Część funkcjonalna i niefunkcjonalna gry

Część funkcjonalna:

Must-have:

1. Ruch postaci gracza:
   Gracz może poruszać się w lewo i prawo, skakać (być może rollować) za pomocą odpowiednich klawiszy czyli WSAD, i skakać za pomocą spacji

2. Interakcja z obiektami:
   Postać może zbierać przedmioty, które dodają punkty, zdrowia lub inne korzyści.
   Możliwość aktywacji przycisków, dźwigni i mechanizmów w grze, które wpływają na środowisko (np. otwieranie drzwi, uruchamianie platform).

3. Przeciwnicy:
   Gra zawiera przeciwników, którzy poruszają się po określonych trasach, atakują lub patrolują.
   Przeciwnicy reagują na obecność postaci gracza i mogą zadawać obrażenia, zmniejszając zdrowie postaci gracza.

4. Mechanika śmierci i odradzania się:
   Postać gracza traci życie, gdy jej zdrowie spadnie do zera lub wpadnie w przepaść.
   Po stracie życia gracz odradza się na na początku rozpoczętego poziomu.

5. Poziomy gry:
   Gra będzie zawierała wiele poziomów, które stopniowo zwiększają poziom trudności.

Każdy poziom kończy się osiągnięciem wyznaczonego celu (np. dotarcie do końca planszy, pokonanie bossa).

6. Punkty kontrolne (checkpointy):
   Gra posiada punkty kontrolne, czyli po przejściu danego poziomu po śmieci gracza będzie się on odradzał już na nowym poziomie (możliwe że poziomy będzie można wybierać po przejściu danego poziomu odblokuje się on do wyboru w menu)

7. System zdrowia i obrażeń:
   Postać gracza ma pasek zdrowia, który zmniejsza się po otrzymaniu obrażeń.
   Gra pozwala na odzyskiwanie zdrowia poprzez zbieranie odpowiednich przedmiotów.

8. Animacje postaci i obiektów:
   Postać gracza i przeciwnicy mają animacje ruchu, ataków, skoków (tylko gracz) i śmierci.
   Niektóre elementy otoczenia (np. przesuwające się platformy) mają animacje.

9. Menu gry:
   Gra zawiera główne menu z opcjami rozpoczęcia nowej gry, wybrania poziomu, ustawień oraz wyjścia.
   W ustawieniach gracz może dostosować np. poziom głośności muzyki i dźwięków.

10. Interfejs ekwipunku:
    Ekran ekwipunku wyświetla wszystkie posiadane przez gracza przedmioty.
    Gracz może przeglądać, wybierać i aktywować przedmioty z ekwipunku za pomocą intuicyjnego interfejsu.

11. Ograniczenia ekwipunku:
    Ekwipunek może mieć ograniczoną pojemność (tylko 6/8 przedmiotów typu head, body etc.). Dodatkowo z 10 przedmiotów w plecaku

12. Przedmioty aktywne i pasywne:
    Gracz może używać niektórych przedmiotów bezpośrednio z ekwipunku (eliksirów leczniczych, many etc.).

    Może zakładać zbroje aby zdobyć defensywne statystki, miecz, łuk itd. aby zwiększyć swoje obrażenia

13. Otwieranie skrzyń z losowym łupem:

    Skrzynki będą posiadały przedmioty związane z samą postacią jak czy miecz, łuk itd. dodatkowo eliksiry, monety lub jakieś krótkie wzmocnienia postaci(o ile zostaną wprowadzone)

14. System klas postaci:

    Będą do wyboru trzy unikalne postacie z własnymi umiejętnościami, ekwipunkiem itd.

Optional:

1. Kategoryzacja przedmiotów:
   Ekwipunek może być podzielony na różne kategorie, takie jak broń, zbroje, eliksiry, klucze, itp., aby ułatwić zarządzanie przedmiotami.

2. System monet:
   Gracz zdobywa monety za pokonywanie przeciwników/bossów/otwieranie skrzyń lub ukończenie poziomów. Później na koniec poziomów będzie mógł je wydać w sklepiku

3. System statusów:
   Wprowadzenie efektów statusowych, takich jak zatrucie, spowolnienie, ogłuszenie, które mogą wpływać na postać gracza lub przeciwników.

