Dokumentacja projektu z Inżynieria oprogramowania A

Main Designer Marcel Kasprzycki, SCRUM director Jan Stachura,
Main developer Piotr Kociołek, Support Developer Paweł Myjak-Zasadni

14 listopada 2023

Streszczenie

Gra karciana z elementami rougelike stworzona za pomocą Godota w metodzie planowania SCRUM.

Spis treści

1 Tytuł

Karthic gra karciana

2 Nazwa robocza

Karcianka

3 Opis

Gra Karthic to karcianka z elementami Rogue-like łącząca elementy gier karcianych, przygodowych i roguelike. Gracz wcielający się w bohatera odkrywa świat poprzez przemierzanie mapy, kolekcjonując talie kart, którą wykorzysta w starciach z przeciwnikami. Talia kart zawierać będzie ataki o różnym koszcie many, i różnych obrażeniach. W walce przeciwnicy losowo będą wybierać z dostępnych ataków. Gracz rozgrywkę zacznie z 3 punktami many i kilkoma podstawowymi kartami i podstawowym ekwipunkiem, który w trakcie gry rozwinie. Celem gry będzie zgromadzenie dobrego ekwipunku pokonanie bossa.

4 Cel

Kolekcjonowanie i rozwój: Poprzez zdobywanie nowych kart i ekwipunku, gracz będzie mógł ulepszać swoją talię i zdolności bohatera, co umożliwi mu przystosowanie się do różnorodnych sytuacji oraz rozwijanie własnej strategii w walce.

Pokonanie bossa: Ostatecznym celem gry będzie stawić czoła potężnemu bossowi, co stanowi wyzwanie wymagające nie tylko odpowiedniej siły, ale także strategicznego myślenia oraz umiejętności planowania.

5 Zakres

5.1 Analiza wymagań

Aplikacja powinna oferować system walki, który bazuje na losowych kartach z kosztem użycia, co sprawia, że rozgrywka jest dynamiczna i strategiczna. Dzięki elementowi losowości w kartach oraz różnym poziomom trudności przeciwników gra zapewnia wysoki poziom re grywalności. Gracze mogą wielokrotnie powtarzać rozgrywkę, eksplorując nowe strategie i osiągając lepsze wyniki. Dodatkowo klarowny system prezentacji wyników gracza dostarcza informacji o osiągnięciach w czytelny sposób, zachęcając graczy do rywalizacji z innymi oraz motywując ich do poprawy swojego wyniku lub odkrycia nowych strategii.

Pierwszym ekranem dostępnym dla użytkownika jest menu główne gry, które zawiera następujące przyciski:

Nowa Gra: Ten przycisk pozwala graczowi rozpocząć nową grę i przenieść się do fazy eksploracji. Tabela Wyników: Przy użyciu tego przycisku gracz może sprawdzić najlepsze wyniki innych graczy. Wyjście: Przycisk ten umożliwia użytkownikowi wyjście z aplikacji. Po

rozpoczęciu gry, gracz zostanie przekierowany do rozgrywki, która jest podzielona na dwie fazy:

Faza Eksploracji: W tej fazie postać gracza porusza się po dwuwymiarowej mapie, eksplorując ją i szukając przeciwników. Po wejściu w obszar z przeciwnikiem rozpoczyna się druga faza.

Faza Walki: W tej fazie gracz i przeciwnik zostają przeniesieni do sceny walki. Walka opiera się na wymianie ataków za pomocą kart. Gracz ma dostęp do kart, które są przypisane od góry oraz kart zdobytych podczas gry. Na początku walki, gracz otrzymuje trzy losowe karty z własnej talii i może je zagrywać, zużywając punkty many. Po zakończeniu swojego ruchu przeciwnik wykonuje swój ruch. Walka trwa do momentu, kiedy któryś z uczestników straci wszystkie punkty zdrowia.

