

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний авіаційний університет
Факультет Комп'ютерних Наук та Технологій
Кафедра комп'ютерних інформаційних технологій

Дизайн-Документ

з дисципліни “Кросплатформне програмування”

Виконали :

Студенти групи ТП-215Б ФКНТ

Пархомчук Дмитро Олександрович

Матвійчук Софія Юріївна

Київ – 2024

Зміст

Вступ	4
Назва програмного продукту :	4
Коротка характеристика області застосування :	4
Стислий опис гри :	4
Позиціонування гри :	4
• Унікальність	4
• Цільова аудиторія	5
• Ключові особливості	5
• Тематика та стиль	5
Основні механіки та сетинг :	6
Наратив	7
Історія місця і події :	7
Інформація про персонажів :	8
Інформація про можливі знахідки :	9
Геймплей	10
Механіки управління та опис персонажів :	10
▪ Головний герой :	10
▪ Окультисти :	11
▪ Уродженці Пітьми :	12
Механіки прокачки та знахідок :	15
• Уламки темних душ :	15
• Частинки світлої енергії :	15
• Додаткові предмети :	16
▪ Артефакти :	16
▪ Вогнище древніх воїнів :	17
Бойова система :	18
▪ Гравець :	18
▪ Уродженці пітьми :	18
▪ Окультисти :	18
Інтерфейс	19
• Персональні комп'ютери :	19
• Мобільні пристрої (Android) :	20

Розробка.....	22
Інструменти :.....	22
▪ <i>Ігровий движок :</i>	22
▪ <i>Редактори :</i>	22
▪ <i>Управління проектом :</i>	22
▪ <i>Схеми та діаграми :</i>	23
Карточки персонажів :.....	24
Карточки прокачки :.....	26
Процес розробки :.....	29
• <i>Анімація персонажів :</i>	29
• <i>Скрипти :</i>	34
• <i>Graphics :</i>	55

Вступ

Даний дизайн-документ створювався з наміром описати цілком і повністю план та структуру створення кросплатформенного додатка “Darkness Survival”, зі всіма деталями механік гри, концептів та інших важливих деталей реалізації. Даний додаток має своєю метою розважити користувача простим, але цікавим і динамічним ігровим процесом.

Назва програмного продукту :

Кросплатформенна гра “Darkness Survival”.

Коротка характеристика області застосування :

Розвага користувачів, орієнтована на короткі ігрові сеанси, які допомагають швидко та цікаво витратити час з мінімальним смисловим навантаженням.

Стислий опис гри :

“Darkness Survival” – це top-down гра, в якій гравцю потрібно буде протриматись як можна довше під постійним напливом унікальних монстрів. В цьому гравцю допоможуть різноманітні навички, які можна буде набувати та удосконалювати, підвищуючи рівень героя, знищуючи цілі орди монстрів, що приходять з півночі.

Позиціонування гри :

- **Унікальність.**

“Darkness Survival” відрізняється від своїх конкурентів захоплюючою атмосферою, що створює непередбачувану та містичну обстановку.

Кожна частина карти містить в собі загадкові події, які відбулися до

прибуття протагоніста, який так само як і гравець, може лише здогадуватись, що тут відбулось раніше.

- **Цільова аудиторія.**

- *За віком :*

- Гра орієнтується на чоловіків та жінок віком від 12 до 35 років.

- *За платоспроможністю:*

- Низька платоспроможність або її відсутність.

- *За інтересами :*

- Перш за все, гра орієнтована на користувачів яким подобається високий рівень складності в іграх та які бажають швидко, з цікавістю провести свій час.

- По-друге, “Darkness Survival” може зацікавити користувачів, яким подобаються ігри з темною графікою, містичною та загадковою атмосферою.

- Також гра буде до вподоби користувачам, які насолоджуються системами прокачки героя, з можливістю підвищувати рівень та особисто обирати навички якими буде володіти герой.

- **Ключові особливості.**

- *Темна атмосфера та графіка.*

- *Top-Down геймплей.*

- *Виживання та прокачка.*

- *Унікальний арт-дизайн.*

- **Тематика та стиль.**

- “Darkness Survival” поєднує в собі стилі фентезі, хоррора та деякі елементи кіберпанку. Адже протагоніст, воїн з майбутнього, портається в містичний, стародавній світ, де зіштовхується зі створіннями породженими пітьмою та збираючи уламки їхніх темних душ, набуває нових, невідомих раніше, здібностей, або удосконалює існуючі.

Основні механіки та сетинг :

Основні механіки гри тісно пов'язані з містичним сетингом, в стилі *dark fantasy*. Гравець в ході ігрового процесу буде досліджувати нові локації старовинного світу, знаходити різноманітні уламки темних душ та удосконалюватись завдяки ним, щоб дати відсіч нескінченним створінням, які завжди знають де він знаходиться.

Коротко про основні механіки геймплею :

- *Можливість досліджувати зациклений простір.*
- *Постійний наступ ворогів, які з часом стають все сильнішими і приходять у більшій кількості.*
- *Система прокачки героя з різноманітними здібностями за рахунок збирання уламків душ.*
- *Збирання додаткових предметів які можуть відновити запас здоров'я героя або дати інші короточасні бонуси.*
- *Збирання артефактів, які з'являються в світі та можуть мати як позитивні, так і негативні ефекти на героя.*
- *Взаємодія героя та ворогів з різними перешкодами, по типу стін, дерев та інших об'єктів, через які неможливо проходити наскрізь.*
- *Різнноманітні дебафи на ворогів від деяких здібностей або знайдених у світі бонусів.*
- *Можливість знаходити під час гри частинки світлої енергії, яка зберігається після завершення ігрового сеансу, та дає можливість відкрити нові кастомізації персонажа.*
- *Використання вогнища древніх воїнів, яке дає випадкову кількість світлої енергії та прокачку випадкової здібності.*

Наратив

Історія місця і події :

Головний герой, з кодовим ім'ям Неро, це найманець з далекого майбутнього, основною задачею якого було подорожувати між світами та збирати дані про цивілізовані раси, з метою використання їх технологій та ресурсів у своєму світі.

Світ з якого походить Неро, процвітає вже багато років та досяг великих вершин завдяки розвиненим технологіям, які власне і дозволяють йому перетинати світовий бар'єр.

Під час останньої подорожі, Неро, в ході своєї розвідки, натрапив на слабо розвинений, але при цьому дивний культ, який, з його спостережень, поклонявся химерному предмету та самі окултисти приносили себе в жертву, окроплюючи його власною кров'ю. Від цього дивного артефакту поширювались настільки сильні енергетичні хвилі, що навіть самі потужні пристрої найманця не могли їх виміряти і давали збій.

Коли герой викрав невідомий артефакт, думаючи, що використання його безмежної енергії зробить новий прорив у розвитку його цивілізації, він використав мітку повернення, але по невідомій причині вона почала дивно поводитись і в результаті він потрапив в інший світ, подібних якому, найманець не зустрічав ніколи.

З цього власне і розпочинається подорож Неро, під управлінням гравця, в цьому загадковому та містичному світі, відкинутому від законів простору і часу та який по невідомій причині має ефект зацикленості. Все що лишається герою - нескінченно бродити по одних і тих самих місцях, сотні, а то і тисячі раз, відбиваючись від безмежних орд монстрів, які приходять з півми, щоб позбутись незваної живої душі та відібрати артефакт.

В цій, на перший погляд, безвихідній ситуації, протагоніст, вбиваючи армії нечисті, відкрив собі нові магічні здібності, які не зустрічались в жодному

зі світів в яких він вже побував. Також на його шляху будуть зустрічатись невідомі артефакти, які допоможуть в цій боротьбі або навпаки послугують перешкодою, створеною темними силами. Саме завдяки набуванню нових здібностей та удосконаленню існуючих, в героя з'являється шанс на виживання в цьому божевільному місці.

Чим саме закінчиться історія Неро? Можливо, смертю. Можливо, вічним блуканням в цій петлі і боротьбою з потворами п'їтьми. Можливо все ж таки знайдеться якась магія або щось, що допоможе найманцю повернутись у свій світ. Кінець цієї історії нікому ще не відомий, і лише гравець, разом з Неро, пройдуть через це та дізнаються що їх чекає в кінці шляху.

Інформація про персонажів :

Неро (головний герой) : Найманець із далекого майбутнього, вправний у володінні мечами, коп'ями та іншими видами зброї. Він має високу концентрацію, витривалість та різноманітні бойові навички, а згодом, збираючи уламки темних душ, набуває абсолютно нових, містичних здібностей.

Окультисти : Послідовники та жертвопринесенці культу, які віддано служать химерному артефакту. Після смерті, їхні душі також переносяться в містичний, зациклений світ та спотворюються, поповнюючи собою армію п'їтьми. Після переродження мають магічні здібності, але їхні слабкі тіла роблять їх вразливими перед Неро. Невідомо які ще секрети приховані в їх існуванні у цьому божевільному світі.

Уродженці П'їтьми : Створіння, що приходять з п'їтьми, щоб завдати шкоди та відібрати життя Неро. Їх різноманітність та чисельність ускладнюють боротьбу, але кожен з них має свої слабкості.

Інформація про можливі знахідки :

Уламки темних душ : Це особливий вид енергії, який властивий лише божевільному світу в який потрапив Неро. За рахунок цих уламків, герой може здобувати надприродні сили та навички, або удосконалювати наявні. Описані уламки можна здобути лише зі знищених потвор пітьми, оскільки вони являються частинами їхніх душ.

Частинки світлої енергії : Можуть бути знайдені в ігровому світі, зберігаються після завершення ігрового сеансу, та дають можливість за певну кількість відкрити нові кастомізації персонажа.

Додаткові предмети : До таких предметів відносяться капсули які відновлюють частину здоров'я героя, древні рунічні символи які збільшують потужність всіх уламків темних душ (окрім наявних), або навпаки, лише збільшують кількість наявних не впливаючи на решту.

Артефакти : Таємничі об'єкти, створені світом, які можуть вносити нові елементи в ігровий процес, які допоможуть або завадять Неро у боротьбі за життя.

Вогнище древніх воїнів : Це вогнище, полум'я якого теж піддається циклу, слугувало як священний об'єкт воїнів, які колись захищали цей світ, допоки сили пітьми не зломали їх та не спотворили їхні душі. Залишившись від тієї давнини, воно є джерелом таємничої енергії, яку вловлюють пристрої Неро та вказують напрям до її джерела. Використовуючи його енергію та викрадений артефакт, найманець отримує частинки світлої енергії та удосконалення здібностей. Вогнище з'являється кожного разу коли Неро вбиває командуючого армією уродженців пітьми.

