHP: int
signal cHP_changed(old, new)
func set_cHP(new: int):
    var old = cHP
    if new <= 0:
        new = 0
        dead = true
        emit_signal("died")

    if new > get_stat_val(HP):
        new = get_stat_val(HP)

    if dead and new > 0:
        dead = false
        emit_signal("revived")

    cHP = new
    if new > old:
        SFX.damage_sfx(get_sfx_node(), new - old, Color.GREEN)
        Util.combat_log(unitName + " recovered " + str(new - old) + " HP!")
    if old > new:
        SFX.damage_sfx(get_sfx_node(), old - new, Color.RED)
        Util.combat_log(unitName + " lost " + str(old - new) + " HP!")

    emit_signal("cHP_changed", old, new)

func get_sfx_node():
    pass

var cMP: int
signal cMP_changed(old, new)
func set_cMP(new: int):
    var old = cMP
    if new <= 0:
        new = 0
    cMP = new
    if old < new:
        Util.combat_log(unitName + " gained " + str(new - old) + " MP!")
    emit_signal("cMP_changed", old, new)

var barrier: int
signal barrier_changed(old, new)
func set_barrier(new: int):
    var old = barrier
    if new > get_stat_val(HP):
        barrier = get_stat_val(HP)
    else:
        barrier = new
    if old != new:
        emit_signal("barrier_changed", old, new)

var burn: int
signal burn_changed(old, new)
func set_burn(new: int):
    var old = burn
    burn = new
    emit_signal("burn_changed", old, new)
```


43

```
var poison: int
signal poison_changed(old, new)
func set_poison(new: int):
    var old = poison
    poison = new
    emit_signal("poison_changed", old, new)

var critical: bool
signal critical_changed(old, new)
func set_critical(new: bool):
    var old = critical
    critical = new
    emit_signal("critical_changed", old, new)

var stun: bool
signal stun_changed(old, new)
func set_stun(new: bool):
    var old = stun
    stun = new
    emit_signal("stun_changed", old, new)

var stun_immune: bool
signal stun_immune_changed(old, new)
func set_stun_immune(new: bool):
    var old = stun_immune
    stun_immune = new
    emit_signal("stun_immune_changed", old, new)

signal turn_started
signal turn_ended

func start_turn():
    if get_stat_val(HPReg) > 0 and not dead:
        deal_healing(get_stat_val(HPReg), self)
    if get_stat_val(MPReg) > 0:
        set_cMP(cMP + get_stat_val(MPReg))
    emit_signal("turn_started")

func end_turn():
    if randf() < Util.stat_to_percent(get_stat_val(CritChance)):
        set_critical(true)
    else:
        set_critical(false)
    emit_signal("turn_ended")

func take_dmg(amount, source, type=PhysDMG):
    var modified_amount: int = amount

    if barrier > 0:
        modified_amount -= barrier
        set_barrier(max(barrier - amount, 0))

    if modified_amount <= 0:
        return

    if type == PhysDMG:
        modified_amount = modified_amount*(1-Util.stat_to_percent(get_stat_val(DEF)))
    if type == MagDMG:
        modified_amount = modified_amount*(1-Util.stat_to_percent(get_stat_val(RES)))
    set_cHP(cHP - modified_amount)

func take_healing(amount, source):
    if amount <= 0:
```


44

```
        return

    var modified_amount: int = amount

    modified_amount *= 1.0+get_stat_val(IncHealMult)/100.0

    set_cHP(cHP + modified_amount)

func deal_dmg(amount, target, type=PhysDMG):
    var modified_amount: int = amount
    if critical:
        modified_amount = modified_amount*(1.0+get_stat_val(CritPow)/100.0)
    target.take_dmg(modified_amount, self, type)

func deal_healing(amount, target):
    var modified_amount: int = amount
    modified_amount = modified_amount*(1.0+get_stat_val(HealPow)/100.0)
    target.take_healing(modified_amount, self)

func get_stat_val(stat: String):
    return baseStats[stat]

func update_skills():
    skills = []
    for i in baseSkills:
        skills.append(load(i).new())
    tags = baseTags.duplicate()

signal reseted

func reset():
    update_skills()
    set_cHP(get_stat_val(HP))
    set_cMP(0)
    if "HalfMaxMP" in tags:
        set_cMP(get_stat_val(MP)/2)
    elif "MaxMP" in tags:
        set_cMP(get_stat_val(MP))
    set_barrier(0)
    set_burn(0)
    set_poison(0)
    set_critical(false)
    set_stun(false)
    set_stun_immune(false)
    reseted.emit()

func _ready():
    reset()
```

FILE: \Combat\AI\RandomSkillNoOverride.gd

```
extends EnemyUnit

func start_turn():
    if super.start_turn():
        var skill = skills.pick_random()
        skill.use(self)
```

FILE: \Combat\AI\RandomSkillRandomTarget.gd