Gra przewiduje dwie walki z podstawowymi przeciwnikami oraz walkę z bossem. Po wygranej walce z ostatnim przeciwnikiem lub po utracie wszystkich punktów zdrowia gracz zostaje przeniesiony do panelu wyników, gdzie zapisywane są jego osiągnięcia. Informacje o wynikach są przechowywane w bazie danych, umożliwiając graczowi śledzenie swojego postępu oraz konkurowanie z innymi graczami.

5.2 Wymagania niefunkcjonalne

- Prosty i responsywny interfejs użytkownika..
- Zapewnienie bezpieczeństwa danych aplikacji.
- Gra musi reagować na ruchy gracza płynnie i szybko.
- Interfejs użytkownika musi być czytelny na różnych rozdzielczościach ekranu.
- Gra powinna być wyważona, tak aby umiejętności gracza miały wpływ na wynik rozgrywki.
- Losowość elementów roguelike powinna być kontrolowana, aby uniknąć zbyt dużego wpływu czynnika losowego.
- Możliwość dodawania nowych kart, poziomów i zawartości w przyszłości poprzez aktualizacje gry.

5.3 Wymaganie Funkcjonalne

- Wybór z różnych rodzajów kart, każda z unikalnymi zdolnościami i wartościami.
- Wprowadzenie systemu trwałej śmierci postaci po przegranej, gracz rozpoczyna od nowa.
- Gracz wykonuje ruchy na mapie gry.
- Walka z przeciwnikami za pomocą kart.
- Graficzny interfejs użytkownika wyświetlający aktualną talię kart, statystyki postaci, poziomy, zdobyte karty i inne informacje.
- Wyświetlanie informacji o postępie i osiągnięciach gracza.

5.4 Wybór technologi

1. Godot to darmowy i otwarty silnik do tworzenia gier, który cechuje się elastycznością, wsparciem dla wielu języków programowania, oferuje wizualny edytor do projektowania gier, obsługuje gry zarówno 2D, jak i 3D, jest wieloplatformowy, umożliwia eksport gier na różne systemy operacyjne i urządzenia, Zapewnia możliwość dostosowywania kodu źródłowego i rozbudowywania go.

2. Do zaimplementowania bazy danych SQL zostanie wykorzystany system obsługi PostgreSQL.

3. Docker – otwarte oprogramowanie służące do realizacji wirtualizacji na poziomie systemu operacyjnego, działające jako „platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych”.

6 Scenariusze

Nazwa	Scenariusz 1 - uruchomienie Gry
Aktor	gracz
Warunki początkowe	użytkownik będzie miał zainstalowaną grę na urządzeniu.
Opis	żeby zagrać wymaga uruchomienie gry.
Ścieżki główne	1. Użytkownik wyszykuje ikonę gry". 2. użytkownik 2x klika (ppm) na ikonę " 3. gracz oczekuje na wczytanie menu głównego"
Ścieżka alternatywna	2a używa przycisku "enter "na klawiaturze..
Warunki końcowe	Pomyślnie uruchomiono grę.

Nazwa	Scenariusz 2 - Wyświetlenie wyniku
Aktor	gracz
Warunki początkowe	Scenariusz 1.
Opis	Wyświetlenie wyników.
Ścieżki główne	Użytkownik z wyświetlonego menu wybiera "Wyniki"co powoduje pobranie wyników z bazy i wyświetlenia ich na ekranie.
Warunki końcowe	Pomyślnie wyświetlenie wyników.

Nazwa	Scenariusz 3 - Wyjście z gry
Aktor	gracz
Warunki początkowe	Scenariusz 1.
Opis	Wyjście z gry.
Ścieżki główne	Użytkownik z wyświetlonego menu wybiera "Wyjdź z gry"co powoduje wyjście z gry do pulpitu urządzenia .
Warunki końcowe	Pomyślnie wyświetlenie wyników.