Геймплей

Механіки управління та опис персонажів :

▪ *Головний герой :*

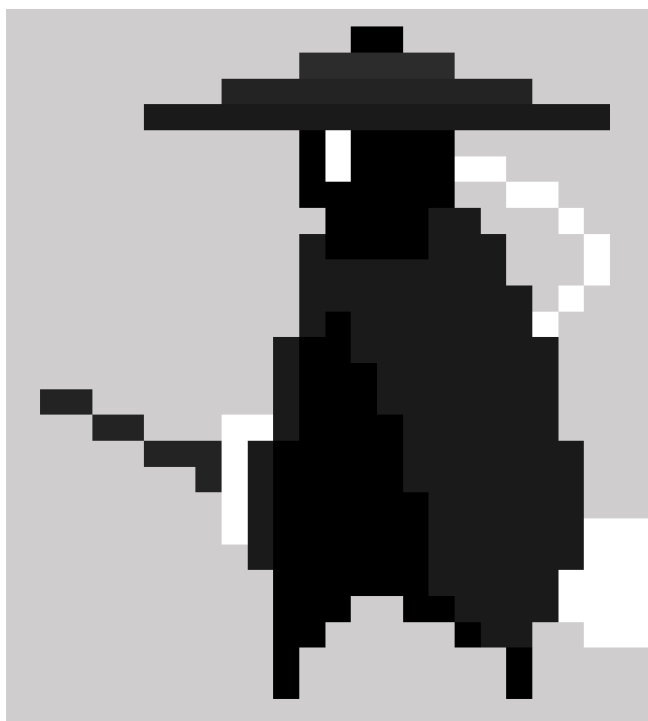
Гравцю надається керування розкритим в історії найманцем – Неро (*ілюстрація.1*).

Граючи на персональному комп'ютері або ноутбукі, керування Неро здійснюється чотирма клавішами (за замовчуванням WASD або стрілочки), відповідальними за переміщення героя в необхідному гравцю напрямку та окрема кнопка (за замовчуванням лкм. - ліва кнопка миші) для здійснення атаки основною зброєю найманця.

На телефоні переміщення героя реалізовано з використанням віртуального джойстика (в лівій частині екрана за замовчуванням) , яким легко керувати переміщення персонажа, а також окрема кнопка для атаки основною зброєю (в правій частині екрана за замовчуванням).

Назначену клавішу атаки на комп'ютерній версії можна в будь-який момент змінити через налаштування. У мобільній версії існує можливість зробити джойстик та кнопку невидимими або поміняти їх місцями для зручності. Крім того, доступна можливість вибрати готовий шаблон для положення елементів керування та їх розмірів із запропонованих варіантів.

Інші здібності героя, які він буде набувати в ході своєї прокачки, регулярно застосовуються автоматизовано. Деякі з них направлені в ту сторону в яку дивиться герой, якась частина автоматично обирає ціллю ближнього ворога, а певні з них можуть відтворювати атаки у випадковому напрямку або по області певного радіуса. Кожна здібність має свій опис та умови застосування, тому гравцю легко буде орієнтуватись як саме працює кожна з них.



Ілюстрація.1 – найманець Неро

▪ **Окультисти :**

На відміну від уродженців пітьми, Окультисти (*ілюстрація.2*) переродились в цьому світі зі спотвореними душами. Вони вирізняються тим, що мають особливі магічні здібності, які були даровані артефактом. Окрім цього, їхні душі були піддані ефекту зациклення, що дарує їм безсмертя у цьому світі. Їм властиві далекобійні атаки, які можуть нести в собі не тільки фізичну шкоду, але й певні побічні ефекти, які накладаються на протагоніста та діють короткий період часу, послаблюючи його.

До основних механік входять : постійне переслідування Неро та автоматичне відтворення далекобійних атак, які мають певні ефекти на головного героя в залежності від здібностей Окультиста. Коли відстань до найманця лишається невеликою, то вони зупиняються та продовжують проводити свої атаки на більш безпечній для себе відстані.

Скільки б разів гравець не знищував їх, з часом вони будуть відновлюватись та повертатись із п'їтьми, щоб вбити того хто викрав їх артефакт.



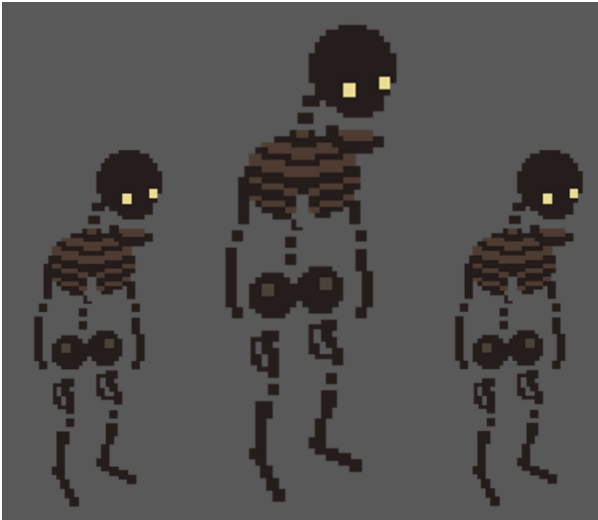
Ілюстрація.2 – Окултисти

■ ***Уродженці П'їтьми :***

Безмежна орда нечисті, яка знищує все живе на своєму шляху. Вони завжди відчують присутність живих, тому будуть безустанно переслідувати Неро в цьому світі, допоки не зломають його і не спотворять його душу, як робили з всіма хто були задовго до нього.

З кожним разом їх приходить все більше і з кожною хвилиною що протагоніст проживає в цьому світі, тьма породжує сильніших потвор, щоб дістатись до його душі.

Кожен різновид нечисті має свої унікальні характеристики. Найслабші з них - Висохші душі (*ілюстрація.3*). Вони досить повільні та слабкі, однак далять ворогів своєю кількістю. Якщо вони не справляються, то приходять Симбіоти (*ілюстрація.4*), які утворились внаслідок мутації спотворених Висохших. Завдяки симбіозу, їхні силові показники вирости, вони більш стійкі та небезпечні чим їхні попередники.



Ілюстрація.3 – Висохші душі



Ілюстрація.4 - Симбіот

Більш швидкі та небезпечні Прокляті духи (*ілюстрація.5*), їх важко знищити за рахунок їхньої швидкості та регенерації. Однак їхня слабкість в тому що вони досить вразливі до сильних атак і не встигають відновитись. Є також три різновиди : звичайні, більш швидкі, з прискореною регенерацією.



Ілюстрація.5 – Прокляті духи

Окремим могутнім підвидом армії нежиті є Древні воїни (*ілюстрація.6*), які колись зломались під натиском сил темряви та поповнили їхню армію піддавшись мутаціям. Вони мають середню швидкість, високий спротив до будь-яких магічних атак і більш смертоносні порівняно з іншими уродженцями півми, тому краще тримати їх на відстані. Деякі з них мають високий спротив,

але вони менш небезпечні в бою, інші ж навпаки більш смертоносні, але при цьому у них слабший захист. Також є і ті хто має певний баланс спротиву та сили. Їх можна легко розрізнати за зовнішнім виглядом.



Ілюстрація.6 – Древні воїни

- **Примітка :** Також є Командуючі армією нечисті, вони відрізняються розмірами, силою та надто високою витривалістю. Вони приходять лише через деякий час, коли звичайна нежить не справляється.

Механіки прокачки та знахідок :

- **Уламки темних душ :**

(Ілюстрація.7)

Ці уламки є основним предметом прокачки Неро. За допомогою цих уламків, найманець підвищує свій рівень та відкриває нові здібності або прокачує існуючі. Самі уламки можуть бути різних видів та містити в собі різну кількість темної енергії. Їх можна получити лише знищуючи ворогів.

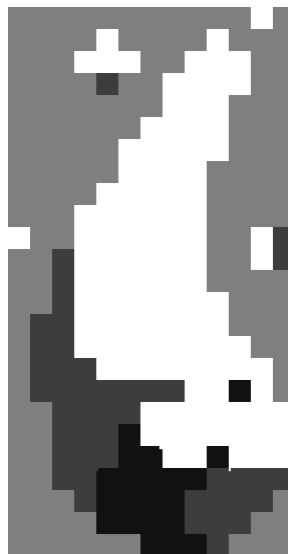


Ілюстрація.7 – Уламки темних душ

- **Частинки світлої енергії :**

(Ілюстрація.8)

Ресурс, який утворюється з деякою ймовірністю у випадковому місці світу. Може бути використаний в меню гри для розблокування нових кастомізацій та бонусів героя.

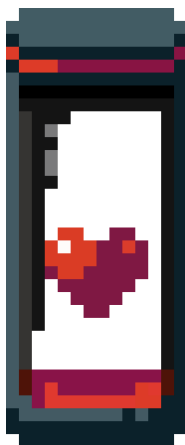


Ілюстрація.8 – Частинка світлої енергії

- **Додаткові предмети :**

(Ілюстрація.9)

До таких предметів відносяться капсули життєвої енергії, які відновлюють певну кількість здоров'я. Також в світі можуть з'являтися древні рунічні символи, які впливають на уламки темних душ : збільшують потужність всіх уламків, або лише збільшують кількість наявних не впливаючи на решту. Ці предмети генеруються у світі випадково з певною ймовірністю.



Ілюстрація.9 – Капсула життєвої енергії

- **Артефакти :**

(Ілюстрація.10)

Артефакти можуть значно допомогти гравцю, наділяючи Неро додатковою силою та бонусами, але інколи це небезпечно. Деякі з артефактів які генеруються у світі мають негативні ефекти, які можуть лише ускладнити виживання в цьому світі, тому наближатися до них потрібно лише на свій страх і ризик.



Ілюстрація.10 – Артефакт

▪ **Вогнище древніх воїнів :**

(Ілюстрація.11)

Дає частинки світлої енергії та удосконалення здібностей. З'являється після знищення Командуючих армією уродженців півми. Випадкового удосконалює одну з освоєних здібностей.



Ілюстрація.11 – Вогнище древніх воїнів

- **Примітка :** Для того щоб використати якусь з описаних знахідок, достатньо просто підійти до них на достатню відстань, тоді герой автоматично їх підніме.