```
extends EnemyUnit
```


45

```
func start_turn():
    if super.start_turn():
        var skill = skills.pick_random()
        var target = [Util.get_hated_doll()]
        skill.use(self, target)
```

FILE: \Combat\EnemySkills\Bite.gd

```
extends BaseSkill

func _init():
    skillName = "Bite"
    skillDesc = "ENEMY SKILL"

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_random_ally)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/basic_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        user.deal_dmg(user.get_stat_val(UnitBase.ATK), targets[0])
        Util.combat_interface_root.find_child("CombatActions").visible = true
        finished.emit()
        user.end_turn()
        reinsert(user, skill_delay_medium)
```

FILE: \Combat\Skills\AllGuard.gd

```
extends BaseSkill

func _init():
    skillName = "All-Guard"
    skillDesc = "Slow.\nCreates a barrier for 10% of user's max HP for all allies."

func use(user, _target_override = null):
    Util.combat_interface_root.find_child("CombatActions").visible = false
    Util.combat_interface_root.announce(skillName)
    Util.combat_log(user.unitName + " used " + skillName)
    var sfx = preload("res://Assets/SFX/SFX/yellow_shield_sfx.tscn").instantiate()
    user.get_sfx_node().add_child(sfx)

    if user is AllyUnit:
        for i in Util.combat_scene.get_allies():
            if i != user:
                var sec_sfx = preload("res://Assets/SFX/SFX/yellow_shield_sfx.tscn").instantiate()
                i.get_sfx_node().add_child(sec_sfx)
    else:
        for i in Util.combat_scene.get_enemies():
            if i != user:
                var sec_sfx = preload("res://Assets/SFX/SFX/yellow_shield_sfx.tscn").instantiate()
                i.get_sfx_node().add_child(sec_sfx)
```


46

```
    await sfx.finished

    if user is AllyUnit:
        for i in Util.combat_scene.get_allies():
            i.set_barrier(user.get_stat_val(UnitBase.HP)*0.10 + i.barrier)
    else:
        for i in Util.combat_scene.get_enemies():
            i.set_barrier(user.get_stat_val(UnitBase.HP)*0.10 + i.barrier)

    Util.combat_interface_root.find_child("CombatActions").visible = true
    finished.emit()
    user.end_turn()
    reinsert(user, skill_delay_medium)
```

FILE: \Combat\Skills\AllHeal.gd

```
extends BaseSkill

func _init():
    skillName = "All-Heal"
    skillDesc = "Costs 50 MP.\nRecover 20% of user's max HP to all allies."

func check_usability(user):
    if user.cMP < 50:
        return false
    return true

func use(user, _target_override = null):
    Util.combat_interface_root.find_child("CombatActions").visible = false
    Util.combat_interface_root.announce(skillName)
    Util.combat_log(user.unitName + " used " + skillName)
    var sfx = preload("res://Assets/SFX/SFX/heal_sfx.tscn").instantiate()
    user.get_sfx_node().add_child(sfx)

    if user is AllyUnit:
        for i in Util.combat_scene.get_allies():
            if i != user:
                var sec_sfx = preload("res://Assets/SFX/SFX/heal_sfx.tscn").instantiate()
                i.get_sfx_node().add_child(sec_sfx)
    else:
        for i in Util.combat_scene.get_enemies():
            if i != user:
                var sec_sfx = preload("res://Assets/SFX/SFX/heal_sfx.tscn").instantiate()
                i.get_sfx_node().add_child(sec_sfx)

    await sfx.finished

    if user is AllyUnit:
        for i in Util.combat_scene.get_allies():
            if i.cHP > 0:
                user.deal_healing(user.get_stat_val(UnitBase.HP)*0.2, i)
    else:
        for i in Util.combat_scene.get_enemies():
            if i.cHP > 0:
                user.deal_healing(user.get_stat_val(UnitBase.HP)*0.2, i)

    Util.combat_interface_root.find_child("CombatActions").visible = true

    user.set_cMP(user.cMP - 50)

    finished.emit()
```


47

```
    user.end_turn()
    reinsert(user, skill_delay_medium)
```

FILE: \Combat\Skills\BarrierBlade.gd

```
extends BaseSkill

func _init():
    skillName = "Barrier Blade"
    skillDesc = "Slow.\nDeal 50% DEF + 100% Barrier as physical damage. Consumes all barrier on use. Only usable when you have barrier."

func check_usability(user):
    if user.barrier <= 0:
        return false
    return true