Nazwa	Scenariusz 4 - Rozpoczęcie gry
Aktor	gracz
Warunki początkowe	Scenariusz 1.
Opis	rozpoczęcie gry.
Ścieżki główne	Użytkownik z wyświetlonego menu wybiera "rozpocznij grę"co powoduje rozpoczęcie gry.
Warunki końcowe	Gracz może grać w grę.

Tabela 4: Scenariusz 4

Nazwa	Scenariusz 5 - Poruszanie się postaci
Aktor	gracz
Warunki początkowe	Scenariusz 4.
Opis	poruszanie postacią .
Ścieżki główne	gracz, by poruszać się postacią używa klawiszy D- ruchu w przód oraz A- ruchu do tyłu co powoduje przemieszczenie się postacią .
Warunki końcowe	Gracz porusza się postacią .

Tabela 5: Scenariusz 5

Nazwa	Scenariusz 6 - Walka z przeciwnikiem
Aktor	gracz
Warunki początkowe	Scenariusz 6.
Opis	interakcja z kartami (walki).
Ścieżki główne	Użytkownikowi na środku dolnej części ekranu wyświetlają się 3 karty każda karta ma odpowiednie wartości, gracz kursorem najeżdża na kartę używa LPM i kieruje kartą na przeciwnika
Warunki końcowe	Użycie karty w celu zadania obrażeń lub obrony .

Tabela 6: Scenariusz 6

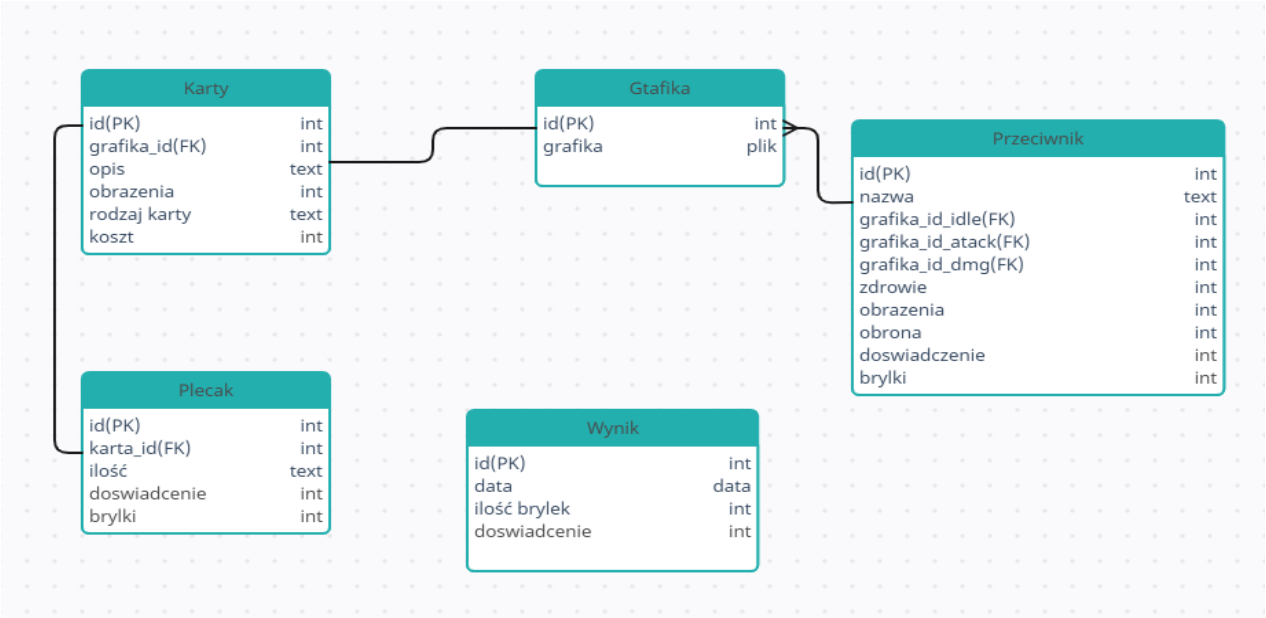
Nazwa	Scenariusz 7 - interakcja z kartami
Aktor	gracz
Warunki początkowe	Scenariusz 1.
Opis	Walka z przeciwnikiem celu przejścia gry.
Ścieżki główne	Użytkownik aktywuje przeniesienie na scenę walki poprzez wejście w przeciwnika, następuje wylosowanie talii kart oraz naprzemienne tury gracza i NPC. Po pokonaniu przeciwnika jego model znika z mapy. W przypadku przegranej gra kończy się wyświetleniem sceny GAME OVER
Warunki końcowe	Pokonanie NPC lub zostanie pokonany, gracz dalej porusza się po scenie world.