Бойова система :

▪ Гравець :

Має можливість використовувати одну керовану атаку, основною зброєю Неро. Решта здібностей, які набуваються в ігровому процесі, використовуються автоматично. Подробиці про здібності та інше описано на карточках, які знаходяться у розділі ***Розробка -> Карточки прокачки та Розробка -> Карточки персонажів.***

▪ Уродженці нітьми :

Завжди переслідують гравця та наносять урон лише коли приблизились майже впритул до Неро. Їхні атаки відрізняються кількістю одиниць урона за секунду, детальніше описано у розділі ***Розробка -> Карточки персонажів.***

▪ Окультисти :

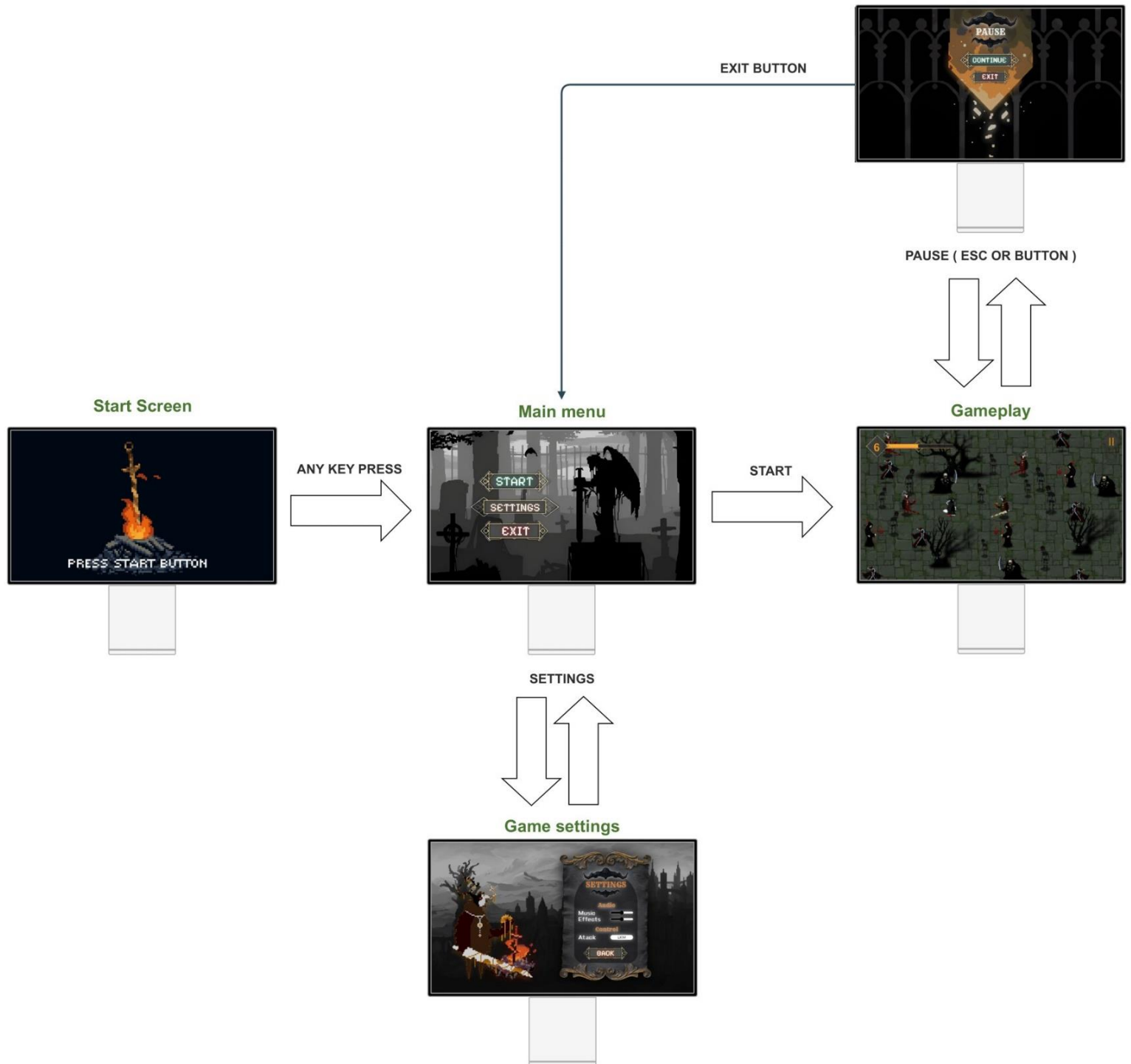
Переслідують героя та на певній відстані від нього зупиняються, атакуючи далекобійними атаками. Урон залежить від виду окультиста, детальніше описано у розділі ***Розробка -> Карточки персонажів.***

Інтерфейс

- Персональні комп'ютери :

Darkness Survival - PC

Розташування елементів інтерфейсу

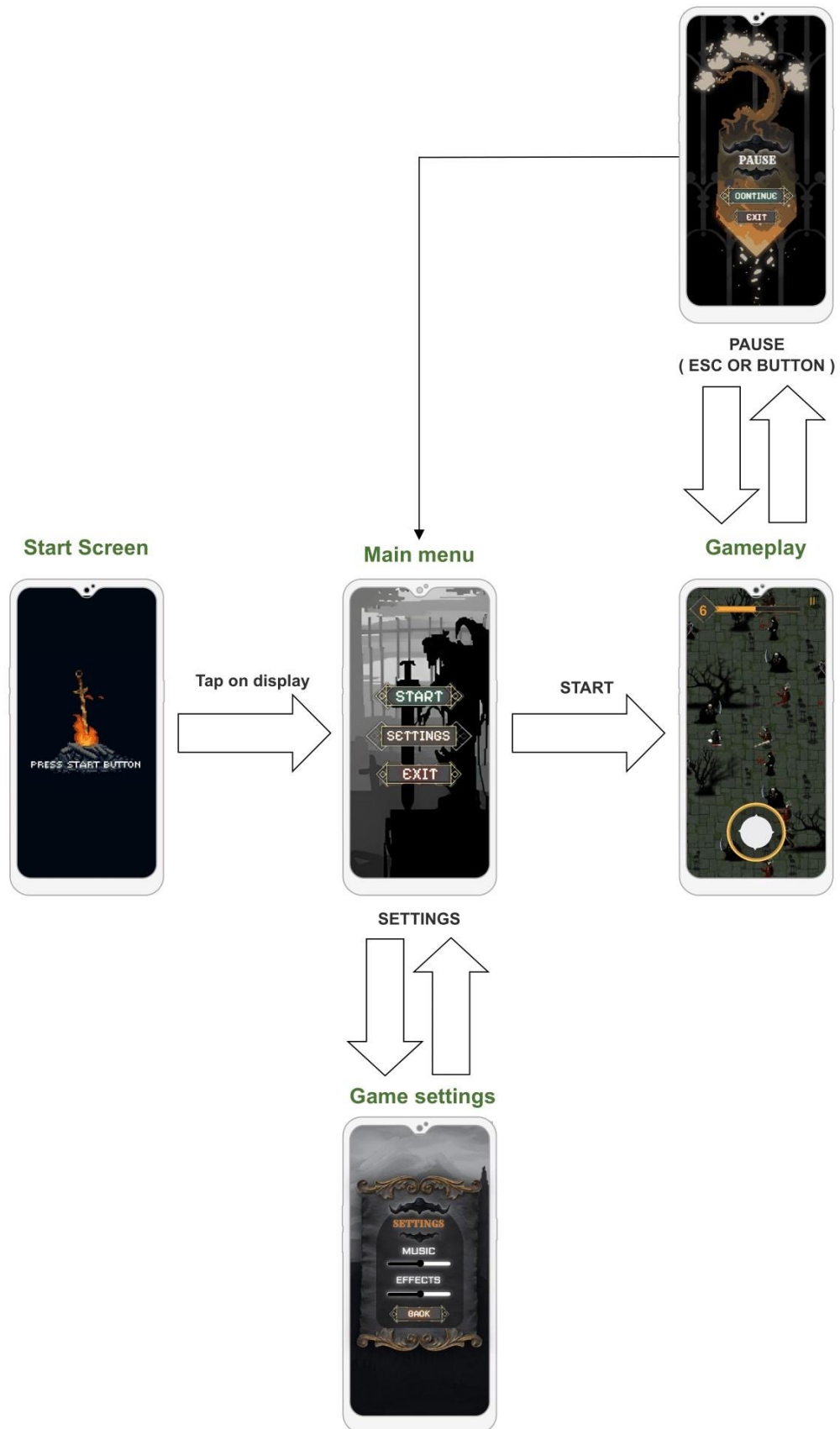


Посилання на дошку Moqups для детального перегляду : <http://surl.li/qtmat>.

- Мобільні пристрої (Android) :

Darkness Survival - Android

Розташування елементів інтерфейсу



Посилання на дошку Moqups для детального перегляду : <http://surl.li/qtqeu>.

Примітка : Дані схеми описують зовнішній дизайн інтерфейсу на PC та Android, можливості взаємодії користувача з його елементами та взаємозв'язки між окремими вікнами. В майбутньому буде додана схема для IOS та сумісність гри з даною операційною системою.

Розробка

Інструменти :

▪ Ігровий движок :

Розробка гри базується на відомому ігровому движку ***Unity***. Його зручність та потужність дозволяє швидше та якісніше реалізувати механіки описаної гри. Всі скрипти будуть реалізовані мовою програмування ***C#***, яка підтримує об'єктно орієнтоване програмування та його основні принципи : інкапсуляцію, поліморфізм та наслідування.

▪ Редактори :

Для реалізації різноманітних елементів інтерфейсу, редагування елементів світу та інших зображень, використовується ***Adobe Photoshop***.

Концепт-арти, персонажі та їхні анімації, різноманітні ефекти та здібності, всі інтерактивні елементи які зустрічаються в ході геймплею - реалізуються в піксельному редакторі ***Aseprite***.

Текстовий редактор який використовується для оформлення Дизайн-Документу та інших можливих додатків\документів - ***Microsoft Word***.

Відео-редактор ***Adobe After Effects 2019*** використовується для редагування аудіо-ефектів та саундтрека гри. Також в ньому корегуються деякі анімації, накладаються фільтри на зображення\відео, для кращої передачі атмосфери.

▪ Управління проектом :

Для зручного та ефективного управління проектом використовується ***Trello***, він надає зручну та багатофункціональну дошку. З її допомогою, можна швидко планувати задачі, ставити їм потрібні мітки та закріплювати за ними

відповідального виконавця. Вона значно допомагає організувати роботу у команді та відслідковувати статус виконання задач.

▪ ***Схеми та діаграми :***

Для реалізації блок-схем, ілюстрації взаємодії інтерфейсів, різноманітних діаграм які будуть створюватись в ході виконання проекту, будуть використовуватись онлайн редактори ***Lucidchart*** та ***Moqups***.