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_select_enemy)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/yellow_ice_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        user.deal_dmg(user.get_stat_val(UnitBase.DEF)*0.5 + user.barrier, targets[0])
        user.set_barrier(0)
        Util.combat_interface_root.find_child("CombatActions").visible = true

        finished.emit()
        user.end_turn()
        reinsert(user, skill_delay_high)
```

FILE: \Combat\Skills\BaseSkill.gd

```
extends Resource
class_name BaseSkill

const skill_delay_low = 7000
const skill_delay_medium = 10000
const skill_delay_high = 13000

const target_self = "self"
const target_allies = "allies"
const target_enemies = "enemies"
const target_random_ally = "random_ally"
const target_random_enemy = "random_enemy"
const target_select_ally = "select_ally"
const target_select_enemy = "select_enemy"

signal finished()

@export var skillName = "Base skill"
@export var skillDesc = "Skill description goes here"
```


48

```
func use(user, target_override = null):
    print("Base skill used by:")
    print(user)
    finished.emit()
    user.end_turn()
    reinsert(user, skill_delay_medium)

func reinsert(user, skill_delay):
    Util.combat_scene.order_insert(user, skill_delay/(100+user.get_stat_val("SPD")))

func check_usability(user):
    return true

func get_targets(user, targeting_mode):
    if targeting_mode == target_self:
        return [user]
    elif targeting_mode == target_allies:
        return Util.combat_scene.get_allies()
    elif targeting_mode == target_enemies:
        return Util.combat_scene.get_enemies()
    elif targeting_mode == target_random_ally:
        return [Util.combat_scene.get_allies().pick_random()]
    elif targeting_mode == target_random_enemy:
        return [Util.combat_scene.get_enemies().pick_random()]
    elif targeting_mode == target_select_ally:
        Util.combat_interface_root.get_target(Util.combat_scene.get_allies())
        var target = await Util.combat_interface_root.target_selected
        return [target]
    elif targeting_mode == target_select_enemy:
        Util.combat_interface_root.get_target(Util.combat_scene.get_enemies())
        var target = await Util.combat_interface_root.target_selected
        return [target]

func get_description():
    return skillDesc
```

FILE: \Combat\Skills\BasicAttack.gd

```
extends BaseSkill

func _init():
    skillName = "Attack"
    skillDesc = "Basic Attack skill. Deals 100% ATK damage to one enemy."

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_select_enemy)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/basic_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        user.deal_dmg(user.get_stat_val(UnitBase.ATK), targets[0])
        Util.combat_interface_root.find_child("CombatActions").visible = true
```


49

```
        finished.emit()
        user.end_turn()
        reinsert(user, skill_delay_medium)
```

FILE: \Combat\Skills\Focus.gd

```
extends BaseSkill

func _init():
    skillName = "Focus"
    skillDesc = "Fast.\nRecovers 20% of max MP."

func use(user, _target_override = null):
    Util.combat_interface_root.find_child("CombatActions").visible = false
    Util.combat_interface_root.announce(skillName)
    Util.combat_log(user.unitName + " used " + skillName)

    var sfx = preload("res://Assets/SFX/SFX/focus_sfx.tscn").instantiate()
    user.get_sfx_node().add_child(sfx)
    await sfx.finished

    user.set_cMP(user.get_stat_val(UnitBase.MP)*0.20 + user.cMP)
    Util.combat_interface_root.find_child("CombatActions").visible = true
    finished.emit()
    user.end_turn()
    reinsert(user, skill_delay_low)
```

FILE: \Combat\Skills\Guard.gd

```
extends BaseSkill

func _init():
    skillName = "Guard"
    skillDesc = "Creates a barrier for 15% of max HP."

func use(user, _target_override = null):
    Util.combat_interface_root.find_child("CombatActions").visible = false
    Util.combat_interface_root.announce(skillName)
    Util.combat_log(user.unitName + " used " + skillName)

    var sfx = preload("res://Assets/SFX/SFX/yellow_shield_sfx.tscn").instantiate()
    user.get_sfx_node().add_child(sfx)
    await sfx.finished

    user.set_barrier(user.get_stat_val(UnitBase.HP)*0.15 + user.barrier)
    Util.combat_interface_root.find_child("CombatActions").visible = true
    finished.emit()
    user.end_turn()
    reinsert(user, skill_delay_medium)
```

FILE: \Combat\Skills\Heal.gd

```
extends BaseSkill

func _init():
    skillName = "Heal"
    skillDesc = "Costs 20 MP.\nRecover 20% of user's max HP to self or ally."