4. .Pułapki i przeszkody:
   Na poziomach mogłyby znajdować się różne pułapki (np. kolce, strzały, zapadnie), które gracz musi ominąć lub dezaktywować, co dodaje dodatkowy element wyzwania.

Część niefunkcjonalna:

Must-have:

1. Wydajność:

- Gra powinna działać w 30 FPS.
- Poziomy powinny ładować się szybko.
- Gra powinna być zoptymalizowana pod kątem zużycia zasobów systemowych.

2. Interfejs użytkownika:

- Gracz powinien intuicyjnie rozumieć zasady gry i sposób interakcji z interfejsem.
- Wszystkie menu powinny być łatwe w obsłudze.
- Interfejs powinien być responsywny.

Użyteczność:

- Sterowanie postacią powinno być wygodne i precyzyjne.
- Gra powinna wspierać obsługę klawiatury i myszki.
- Sterowanie postacią odbywa się za pomocą klawiszy WASD i spacji.
- Gra powinna zawierać NPC który wprowadzi gracza do gry.

Kompatybilność:

- Gra powinna działać na systemie Windows.

Rozgrywka:

- Gra powinna zapewniać odpowiedni balans poziomu trudności, tak aby wyzwanie stopniowo rosło, ale jednocześnie nie było zbyt wysokie na początku.
- Gra powinna być możliwa do przejścia niezależnie od wybranego bohatera.

Dźwięk i Grafika:

- Wszystkie animacje powinny być płynne i spójne.
- Dźwięki powinny być odpowiednio dobrane do konkretnych akcji w grze.
- Grafiki obiektów i postaci powinny być spójne z ogólną stylistyką gry.
- Każdy poziom powinien mieć unikalny wygląd.

Testowanie:

- W miarę dodawania nowych funkcji do gry, powinniśmy testować grę, aby upewnić się, że wprowadzone zmiany nie mają wpływu na ogólną stabilność gry oraz działanie już istniejących mechanik

Optional:

1. Gra powinna obsługiwać różne rozdzielczości ekranu

Dlaczego unity:

- **Tilemaps** – system do budowania poziomów z siatek kafelków, idealny do tworzenia platformówek. Umożliwia łatwe i szybkie projektowanie złożonych światów.

- **Cinemachine** – narzędzie do dynamicznego tworzenia kamer, które świetnie współgra z ruchem i akcją w platformówkach. Pozwala na zaawansowane kontrolowanie ujęć bez potrzeby pisania skomplikowanego kodu.
- **Ogromne możliwości animacyjne** – Unity oferuje szeroki zakres narzędzi do animacji większości elementów gry. Można łatwo edytować animacje na różne sposoby, np. dodawać eventy w trakcie animacji, które wykonają określone akcje w grze.
- **Wsparcie społeczności** – Unity ma jedną z największych i najbardziej aktywnych społeczności twórców gier. Dokumentacja jest obszerna, ale też dobrze napisana, co ułatwia naukę. Jest mnóstwo poradników, gotowych rozwiązań, darmowych zasobów oraz bibliotek i narzędzi dostępnych na Asset Store, które mogą przyspieszyć rozwój gry.
- **Monetyzacja** – Unity dostarcza wbudowane rozwiązania, takie jak Unity Ads, umożliwiające łatwą integrację różnych form zarabiania na grze. W przeciwieństwie do Unity, Godot nie oferuje takich rozwiązań i trzeba samodzielnie je implementować.
- **Doświadczenie i prostota** – Mam już pewne doświadczenie z Unity, ponieważ zrealizowałem dwa mniejsze projekty w tym środowisku. Unity jest intuicyjne i proste w obsłudze, a jednocześnie oferuje wiele możliwości oraz funkcji, które ułatwiają programowanie. To szczególnie przydatne, gdy ktoś dopiero zaczyna swoją przygodę z tworzeniem gier, a tylko ja mam minimalne doświadczenie, reszta zespołu nie posiada.
- **Asset Store vs. Godot Asset Library** – Unity posiada bardzo rozbudowany Asset Store, w którym można znaleźć zarówno darmowe, jak i płatne zasoby, skrypty i narzędzia, które znacznie przyspieszają proces tworzenia gry. Godot ma swoją bibliotekę zasobów (Godot Asset Library), ale jest ona mniej rozbudowana i zawiera mniej gotowych narzędzi w porównaniu do tego, co oferuje Unity.