▪ ***Ідеї та концепти :***

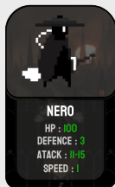

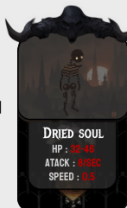




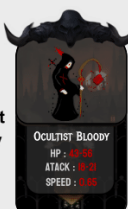



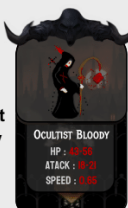



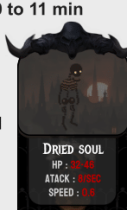


Для пошуку ідей та початкових концептів використовується онлайн платформа ***Pinterest*** та інші інтернет ресурси. Деякі елементи оточення напряду імпортуються в проект, персонажі та різноманітні анімації лише беруться в якості ідей та значно перероблюються (*Ілюстрація.12*).





Ілюстрація.12 – Від концепту до фінального результату

Карточки персонажів :

Створені для опису характеристик персонажів на певних хвилинах гри.

Hero				
Nero	<div><p>NERO HP : 600 DEFENCE : 9 ATTACK : 600 SPEED : 1</p></div>			
To 1 min				
Dried Soul	<div><p>DRIED SOUL HP : 300 ATTACK : 1000 SPEED : 0.5</p></div>			
From 1 to 3 min				
Dried Soul	Simbiot			
<div><p>DRIED SOUL HP : 300 ATTACK : 1000 SPEED : 0.5</p></div>	<div><p>SIMBIOT HP : 600 ATTACK : 1000 SPEED : 0.4</p></div>			
From 3 to 5 min				
Dried Soul	Simbiot	Ocultist Toxin	Ocultist Bloody	
<div><p>DRIED SOUL HP : 300 ATTACK : 1000 SPEED : 0.5</p></div>	<div><p>SIMBIOT HP : 600 ATTACK : 1000 SPEED : 0.4</p></div>	<div><p>OCULTIST TOXIN HP : 600 ATTACK : 1000 (3-5) SPEED : 0.5</p></div>	<div><p>OCULTIST BLOODY HP : 600 ATTACK : 600 SPEED : 0.55</p></div>	
From 5 to 7 min				
Damned Spirit Bloody	Damned Spirit Toxin	Ocultist Toxin	Ocultist Bloody	
<div><p>DAMNED SPIRIT BLOODY HP : 300 (3) ATTACK : 1000 SPEED : 0.7</p></div>	<div><p>DAMNED SPIRIT TOXIN HP : 300 (3) ATTACK : 600 (1) 1000 SPEED : 1</p></div>	<div><p>OCULTIST TOXIN HP : 600 ATTACK : 1000 (3-5) SPEED : 0.5</p></div>	<div><p>OCULTIST BLOODY HP : 600 ATTACK : 600 SPEED : 0.55</p></div>	
From 7 to 9 min				
Damned Spirit Bloody	Damned Spirit Toxin	Damned Spirit Golden		
<div><p>DAMNED SPIRIT BLOODY HP : 300 (3) ATTACK : 1000 SPEED : 0.7</p></div>	<div><p>DAMNED SPIRIT TOXIN HP : 300 (3) ATTACK : 600 (1) 1000 SPEED : 1</p></div>	<div><p>DAMNED SPIRIT GOLDEN HP : 600 ATTACK : 600 SPEED : 0.6</p></div>		
From 9 to 11 min				
Dried Soul	Simbiot	Damned Spirit Golden		
<div><p>DRIED SOUL HP : 300 ATTACK : 1000 SPEED : 0.5</p></div>	<div><p>SIMBIOT HP : 600 ATTACK : 1000 SPEED : 0.4</p></div>	<div><p>DAMNED SPIRIT GOLDEN HP : 600 ATTACK : 600 SPEED : 0.6</p></div>		

Посилання на дошку Moqurs для детального перегляду : <http://surl.li/qtqih>.

From 11 to 13 min				
Ancient Warrior	Simbiot	Ocultist Toxin		
				
<hr/>				
From 13 to 15 min				
Ancient Warrior Bloody	Simbiot	Ocultist Bloody	Damned Spirit Bloody	
				
<hr/>				
From 15 to 17 min				
Ancient Warrior Golden	Ancient Warrior	Damned Spirit Golden	Damned Spirit Toxin	
				
<hr/>				
From 17 to 19 min				
Ancient Warrior Golden	Ancient Warrior	Ancient Warrior Bloody	Ocultist Bloody	
				
<hr/>				
From 19 to 20 min - Final				
Ancient Warrior Golden	Ancient Warrior Bloody	Damned Spirit Golden	Ocultist Bloody	
				

Посилання на дошку Moqurs для детального перегляду : <http://surl.li/qtqmb>.

Карточки прокачки :

Дані карточки створені для опису всіх здібностей та їх характеристик на певних рівнях прокачки.

Demonic Aura						
Soul Magnet						
Needle Step						
Bloody Peaks						
Demonic Spikes						

Посилання на дошку Moqurs для детального перегляду : <http://surl.li/qwozs>.

Cyber Heart	 CYBER HEART LVL: 1 → 2 / 7 100% → 100% MAX HP	 CYBER HEART LVL: 2 → 3 / 7 100% → 120% MAX HP	 CYBER HEART LVL: 3 → 4 / 7 120% → 130% MAX HP	 CYBER HEART LVL: 4 → 5 / 7 130% → 145% MAX HP	 CYBER HEART LVL: 5 → 6 / 7 145% → 160% MAX HP	 CYBER HEART LVL: 6 → 7 / 7 160% → 180% MAX HP + REGENERATION 160EC
	 CYBER SWORD LVL: 1 → 2 / 7 100% → 100% DAMAGE	 CYBER SWORD LVL: 2 → 3 / 7 100% → 125% DAMAGE	 CYBER SWORD LVL: 3 → 4 / 7 125% → 140% DAMAGE	 CYBER SWORD LVL: 4 → 5 / 7 140% → 160% DAMAGE	 CYBER SWORD LVL: 5 → 6 / 7 160% → 185% DAMAGE	 CYBER SWORD LVL: 6 → 7 / 7 185% → 200% DAMAGE + LIGHT AURA
	 CYBER ARMOR LVL: 1 → 2 / 7 100% → 100% DEFENCE	 CYBER ARMOR LVL: 2 → 3 / 7 100% → 117.5% DEFENCE	 CYBER ARMOR LVL: 3 → 4 / 7 117.5% → 125% DEFENCE	 CYBER ARMOR LVL: 4 → 5 / 7 125% → 130% DEFENCE	 CYBER ARMOR LVL: 5 → 6 / 7 130% → 137.5% DEFENCE	 CYBER ARMOR LVL: 6 → 7 / 7 137.5% → 160% DEFENCE + ENERGY AURA
	 CYBER WALK LVL: 1 → 2 / 7 100% → 101.5% SPEED	 CYBER WALK LVL: 2 → 3 / 7 101.5% → 103% SPEED	 CYBER WALK LVL: 3 → 4 / 7 103% → 104.5% SPEED	 CYBER WALK LVL: 4 → 5 / 7 104.5% → 106% SPEED	 CYBER WALK LVL: 5 → 6 / 7 106% → 107.5% SPEED	 CYBER WALK LVL: 6 → 7 / 7 107.5% → 108% SPEED + LIGHT AURA
	 SHARP BLADE LVL: 0 → 1 / 6 + EXTRA SHARP DPS 1.5% + ENERGY AURA	 SHARP BLADE LVL: 1 → 2 / 6 9-12 → 10 DAMAGE	 SHARP BLADE LVL: 2 → 3 / 6 11-13 → 11 DAMAGE	 SHARP BLADE LVL: 3 → 4 / 6 13-15 → 12 DAMAGE	 SHARP BLADE LVL: 4 → 5 / 6 15-17 → 13 DAMAGE	 SHARP BLADE LVL: 5 → 6 / 6 17-19 → 14 DAMAGE + ENERGY AURA

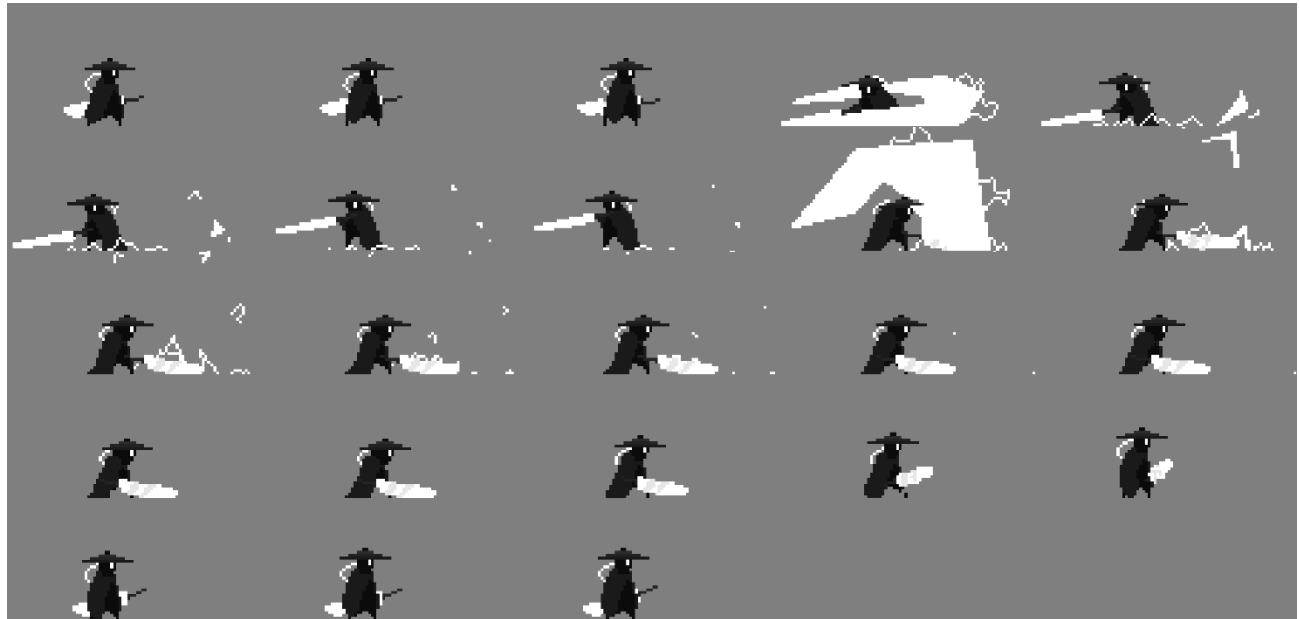
Shadow Weakness						
Deadly Toxin						
Darkness Aura						
Light Explosion						
Demonic Absorption						

Посилання на дошку Moqurs для детального перегляду : <http://surl.li/qwozm>.