func check_usability(user):
    if user.cMP < 20:
```


50

```
        return false
    return true

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_select_ally)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/heal_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        if targets[0].cHP > 0:
            user.deal_healing(user.get_stat_val(UnitBase.HP)*0.2, targets[0])
        Util.combat_interface_root.find_child("CombatActions").visible = true

        user.set_cMP(user.cMP - 20)

        finished.emit()
        user.end_turn()
        reinsert(user, skill_delay_medium)



FILE: \Combat\Skills\LightningStrike.gd
extends BaseSkill

func _init():
    skillName = "Lightning Strike"
    skillDesc = "Fast, costs 30 MP.\nDeals 140% ATK magic damage to one enemy."

func check_usability(user):
    if user.cMP < 30:
        return false
    return true

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_select_enemy)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/thunder_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        user.deal_dmg(user.get_stat_val(UnitBase.ATK)*1.4, targets[0])
        Util.combat_interface_root.find_child("CombatActions").visible = true

        user.set_cMP(user.cMP - 30)

        finished.emit()
```


51

```
        user.end_turn()
        reinsert(user, skill_delay_low)
```

FILE: \Combat\Skills\PowerStrike.gd

```
extends BaseSkill

func _init():
    skillName = "Power Strike"
    skillDesc = "Slow.\nDeals 140% ATK damage to one enemy."

func use(user, target_override = null):
    var targets
    if target_override == null:
        targets = await get_targets(user, target_select_enemy)
    else:
        targets = target_override
    if targets[0] != null:
        Util.combat_interface_root.find_child("CombatActions").visible = false
        Util.combat_interface_root.announce(skillName)
        Util.combat_log(user.unitName + " used " + skillName)

        var sfx = preload("res://Assets/SFX/SFX/power_strike_sfx.tscn").instantiate()
        targets[0].get_sfx_node().add_child(sfx)
        await sfx.finished

        user.deal_dmg(user.get_stat_val(UnitBase.ATK)*1.4, targets[0], UnitBase.MagDMG)
        Util.combat_interface_root.find_child("CombatActions").visible = true
        finished.emit()
        user.end_turn()
        reinsert(user, skill_delay_high)
```

FILE: \Effects\TextureScroll.gd

```
extends MeshInstance3D

@export var scroll = Vector3(0, 0, 0)

func _ready():
    pass

func _process(delta):
    get_surface_override_material(0).uv1_offset += scroll*delta
```

FILE: \Interactibles\CombatInteractible.gd

```
extends WorldInteractactible

func _ready():
    if get_parent().first_encounter in Player.flags["completed_encounters"]:
        collision_mask = 2
        collision_layer = 2

func interact():
    Util.combat_scene.connect("combat_ended", finish_interact, CONNECT_ONE_SHOT)
    Util.combat_scene.start_combat(get_parent().first_encounter)

func finish_interact():
    emit_signal("interaction_finished")
    if get_parent().first_encounter in Player.flags["completed_encounters"]:
```


52

```
        collision_mask = 2
        collision_layer = 2
    else:
        collision_mask = 3
        collision_layer = 3
```

FILE: \Interactibles\NoteInteractible.gd

```
extends WorldInteractactible

func interact():
    var msg = AcceptDialog.new()
    add_child(msg)
    msg.dialog_text = get_parent().note_text
    msg.exclusive = true
    msg.popup_centered()
    msg.confirmed.connect(finish_interact)
    msg.confirmed.connect(msg.queue_free)

    msg.canceled.connect(finish_interact)
    msg.canceled.connect(msg.queue_free)
```

FILE: \Resources\BaseGem.gd

```
extends Resource
class_name BaseGem

@export var shape = [
    [false, false, false],
    [false, true, true],
    [false, true, false],
]
@export var core_location = Vector2(1, 1)
@export var color = Color(1, 0, 1, 1)
@export var skills = []
@export var stat_adjustments = {}
@export var tags = []
@export var amp_skills = []
@export var amp_stat_adjustments = {}
@export var amp_tags = []

@export var gem_name = "Base gem"

@export_multiline var description = "Normal gem description"
@export_multiline var amp_description = "Amp gem description"

@export var hate_mod = 0

func rotate_shape():
    var new_shape = shape.duplicate(true)
    core_location = Vector2(2 - core_location.y, core_location.x)
    for i in range(3):
        for j in range(3):
            new_shape[i][j] = shape[2 - j][i]
    shape = new_shape

func get_desc(amp = false):
    if not amp:
        return description
    else:
        return amp_description
```


53

FILE: \Resources\GemBoard.gd