Tabela 7: Scenariusz 7

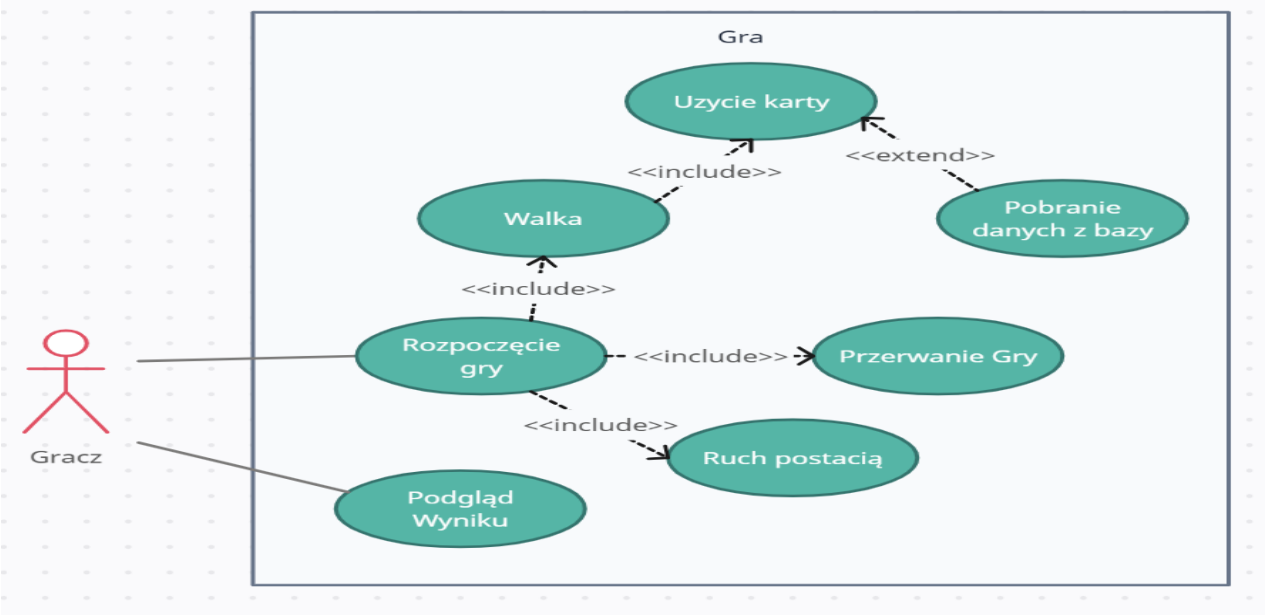
Nazwa	Scenariusz 8 - Ukończenie gry
Aktor	gracz
Warunki początkowe	Scenariusz 4.
Opis	ukończenie gry.
Ścieżki główne	gracz grają w grę jest zmuszony to walki z przeciwnikami po pokonaniu wszystkich przeciwników następuje ukończenie gry
Warunki końcowe	ukończenie gry gracz berdzie mógł się poruszać po oczyszczonej mapie .