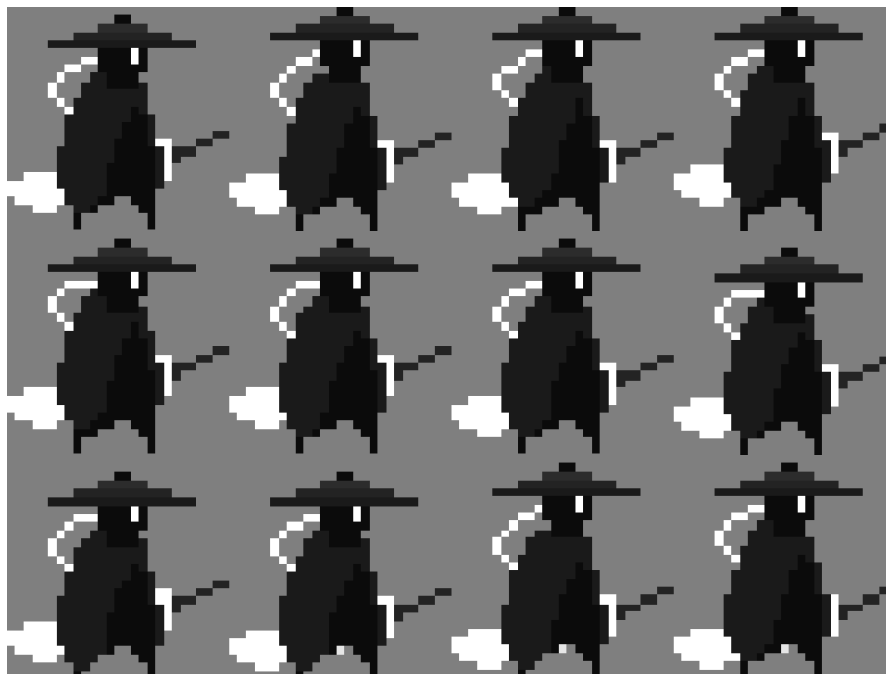
Примітка : Характеристики даних карточок можуть бути змінені в разі порушення балансу ігрового процесу.

Процес розробки :

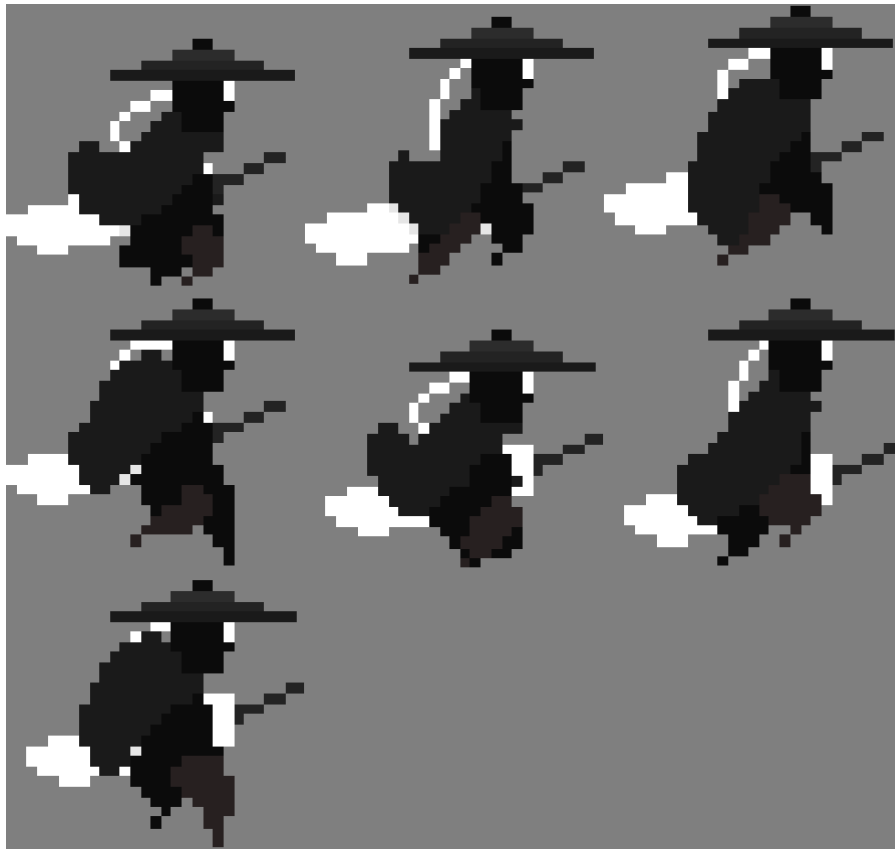
- Анімація персонажів :