```
extends Resource
class_name GemBoard

var unlocked_spaces = [
    [false, false, false, false, false],
    [false, true, true, true, false],
    [false, true, true, true, false],
    [false, true, true, true, false],
    [false, false, false, false, false],
]

var free_spaces = [
    [true, true, true, true, true],
    [true, true, true, true, true],
    [true, true, true, true, true],
    [true, true, true, true, true],
    [true, true, true, true, true],
]

var inserted_gems = []

func is_space_empty(x, y):
    if x < 0 or x > 4 or y < 0 or y > 4:
        return false
    return unlocked_spaces[y][x] and free_spaces[y][x]

func check_gem_insertable(shape, offset_x, offset_y):
    for i in range(3):
        for j in range(3):
            if shape[j][i] == false or is_space_empty(i+offset_y, j+offset_x):
                continue
            else:
                return false
    return true

func insert_gem(gem, offset_x, offset_y):
    inserted_gems.append([gem, offset_x, offset_y, false])
    for i in range(3):
        for j in range(3):
            if gem.shape[i][j] == true:
                free_spaces[i + offset_y - 1][j + offset_x - 1] = false
    recheck_links()

func remove_gem(gem):
    for ins_gem in inserted_gems:
        if ins_gem[0] == gem:
            for i in range(3):
                for j in range(3):
                    if gem.shape[i][j] == true:
                        free_spaces[i + ins_gem[2] - 1][j + ins_gem[1] - 1] = true
            inserted_gems.erase(ins_gem)
    recheck_links()

func recheck_links():
    var core_positions = []
    for ins_gem in inserted_gems:
        core_positions.append(ins_gem[0].core_location + Vector2(ins_gem[1], ins_gem[2]))
    for ins_gem in inserted_gems:
        ins_gem[3] = false
```


54

```
        var gem_core_loc = ins_gem[0].core_location + Vector2(ins_gem[1], ins_gem[2])
        for i in [Vector2(1, 0), Vector2(0, 1), Vector2(-1, 0), Vector2(0, -1)]:
            if core_positions.has(gem_core_loc + i):
                ins_gem[3] = true


func get_gem_at_pos(x, y):
    if free_spaces[x][y] == true:
        return null
    for ins_gem in inserted_gems:
        for i in range(3):
            for j in range(3):
                if y == i+ins_gem[1]-1 and x == j+ins_gem[2]-1:
                    if ins_gem[0].shape[j][i]:
                        return [ins_gem[0], ins_gem[3]]
    return null
```

FILE: \UI\2dAnchorRoot.gd

```
extends Control


# Called when the node enters the scene tree for the first time.
func _ready():
    pass # Replace with function body.


func _process(_delta):
    if is_instance_valid(get_parent()):
        position = get_viewport().get_camera_3d().unproject_position(get_parent().to_global(Vector3.ZERO))
    else:
        queue_free()
```

FILE: \UI\2Droot.gd

```
extends Control

var pause: Node

func _ready():
    pause = $Pause

func open_pause():
    for i in $Dolls.get_children():
        i.set_opaque()
    pause.visible = true
    pause.set_process(true)
    Util.active_input_object = pause
    $Pause/PauseMenu/GemcraftBtn.grab_focus()
    Util.doll_root.visible = true

func close_pause():
    pause.visible = false
    pause.set_process(false)
    Util.active_input_object = Util.world_player
    Util.doll_root.visible = false
```

FILE: \UI\AppearanceBtn.gd

```
extends GrabberButton
```

```
var idx = 0

func _on_pressed():
    $PopupMenu.clear()
    for i in Util.doll_root.get_children():
        $PopupMenu.add_item(i.unitName)

    $PopupMenu.popup_centered()

func _on_popup_menu_index_pressed(index):
    idx = index
    $PopupMenu.hide()
    $FileDialog.popup_centered()

func _on_file_dialog_dir_selected(dir):
    Util.doll_root.get_child(idx).load_sprite(dir)
```

FILE: \UI\CombatActions.gd

```
extends VBoxContainer

func _ready():
    Util.node_references_reloaded.connect(upd, CONNECT_ONE_SHOT)

func upd():
    for i in Util.doll_root.get_children():
        i.connect("turn_started", display_actions.bind(i))
        i.connect("turn_ended", hide_actions)

func display_actions(unit):
    hide_actions()
    for i in unit.skills:
        var btn = Button.new()
        btn.text = i.skillName
        if not i.check_usability(unit):
            btn.disabled = true
        else:
            btn.connect("pressed", i.use.bind(unit))
        add_child(btn)
    get_children()[0].grab_focus()

func hide_actions():
    for i in get_children():
        remove_child(i)
        i.queue_free()
```

FILE: \UI\CombatInterfaceRoot.gd

```
extends Control

signal target_selected(target)

func get_target(choices):
    $CombatActions.set_process(false)
    $CombatActions.modulate = Color(1.0, 1.0, 1.0, 0.5)
    for i in choices:
        var btn = Button.new()
        btn.text = i.unitName
        btn.pressed.connect(target_button_pressed.bind(i))
        $TargetChoice.add_child(btn)
```

```
    var cancel_btn = Button.new()
    cancel_btn.text = "Cancel"
    cancel_btn.pressed.connect(target_button_pressed.bind(null))
    $TargetChoice.add_child(cancel_btn)
    $TargetChoice.visible = true
    $CombatActions.visible = false

    $TargetChoice.get_children()[0].grab_focus()

func target_button_pressed(target):
    $TargetChoice.visible = false
    $CombatActions.visible = true
    Util.annihilate_children($TargetChoice)
    target_selected.emit(target)
    $CombatActions.modulate = Color(1.0, 1.0, 1.0, 1.0)
    $CombatActions.set_process(true)
    if target == null:
        $CombatActions.get_children()[0].grab_focus()

func announce(text):
    $Announcement/Label.text = text
    $Announcement/AnimationPlayer.stop()
    $Announcement/AnimationPlayer.play("announce")

func c_log(text):
    $CombatLog.text += "\n" + text
```

FILE: \UI\damage_SFX.gd

```
extends Control

func _ready():
    $Control/AnimationPlayer.animation_finished.connect(self_destruct)

func upd(text, color, offset):
    $Control/Label.modulate = color
    $Control/Label.text = str(text)
    position = offset

func self_destruct(_name):
    self.queue_free()
```

FILE: \UI\DollDescriptor.gd

```
extends RichTextLabel

func upd(doll):
    var txt = doll.unitName + "\n"
    for i in UnitBase.stats:
        if doll.get_stat_val(i) != 0:
            txt += i + ": "
            txt += str(doll.get_stat_val(i))
            if i in UnitBase.percentile_stats:
                txt += " (" + ("%.2f" % (Util.stat_to_percent(doll.get_stat_val(i))*100)) + " %)"
            txt += "\n"
    text = txt
```

FILE: \UI\GemcraftScreen.gd

```
extends Control
```


57

```
@export var visual_placement_offset = Vector2(-6, -6)

var inserting_gem = false
var held_gem: BaseGem = null
var visualgem = null
var curr_doll = null
var visualgem_template = preload("res://Scenes/2D/visual_gem.tscn")
var gembtn_template = preload("res://Scenes/2D/gem.tscn")

var current_page = 1
var pages = 1

func gems_right():
    current_page += 1
    update(curr_doll)

func gems_left():
    current_page -= 1
    update(curr_doll)

func update(doll):
    pages = Player.gems.size()/7
    if Player.gems.size() % 7 != 0:
        pages += 1

    if pages == 0:
        pages = 1

    $GemlistButtonLeft.disabled = false
    $GemlistButtonRight.disabled = false

    if current_page >= pages:
        current_page = pages
        $GemlistButtonRight.disabled = true
    if current_page <= 1:
        current_page = 1
        $GemlistButtonLeft.disabled = true

    for i in Util.doll_root.get_children():
        i.set_transparent()
    curr_doll = doll
    curr_doll.set_opaque()
    doll.reset()
    Util.annihilate_children($Gemlist)
    for i in Player.gems.slice((current_page-1)*7, current_page*7):
        var gembtn = gembtn_template.instantiate()
        gembtn.gem = i
        gembtn.pressed.connect(gem_button_pressed.bind(gembtn))
        $Gemlist.add_child(gembtn)

    $GemPageLabel.text = str(current_page) + " / " + str(pages)

    var iter = 0
    for i in doll.gem_board.unlocked_spaces:
        for j in i:
            if not j:
                $Board/Blockers.get_children()[iter].visible = true
            else:
                $Board/Blockers.get_children()[iter].visible = false
            iter += 1

    $TopLabel.text = doll.unitName
```


58

```
    display_inserted_gems()
    $DollDescriptor.upd(doll)

    $NotchesLabel.text = "Notches: " + str(Player.notches)

func display_inserted_gems():
    Util.annihilate_children($Board/InsertedGems)
    for i in curr_doll.gem_board.inserted_gems:
        var static_visualgem = visualgem_template.instantiate()
        static_visualgem.upd(i[0].shape, i[0].core_location, i[3])
        static_visualgem.modulate = i[0].color
        static_visualgem.modulate.a = 0.8
        $Board/InsertedGems.add_child(static_visualgem)
        var btn_idx = i[1]+i[2]*5
        static_visualgem.global_position = $Board/PlacementButtons.get_child(btn_idx).global_position

func gem_button_pressed(btn):
    if inserting_gem:
        cancel_gem_insertion()
    start_gem_insertion(btn.gem)
    $Board/PlacementButtons.get_child(12).grab_focus()

warp_mouse($Board/PlacementButtons.get_child(12).global_position+$Board/PlacementButtons.get_child(12).size/2)