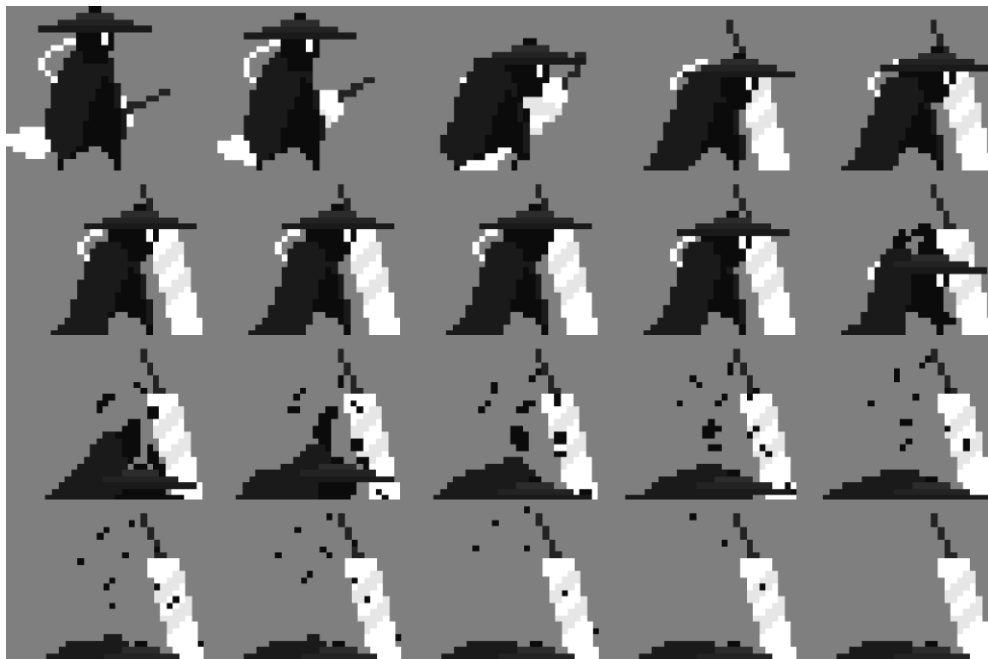
Attack animation



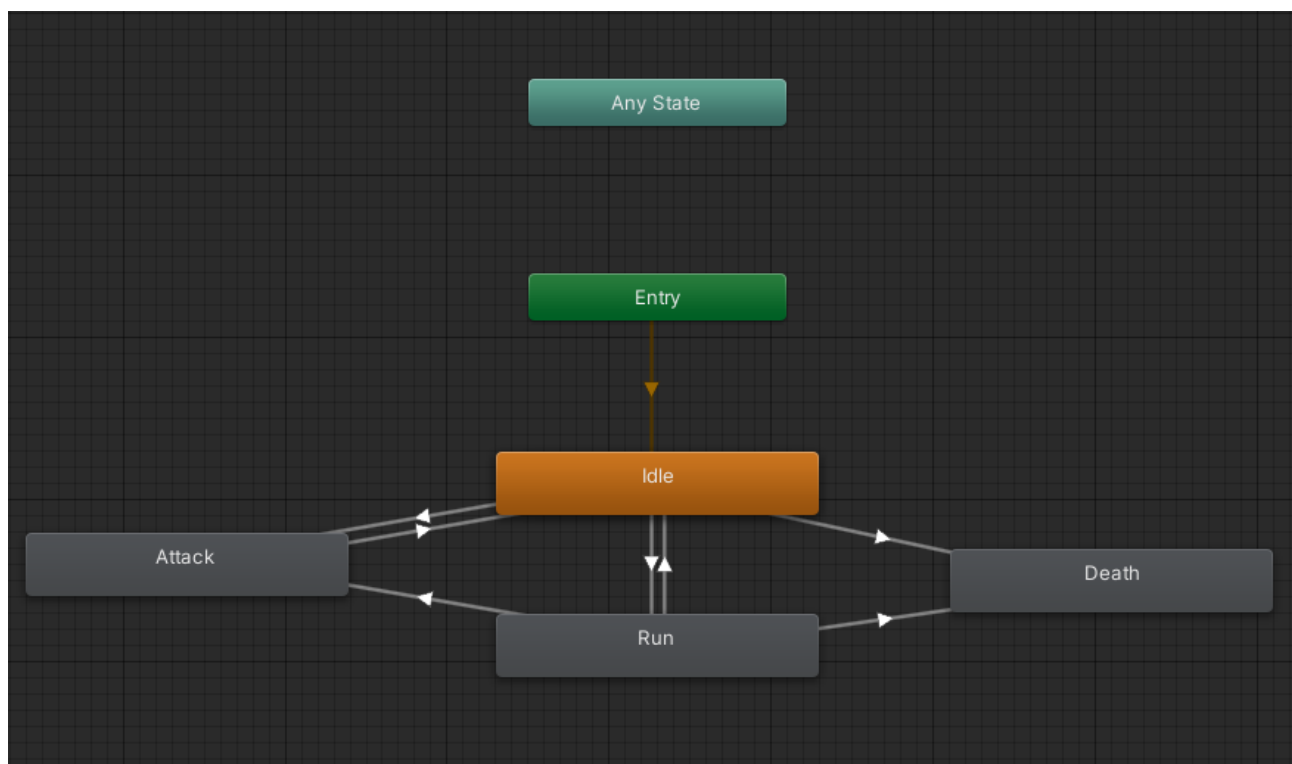
Idle animation



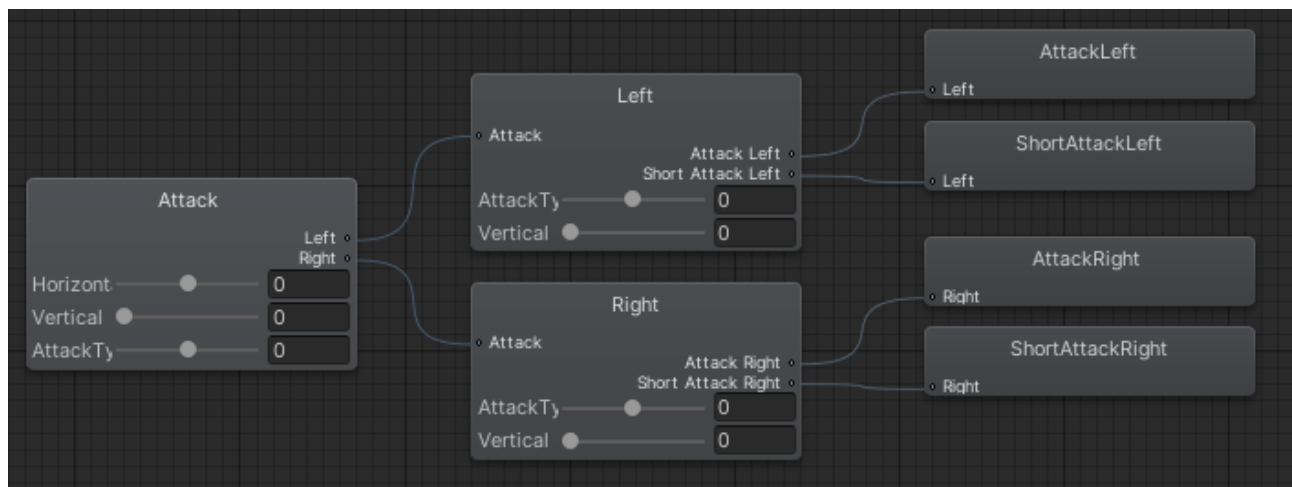
Run animation



Death animation

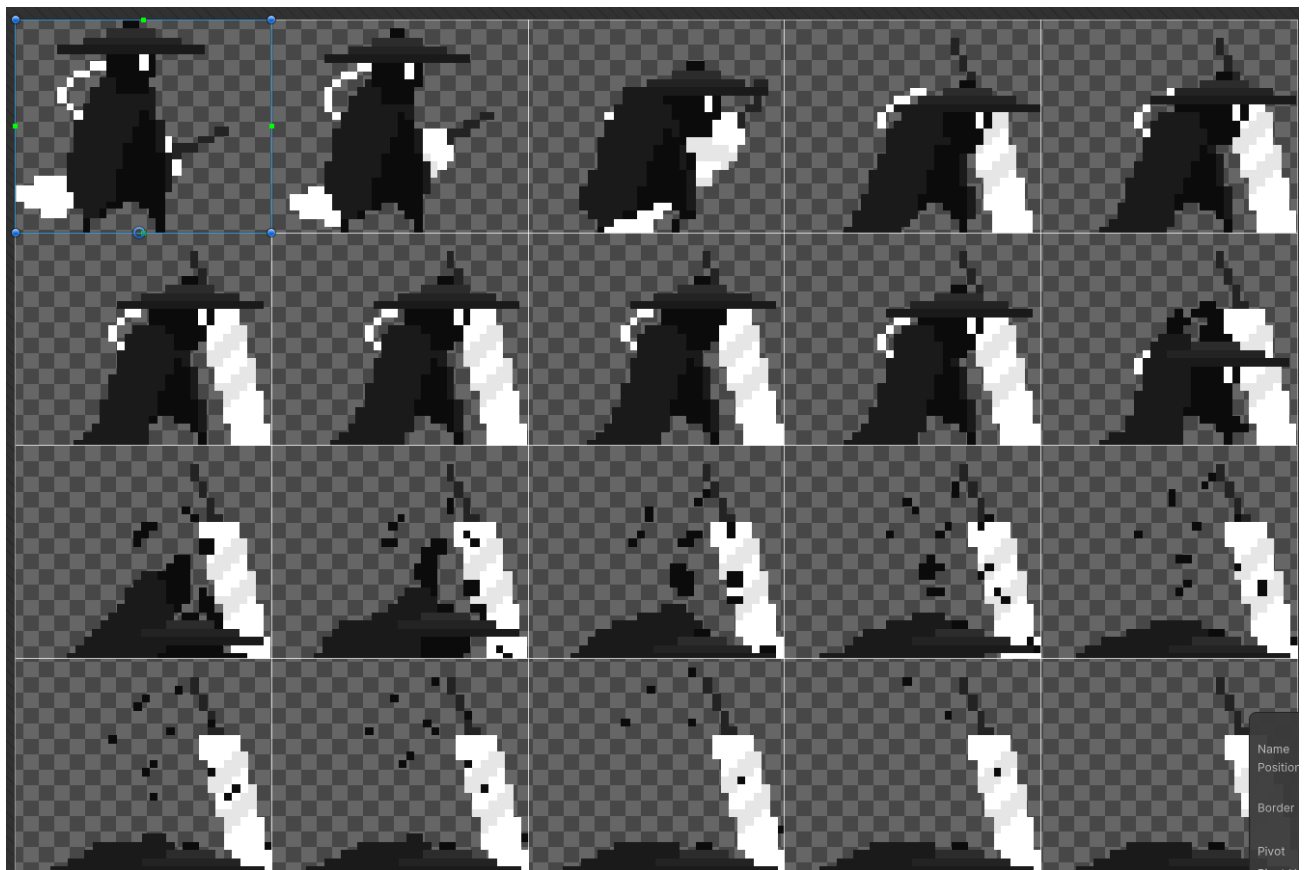


Переходи між анімаціями



Дерева анімацій для атаки

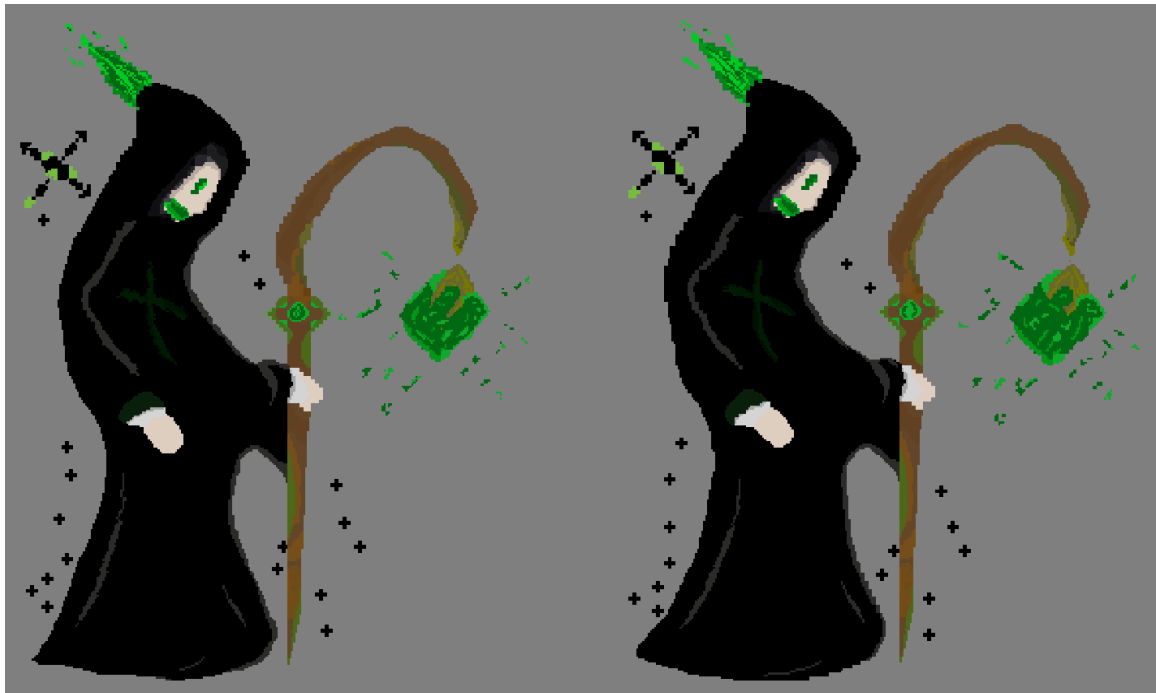
Такі анімації як : біг, стан спокою містять в собі Blend tree в якому декілька можливих анімацій та умови переходу між ними. На прикладі дерева анімацій для атаки показано як це виглядає всередині. Також переходи між анімаціями працюють через тригери та інші визначенні параметри, які встановлюються в скриптах.



Приклад розділення таблиці спрайтів на окремі спрайти



Simbiot animation



Ocultist animation



Damned Spirit animation

- **Скpunmu :**

- **World :**

WorldScrolling.cs

```
using UnityEngine;

public class WorldScrolling : MonoBehaviour
{
    [SerializeField] Transform playerTransform;
    Vector2Int currentPlayerTilePosition = new Vector2Int(0,0);
    [SerializeField] Vector2Int playerTilePosition;

    Vector2Int onTileGridPosition;
    [SerializeField] float blockSize = 36;
    GameObject[,] blockTiles;

    [SerializeField] int blockTileHorizontalCount;
    [SerializeField] int blockTileVerticalCount;

    [SerializeField] int fieldOfVisionHeight = 3;
    [SerializeField] int fieldOfVisionWidth = 3;

    private void Awake()
    {
        blockTiles = new GameObject[blockTileHorizontalCount, blockTileVerticalCount];
    }

    private void Start()
    {
        UpdateTilesOnScreen();
    }

    private void Update()
    {
        playerTilePosition.x = (int)(playerTransform.position.x / blockSize);
        playerTilePosition.y = (int)(playerTransform.position.y / blockSize);

        playerTilePosition.x -= playerTransform.position.x < 0 ? 1 : 0;
        playerTilePosition.y -= playerTransform.position.y < 0 ? 1 : 0;

        if(currentPlayerTilePosition != playerTilePosition)
        {
            currentPlayerTilePosition = playerTilePosition;

            onTileGridPosition.x = calculatePositionOnAxis(onTileGridPosition.x, true);
            onTileGridPosition.y = calculatePositionOnAxis(onTileGridPosition.y, false);
            UpdateTilesOnScreen();
        }
    }

    private void UpdateTilesOnScreen()
    {
        for(int pov_x = -(fieldOfVisionWidth/2); pov_x <= fieldOfVisionWidth/2; pov_x++)
        {
            for(int pov_y = -(fieldOfVisionHeight/2); pov_y <= fieldOfVisionHeight/2;
pov_y++)
            {
```

```

        int tileToUpdate_x = calculatePositionOnAxis(playerTilePosition.x +
pov_x, true);
        int tileToUpdate_y = calculatePositionOnAxis(playerTilePosition.y +
pov_y, false);

        GameObject tile = blockTiles[tileToUpdate_x, tileToUpdate_y];
        tile.transform.position = calculateTilePosition(
            playerTilePosition.x + pov_x,
            playerTilePosition.y + pov_y
        );
    }
}

private Vector3 calculateTilePosition(int x, int y)
{
    return new Vector3(x * blockSize, y * blockSize, 0f);
}

private int calculatePositionOnAxis(float currentValue, bool horizontal)
{
    if(horizontal)
    {
        if(currentValue >= 0)
        {
            currentValue = currentValue % blockTileHorizontalCount;
        }
        else
        {
            currentValue += 1;
            currentValue = blockTileHorizontalCount - 1
                + currentValue % blockTileHorizontalCount;
        }
    }
    else
    {
        if (currentValue >= 0)
        {
            currentValue = currentValue % blockTileVerticalCount;
        }
        else
        {
            currentValue += 1;
            currentValue = blockTileVerticalCount - 1
                + currentValue % blockTileVerticalCount;
        }
    }

    return (int)currentValue;
}

public void Add(GameObject tileGameObject, Vector2Int tilePosition)
{
    blockTiles[tilePosition.x, tilePosition.y] = tileGameObject;
}
}

```

BlockTile.cs

using System.Collections;

```

using System.Collections.Generic;
using UnityEngine;

public class BlockTile : MonoBehaviour
{
    [SerializeField] Vector2Int tilePosition;

    private void Start()
    {
        GetComponentInParent<WorldScrolling>().Add(gameObject, tilePosition);

        transform.position = new Vector3(-100, -100, 0);
    }
}

```

Ці скрипти реалізують циклічну зміну блоків ігрового світу, що надає нам можливість безкінечно рухатись в будь-якому напрямку і по тим же блокам. Логіка працює так що рухаючись в напрямку вправо, при наближенні до крайніх правих блоків, вперед переміщаються задні ліві, тобто ми попадаємо на ті ж блоки але з протилежного кінця. Вверх та вниз працює так само. При необхідності можна змінювати розміри сітки блоків, розширюючи або обрізаючи варіативність світу.



Налаштування сітки блоків

- ***Player :***

Character.cs

```
using UnityEngine;

public class Character : MonoBehaviour
{
    [Header("Stats")]
    [SerializeField] float maxHp = 100f;

    [Space]
    [Header("Settings")]
    [SerializeField] float currentHp;
    [SerializeField] StatusBar hpBar;

    [HideInInspector]
    public bool isDamaged;

    // Attack attributes
    Animator animator;
    AttackControl attackControl;

    [SerializeField] ControlCyberSword cyberSword;

    // Start Parameters

    private void Start()
    {
        // Set HP states

        hpBar.SetState(currentHp, maxHp);
        if (currentHp < maxHp)
            isDamaged = true;
        else
            isDamaged = false;
    }
}
```

```

// Set attack states

animator = GetComponent<Animator>();
attackControl = animator.GetBehaviour<AttackControl>();
}

// Take Damage function

public void TakeDamage(float damage)
{
    currentHp -= damage;
    isDamaged = true;

    if (currentHp <= 0)
    {
        Debug.Log("Character is dead GAME OVER");
        currentHp = -1;
    }
    hpBar.SetState(currentHp, maxHp);
}

// Heal Function

public void Heal(float amount)
{
    if (currentHp <= 0)
        return;

    currentHp += amount;
    if (currentHp >= maxHp)
    {
        currentHp = maxHp;
        isDamaged = false;
    }
}

```

```

hpBar.SetState(currentHp, maxHp);
}

// Main Attack Function

void Update()
{
// Check if btn is clicked and attack not started yet

// Input.GetKeyDown(KeyCode.Space)      - Space btn
// Input.GetMouseButtonDown(0)          - Left mouse btn

bool isShortAttack = Input.GetKeyDown(KeyCode.Q);
bool isHeavyAttack = Input.GetKeyDown(KeyCode.E);

if (!attackControl.IsAttacking)
{
if (isShortAttack || isHeavyAttack)
{
// Start attack animation
animator.SetTrigger("isAttack");
animator.SetFloat("AttackType", (isShortAttack ? -1f : 1f));
}
}
}

private void ActivateHit(bool isFirstHit)
{
if (cyberSword == null)
{
Debug.Log(" Error :: cyberSword is null !");
return;
}
}

```

```
cyberSword.Hit(isFirstHit);  
}
```

```
public void FirstHit()  
{  
    ActivateHit(true);  
}
```

```
public void SecondHit()  
{  
    ActivateHit(false);  
}  
}
```

PlayerMove.cs

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
[RequireComponent(typeof(Rigidbody2D))]  
  
public class PlayerMove : MonoBehaviour  
{  
  
    [HideInInspector]  
    public Vector3 movementVector;  
  
    [HideInInspector]  
    public float lastHorizontalVector;  
  
    [HideInInspector]  
    public float lastVerticalVector;  
  
    AttackControl attackControl;
```



```

Rigidbody2D rigidbody2d;
Animator animator;

// Movement Speed

[SerializeField] float defaultSpeed = 3.0f;
float speed;

// Look Direction for animations
private Vector2 lookDirection = new Vector2(1,0);

private void Awake()
{
    rigidbody2d = GetComponent<Rigidbody2D>();
    movementVector = new Vector3();

    animator = GetComponent<Animator>();
    attackControl = animator.GetBehaviour<AttackControl>();
    speed = defaultSpeed;
}

// Update is called once per frame
void Update()
{
    if (attackControl.IsAttacking)
    {
        speed = 0f;
        return;
    }
    speed = defaultSpeed;

    movementVector.x = Input.GetAxisRaw("Horizontal");
    movementVector.y = Input.GetAxisRaw("Vertical");
    movementVector = movementVector.normalized;

```

```

if (movementVector.x != 0)
{
    lastHorizontalVector = movementVector.x;
}
if (movementVector.y != 0)
{
    lastVerticalVector = movementVector.y;
}

if (!Mathf.Approximately(movementVector.x, 0.0f))
{
    lookDirection.Set(movementVector.x, movementVector.y);
    lookDirection.Normalize();
}

animator.SetFloat("Horizontal", lookDirection.x);
animator.SetFloat("Speed", movementVector.magnitude);
//}
}

void FixedUpdate()
{
    Vector2 position = rigidbody2d.position;
    position.x += speed * movementVector.x * Time.deltaTime;
    position.y += speed * movementVector.y * Time.deltaTime;
    rigidbody2d.MovePosition(position);
}
}

```

StatusBar.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StatusBar : MonoBehaviour

```

```

{
[SerializeField] Transform bar;

public void SetState(float current, float max)
{
float state = (float)current;
state /= max;

if (state < 0f)
{
state = 0f;
};

bar.transform.localScale = new Vector3(state, 1f, 1f);
}
}

```

Дані скрипти описують основні характеристики головного героя, окремий скрипт для його руху який взаємодіє з аніматором та скрипт що керує шкалою здоров'я.

- ***Weapons :***

WhipWeapon.cs

```

using System;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WhipWeapon : MonoBehaviour
{
    [SerializeField] float timeToAttack = 4f;
    float timer;

```

```

[SerializeField] GameObject leftWhipObject;
[SerializeField] GameObject rightWhipObject;

PlayerMove playerMove;

[SerializeField] float whipDamage = 1f;

private void Awake()
{
    timer = timeToAttack;
    playerMove = GetComponentInParent<PlayerMove>();
}

private void Update()
{
    timer -= Time.deltaTime;
    if(timer < 0f)
    {
        Attack();
    }
}

private void Attack()
{
    timer = timeToAttack;

    if (playerMove.lastHorizontalVector >= 0)
    {
        rightWhipObject.SetActive(true);
        Collider2D rightWhipCollider = rightWhipObject.GetComponent<Collider2D>();

        // Check if trigger
        if (!rightWhipCollider.isTrigger)
            rightWhipCollider.isTrigger = true;

        // Damage monsters
        if (rightWhipCollider != null)
        {

```

```

        Collider2D[] colliders =
Physics2D.OverlapBoxAll(rightWhipCollider.bounds.center, rightWhipCollider.bounds.size, 0f);
        ApplyDamage(colliders);
    }
    else
    {
        Debug.Log("Error : Object -> " + gameObject.name + " must be with
Collider2D.");
    }
}
else
{
    LeftWhipObject.SetActive(true);
    Collider2D leftWhipCollider = LeftWhipObject.GetComponent<Collider2D>();

    // Check if trigger
    if (!leftWhipCollider.isTrigger)
        leftWhipCollider.isTrigger = true;

    // Damage monsters
    if (leftWhipCollider != null)
    {
        Collider2D[] colliders =
Physics2D.OverlapBoxAll(leftWhipCollider.bounds.center, leftWhipCollider.bounds.size, 0f);
        ApplyDamage(colliders);
    }
    else
    {
        Debug.Log("Error : Object -> " + gameObject.name + " must be with
Collider2D.");
    }
}

}

private void ApplyDamage(Collider2D[] colliders)
{
    for(int i = 0; i < colliders.Length; i++)
    {
        IDamageable monster = colliders[i].GetComponent<IDamageable>();
        if (monster != null)

```

```

        {
            monster.TakeDamage(whipDamage);
        }
    }
}

```

DisableAfterTime.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DisableAfterTime : MonoBehaviour
{
    [SerializeField] float timeToDisable = 2f;
    float timeToFadeOut;

    bool isAlphaStarted;
    float timer;
    float elapsedTime;

    private void OnEnable()
    {
        elapsedTime = 0f;
        timer = timeToDisable;
        timeToFadeOut = timeToDisable / 2f;
        isAlphaStarted = false;
    }

    private void LateUpdate()
    {
        timer -= Time.deltaTime;

        if (timer <= timeToFadeOut && !isAlphaStarted)
        {
            StartCoroutine(FadeOutAndDisable());
        }
    }

    private IEnumerator FadeOutAndDisable()
    {
        isAlphaStarted = true;

        SpriteRenderer spriteRenderer = GetComponent<SpriteRenderer>();
        Color originalColor = spriteRenderer.color;

        while (elapsedTime < timeToFadeOut)
        {
            float alpha = Mathf.Lerp(1f, 0f, elapsedTime / timeToFadeOut);
            spriteRenderer.color = new Color(originalColor.r, originalColor.g,
originalColor.b, alpha);
            elapsedTime += Time.deltaTime;
            yield return null;
        }

        // Delay for a short time to make sure alpha = 0
    }
}

```

```

        yield return new WaitForSeconds(0.1f);

        // Вимкнути об'єкт після плавного зникнення
        gameObject.SetActive(false);
        spriteRenderer.color = new Color(originalColor.r, originalColor.g, originalColor.b,
1f);
    }

}

```

AttackControl.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AttackControl : StateMachineBehaviour
{
    // Check attack status
    bool isAttacking = false;

    // OnStateEnter is called when attack state is started
    override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int
layerIndex)
    {
        isAttacking = true; // Check attack is started
    }

    // OnStateExit is called when attack state is finished
    override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int
layerIndex)
    {
        isAttacking = false; // Check attack is finished
    }

    // Taking isAttacking from other classes
    public bool IsAttacking
    {
        get { return isAttacking; }
        private set { }
    }
}