func start_gem_insertion(gem):
    inserting_gem = true
    held_gem = gem
    visualgem = visualgem_template.instantiate()
    visualgem.upd(held_gem.shape, held_gem.core_location)
    visualgem.modulate = held_gem.color
    visualgem.modulate.a = 0.5
    add_child(visualgem)

func cancel_gem_insertion():
    inserting_gem = false
    held_gem = null
    if visualgem != null:
        remove_child(visualgem)
        visualgem.free()
        visualgem = null

func placement_focused(btn):
    if inserting_gem:
        visualgem.position = btn.global_position + visual_placement_offset
    else:
        var btn_x = btn.get_index()%5
        var btn_y = (btn.get_index()/5)%5
        var focused_gem = curr_doll.gem_board.get_gem_at_pos(btn_y, btn_x)
        if focused_gem != null:
            $GemDescriptor.upd_desc(focused_gem[0], focused_gem[1])

func placement_unfocused():
    if not inserting_gem:
        return

    await get_tree().process_frame
    var grid_focused = false
    for i in $Board/PlacementButtons.get_children():
        if i.has_focus():
            grid_focused = true
            break
```


59

```
    if not grid_focused:
        cancel_gem_insertion()

func placement_pressed(btn):
    var btn_x = btn.get_index()%5
    var btn_y = (btn.get_index()/5)%5
    if not inserting_gem:
        var clicked_gem = curr_doll.gem_board.get_gem_at_pos(btn_y, btn_x)
        if clicked_gem != null:
            clicked_gem = clicked_gem[0]
            curr_doll.gem_board.remove_gem(clicked_gem)
            Player.gems.append(clicked_gem)
            Player.sort_player_gems()
            update(curr_doll)
            return
        if not curr_doll.gem_board.unlocked_spaces[btn_y][btn_x] and Player.notches > 0:
            Player.notches -= 1
            curr_doll.gem_board.unlocked_spaces[btn_y][btn_x] = true
            update(curr_doll)
            return
    elif curr_doll.gem_board.check_gem_insertable(held_gem.shape, btn_y-1, btn_x-1):
        curr_doll.gem_board.insert_gem(held_gem, btn_x, btn_y)
        Player.gems.erase(held_gem)
        cancel_gem_insertion()
        update(curr_doll)

func _ready():
    for i in $Board/PlacementButtons.get_children():
        i.pressed.connect(placement_pressed.bind(i))
        i.focus_entered.connect(placement_focused.bind(i))
        i.focus_exited.connect(placement_unfocused)
    set_process(false)

func _process(_delta):
    if Input.is_action_just_pressed("cancel") and Util.active_input(self) and not inserting_gem:
        close_self()
    if Input.is_action_just_pressed("cancel") and Util.active_input(self) and inserting_gem:
        cancel_gem_insertion()

    if Input.is_action_just_pressed("shift") and Util.active_input(self) and inserting_gem:
        held_gem.rotate_shape()
        visualgem.upd(held_gem.shape, held_gem.core_location)

func load_next_doll():
    var idx = (curr_doll.get_index() + 1) % 3
    update(Util.doll_root.get_child(idx))

func load_previous_doll():
    var idx = (curr_doll.get_index() + 2) % 3
    update(Util.doll_root.get_child(idx))

func close_self():
    cancel_gem_insertion()
    get_parent().close_gemcraft()
```

FILE: \UI\GemDescriptor.gd

```
extends RichTextLabel

func bind_gem(node: Button):
    node.mouse_entered.connect(upd_desc.bind(node.gem))
    node.focus_entered.connect(upd_desc.bind(node.gem))
```


60

```
func upd_desc(gem, amp = false):
    text = "\nNormal effect:\n"
    text += gem.get_desc(false)
    text += "\n\nAmplified effect:\n"
    text += gem.get_desc(true)
```

FILE: \UI\GemlistGem.gd

```
extends GrabberButton

var gem: BaseGem

func _ready():
    super._ready()
    text = gem.gem_name
    $Display.modulate = gem.color
    var iter = 0
    for i in gem.shape:
        for j in i:
            if j:
                $Display.get_children()[iter].visible = true
            else:
                $Display.get_children()[iter].visible = false
            if Vector2(iter%3, iter/3) == gem.core_location:
                $Core.global_position = $Display.get_children()[iter].global_position + $Display.get_children()[iter].size/2
            iter += 1
```

FILE: \UI\GrabberButton.gd

```
extends Button
class_name GrabberButton

func _ready():
    mouse_entered.connect(grab_focus)
```

FILE: \UI\HPLabel.gd

```
extends Label

@export var unit: UnitBase

func _ready():
    unit.cHP_changed.connect(upd)
    upd()

func upd(_old = 0, _new = 0):
    text = str(unit.cHP) + " / "  + str(unit.get_stat_val(UnitBase.HP))
```

FILE: \UI\Pause.gd

```
extends Control

func _ready():
    set_process(false)

func _process(_delta):
    if Input.is_action_just_pressed("cancel"):
        if Util.active_input(self):
```


61

```
        close_self()

func close_self():
    await get_tree().process_frame
    Util.player_interface.close_pause()

func open_gemcraft():
    set_process(false)
    $PauseMenu.visible = false
    $GemcraftScreen.visible = true
    Util.active_input_object = $GemcraftScreen
    $GemcraftScreen.set_process(true)
    $GemcraftScreen.update(Util.doll_root.get_child(0))
    $GemcraftScreen/ButtonLeft.grab_focus()

func close_gemcraft():
    set_process(true)
    $PauseMenu.visible = true
    $GemcraftScreen.visible = false
    Util.active_input_object = self
    $GemcraftScreen.set_process(false)
    $PauseMenu/GemcraftBtn.grab_focus()
    for i in Util.doll_root.get_children():
        i.get_node("Bust").modulate = Color(1, 1, 1, 0.5)
```

FILE: \UI\TurnOrderDisplay.gd

```
extends HBoxContainer

var combat_scene

func _ready():
    Util.node_references_reloaded.connect(upd)

func upd():
    combat_scene = Util.combat_scene
    combat_scene.connect("turn_order_changed", update_order)

func update_order():
    for i in get_children():
        remove_child(i)
        i.free()

    for i in combat_scene.turn_order:
        var lab = preload("res://Scenes/2D/turn_order_panel.tscn").instantiate()
        lab.upd(i[1].unitName, i[0])
        if i[1] in Util.doll_root.get_children():
            lab.modulate = Color.GREEN
        elif i[1] in Util.combat_scene.get_enemies():
            lab.modulate = Color.RED
        add_child(lab)
```

FILE: \UI\TurnOrderPanel.gd

```
extends Panel

func upd(name, delay):
    $UnitName.text = name
    $Delay.text = str(delay)

func _process(_delta):
```


62

```
    if Input.is_action_pressed("shift"):
        $Delay.visible = true
    else:
        $Delay.visible = false
```

FILE: \UI\unit_plaque.gd

```
extends Control

@export var creature: UnitBase

# Called when the node enters the scene tree for the first time.
func _ready():
    if creature == null:
        creature = get_parent()
    creature.cHP_changed.connect(on_cHP_changed)
    creature.cMP_changed.connect(on_cMP_changed)
    creature.barrier_changed.connect(on_barrier_changed)
    creature.burn_changed.connect(on_burn_changed)
    creature.poison_changed.connect(on_poison_changed)
    creature.critical_changed.connect(on_critical_changed)
    creature.reseted.connect(on_reset)
    on_reset()

func on_cHP_changed(_old, new):
    $HPBar.value = new
    $HPLabel.text = str(new)

func on_cMP_changed(_old, new):
    $MPBar.visible = not new == 0
    $MPLabel.visible = not new == 0

    $MPBar.value = new
    $MPLabel.text = str(new)

func on_barrier_changed(_old, new):
    $BarrierBar.visible = not new == 0
    $BarrierLabel.visible = not new == 0

    $BarrierBar.value = new
    $BarrierLabel.text = str(new)

func on_burn_changed(_old, new):
    $BurnLabel.visible = not new == 0

    $BurnLabel.text = str(new)

func on_poison_changed(_old, new):
    $PoisonLabel.visible = not new == 0

    $PoisonLabel.text = str(new)

func on_critical_changed(_old, new):
    $CritIndicator.visible = new

func on_reset():
    $HPBar.max_value = creature.get_stat_val(UnitBase.HP)
    $HPBar.value = creature.cHP
    $BarrierBar.max_value = creature.get_stat_val(UnitBase.HP)
    $MPBar.max_value = creature.get_stat_val(UnitBase.MP)
    $MPBar.value = creature.cMP
    $UnitName.text = creature.unitName
```


63

FILE: \UI\VisualGem.gd

```
extends Control

func upd(shape, core_pos, linked = false):
    var iter = 0
    for i in shape:
        for j in i:
            if j:
                get_children()[iter].visible = true
            else:
                get_children()[iter].visible = false
            if Vector2(iter%3, iter/3) == core_pos:
                $Core.position = get_children()[iter].position
            iter += 1

    if linked:
        $Core.self_modulate = Color(1, 1, 1, 0.7)
    else:
        $Core.self_modulate = Color(0.1, 0.1, 0.1, 0.5)
```


64