```

CyberSword.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CyberSword : MonoBehaviour
{
    [SerializeField] private Collider2D firstHit;
    [SerializeField] private Collider2D secondHit;

    // Activates

    public Collider2D onFirstHit()
    {

```

```

        firstHit.gameObject.SetActive(true);
        return firstHit;
    }
    public Collider2D onSecondHit()
    {
        secondHit.gameObject.SetActive(true);
        return secondHit;
    }

    // Deactivates

    public void offFirstHit()
    {
        firstHit.gameObject.SetActive(false);
    }
    public void offSecondHit()
    {
        secondHit.gameObject.SetActive(false);
    }
}

```

Дані скрипти відносяться до бойової системи, та описують атаки персонажа через певні періоди часу та по натисканню клавіш, далі добавляться скрипти які будуть відповідати за певні здібності героя.

- **Items :**

DestructibleObject.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DestructibleObject : MonoBehaviour, IDamageable
{
    float objectHp = 2f;

    public void TakeDamage(float damage)
    {
        --objectHp;
        if(objectHp <= 0)
        {
            Destroy(gameObject);
        }
    }
}

```

DropOnDestroy.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DropOnDestroy : MonoBehaviour
{

```



```

[SerializeField] GameObject healthPickUp;
[SerializeField] [Range(0f, 1f)] float chance = 1f;

// Flag to check if the application is quitting.
private static bool isQuitting = false;

private void OnApplicationQuit()
{
    isQuitting = true;
}

private void OnDestroy()
{
    // Check if the game is quitting or the object is being destroyed due to scene
    unloading.
    if (isQuitting || !gameObject.scene.isLoaded)
    {
        return;
    }

    if (healthPickUp != null && Random.value < chance)
    {
        Vector3 spawnPosition = new Vector3(transform.position.x, transform.position.y -
0.05f, transform.position.z);
        Instantiate(healthPickUp, spawnPosition, Quaternion.identity);
    }
}
}

```

IDamageable.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public interface IDamageable
{
    public void TakeDamage(float damage)
    {
    }
}

```

PickUp.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pickup : MonoBehaviour
{
    [SerializeField] int healAmount;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        Character c = collision.GetComponent<Character>();
        if (c != null)

```

```

    {
        if (c.isDamaged)
        {
            c.Heal(healAmount);
            Destroy(gameObject);
        }
    }
}

```

Дані скрипти описують поведінку предметів взаємодії. Наприклад при знищенні коробки буде випадати предмет який відновлює шкалу hp, з ворогів уламки темних душ для прокачки і подібне.

- **Monsters :**

Monsters.h

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Monsters : MonoBehaviour, IDamageable
{
    ///////////////////////////////////
    //          Stats          //
    ///////////////////////////////////

    Transform targetDestination;
    GameObject targetGameobject;
    Character targetCharacter;
    ObjectsDetection objectsDetection;
    Rigidbody2D rigidbody2d;
    ColorChange colorChange;

    [Header("Stats")]

    [SerializeField] float hp = 4f;
    [SerializeField] float damage = 12f;
    [SerializeField] float speed = 2.5f * 0.4f;

    // Changeable stats //

    float currentDamage;
    float damageMultiplier = 0.4f;
    float currentSpeed;

    ///////////////////////////////////

    Color defaultColor;
    bool isLeft;
    bool isSpirit;
    float swapTimer;
}

```

```

// Settings //

[Space]
[Header("Settings")]
[SerializeField] Color spiritColor;
[SerializeField] float colorChangeTime = 1f;
[SerializeField] float slowDownSpeed = 0.3f;
[SerializeField] float sideSwapDelay = 4f;

//////////

//private Vector2[] path;
//private int targetIndex;

//////////
//      Methods      //
//////////

// - Awake

private void Awake()
{
    rigidbody2d = GetComponent<Rigidbody2D>();
    objectsDetection = transform.Find("ObjectTrigger").GetComponent<ObjectsDetection>();
    defaultColor = GetComponent<SpriteRenderer>().color;
    colorChange = GetComponent<ColorChange>();
}

// - Start

private void Start()
{
    swapTimer = 0f;
    isSpirit = false;
    currentDamage = damage;
    currentSpeed = speed;
}

// - Set Target

public void SetTarget(GameObject target)
{
    targetGameobject = target;
    targetDestination = target.transform;
}

// - Updates

private void Update()
{
    ////////////
    // Check and change spirit mode //
    ////////////

```

```

    if (objectsDetection != null)
    {
        if (objectsDetection.IsDetected() && !isSpirit)
        {
            isSpirit = true;
            SpiritSettings(isSpirit);
        }
        if (!objectsDetection.IsDetected() && isSpirit)
        {
            isSpirit = false;
            SpiritSettings(isSpirit);
        }
    }
    else
    {
        Debug.Log(" Error :: Monsters -> ObjectTrigger can not be found! Object
Detection fail.");
    }

    //////////////////////////////////////

}

private void FixedUpdate()
{
    Vector3 direction = (targetDestination.position - transform.position).normalized;
    Vector3 swapSide = new Vector3(transform.localScale.x * -1f, transform.localScale.y,
transform.localScale.z);

    //////////////////////////////////////
    // Check and change look side //
    //////////////////////////////////////

    if (targetDestination.position.x - transform.position.x < 0f)
        isLeft = true;
    else
        isLeft = false;

    if ((isLeft && transform.localScale.x > 0f) || (!isLeft && transform.localScale.x <
0f))
    {
        swapTimer += Time.deltaTime;
        currentSpeed *= slowDownSpeed;

        if (swapTimer >= sideSwapDelay)
        {
            currentSpeed = speed;
            transform.localScale = swapSide;
            swapTimer = 0f;
        }
    }

    //////////////////////////////////////

    rigidbody2d.velocity = direction * currentSpeed;
}

// - Collisions

```

```

private void OnCollisionStay2D(Collision2D collision)
{
    if (collision.gameObject == targetGameobject)
    {
        Attack();
    }
}

// - Gameplay methods

private void Attack()
{
    if (targetCharacter == null)
    {
        targetCharacter = targetGameobject.GetComponent<Character>();
    }

    targetCharacter.TakeDamage(currentDamage);
}

public void TakeDamage(float damage)
{
    hp -= damage;

    if (hp < 1)
    {
        Destroy(gameObject);
    }
}

// Method for objects colliding

public void SpiritSettings(bool isSpirit)
{
    if (isSpirit)
    {
        currentDamage *= damageMultiplier;
        StartCoroutine(colorChange.ChangeColor(spiritColor, colorChangeTime));
    }
    else
    {
        currentDamage = damage;
        StartCoroutine(colorChange.ChangeColor(defaultColor, colorChangeTime));
    }
}
}

```

EnemiesManager.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemiesManager : MonoBehaviour
{
    [SerializeField] GameObject enemy;
    [SerializeField] Vector2 spawnArea;
    [SerializeField] Vector2 spawnRange;
    [SerializeField] float spawnTimer;

```

```

[SerializeField] GameObject player;
float timer;

private void Update()
{
    timer -= Time.deltaTime;
    if(timer< 0f)
    {
        SpawnEnemy();
        timer = spawnTimer;
    }
}

private void SpawnEnemy()
{
    Vector3 position = GenerateRandomPosition();

    position += player.transform.position;

    GameObject newEnemy = Instantiate(enemy);

    // Check and set start side

    if (player.transform.position.x - position.x < 0f)
    {
        newEnemy.transform.localScale = new Vector3(
            newEnemy.transform.localScale.x * -1,
            newEnemy.transform.localScale.y,
            newEnemy.transform.localScale.z
        );
    }

    newEnemy.transform.position = position;
    newEnemy.GetComponent<Monsters>().SetTarget(player);
}

private Vector3 GenerateRandomPosition()
{
    Vector3 position = new Vector3();

    float axisSide = UnityEngine.Random.value > 0.5f ? -1f : 1f;

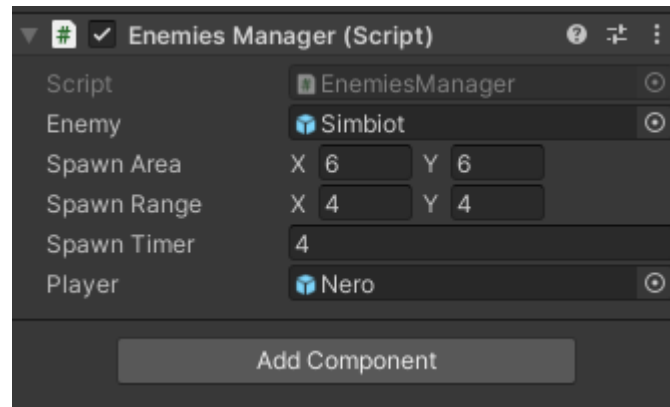
    if(UnityEngine.Random.value > 0.5f)
    {
        position.x = UnityEngine.Random.Range(-spawnArea.x, spawnArea.x);
        position.y = spawnArea.y * axisSide + UnityEngine.Random.Range(0, spawnRange.y);
    }
    else
    {
        position.x = spawnArea.x * axisSide + UnityEngine.Random.Range(0, spawnRange.x);
        position.y = UnityEngine.Random.Range(-spawnArea.y, spawnArea.y);
    }

    position.z = 0;

    return position;
}
}

```

Дані скрипти відповідають за основні характеристики ворогів та їх можливості, окремий скрипт для системи періодичного спавна, де можна динамічно змінювати параметри.



Enemies Manager

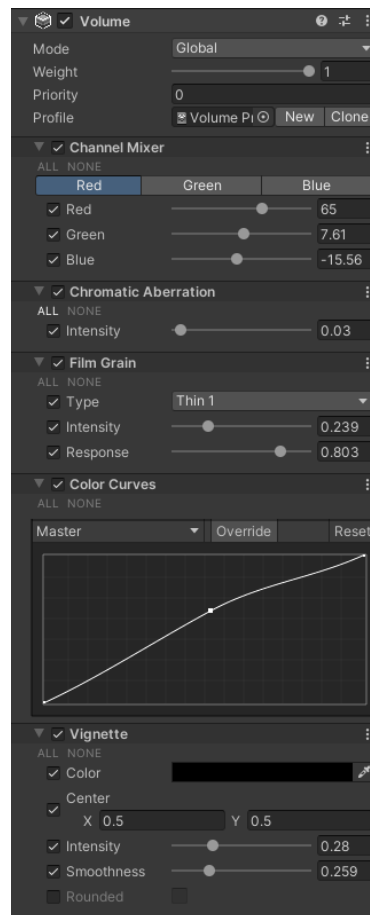
- *Graphics :*



Before custom settings



After custom settings



Volume Settings

Майбутні оновлення...