Tabela 8: Scenariusz 8

7 Diagramy



Rysunek 1: Diagram bazy danych



Rysunek 2: Diagram przypadków użycia

8 Implementacja

8.0.1 implementacja bazy danych kart

```
# Unitinfo = [Type, Attack, Retaliation, Health, Cost, Name, Melee or Ranged, Special Text]
# Eventinfo = [Type, Cost, Effect]
enum {Footman, Archer, SquadLeader, Warrior, Guardian, Knight, Mercenary, Spearman, Mentor, Trebuchet}

const DATA = {
    Footman :
        ["Units", 1, 1, 2, 1, "Footman", "Melee"],
    Archer :
        ["Units", 2, 1, 3, 2, "Archer", "Ranged,\nImmune to\nRetaliation"],
    SquadLeader :
        ["Units", 2, 2, 3, 3, "Squad Leader", "Melee,\nGive all friendly\n+1 Attack and \nRetaliation"],
    Warrior :
        ["Units", 4, 0, 2, 3, "Rogue", "Melee,\nImmune to\nRetaliation"],
    Guardian :
        ["Units", 1, 3, 6, 3, "Guardian", "Melee,\nProtector - stops the unit\nbehind it\nbeing attacked"],
    Knight :
        ["Units", 2, 3, 6, 4, "Knight", "Melee"],
    Mercenary :
        ["Units", 2, 2, 0, 2, "Mercenary", "Melee,\nAlways Retaliates\nReturn to Supply when damaged\nor at start of ne
    Spearman :
        ["Units", 2, 2, 5, 3, "Spearman", "Melee or Ranged"],
    Mentor :
        ["Units", 3, 0, 1, 2, "Mentor", "Melee,\nWhen played give\nfriendly unit +2 Attack\nand Retaliation"],
    Trebuchet :
        ["Events", 4, "Deal 6 damage\nto a unit"],
    }
```

Rysunek 3: implementacja testowej struktury kart

9 Podsumowanie

10 Plan Zadań Scrum

10.0.1 Sprint 1 ""od 03.10 Do 10.10.2023

1. Wybór tematu — Jan Stachura 1h
2. Przygotowanie gita — Marcel Kasprzycki 1h
3. Przygotowanie szkieletu dokumentacji — Piotr Kociołek 2h
4. Analiza rynku — Paweł Myjak- Zasadni 1h
5. Wybranie technologi — Marcel Kasprzycki 2h

10.0.2 Sprint 2 ""od 10.10 Do 17.10.2023

1. Przygotowanie wstępnych pre-asetów graficznych — Jan Stachura 10h
2. Przygotowanie klasy postaci — Marcel Kasprzycki 8h
3. Przygotowanie szkieletu ruchu i przemieszczania się po mapie — Paweł Myjak-Zasadni 12h
4. Przygotowanie systemu obrażeń i przeciwników — Piotr Kociołek 7h
5. Rozwój dokumentacji — Wszyscy 5h

10.0.3 Sprint 3 ""od 17.10 Do 24.10.2023

1. Grafiki pierwszej warstwy sceny i warstwy drugiej — Jan Stachura
2. Menu główne i ekran porażki 6h — Marcel Kasprzycki
3. System losowania kart — Paweł Myjak-Zasadni
4. Scena walki po wejściu w przeciwnika, z opcją ucieczki — Piotr Kociołek
5. Rozwój dokumentacji — Wszyscy

10.0.4 Sprint 4 ""od 24.10 Do 31.10.2023

1. Przygotowanie szkieletu bazy danych w SQL wraz z poznaniem się technologi 6h— Jan Stachura
2. Obróbka graficzna assetów. Edycja sceny Game Over 6h— Marcel Kasprzycki
3. stworzenie bazy kart oraz assetów 5h — Paweł Myjak-Zasadni
4. — dodanie typów kart, rzucania kart oraz pola dropzone. 4h Piotr Kociołek
5. Rozwój dokumentacji — Wszyscy

10.0.5 Sprint 5 ""od 31.10 Do 07.11.2023

1. Podpięcie bazy danych do głównej aplikacji 10h — Jan Stachura
2. Poprawki graficzne assetów kart, przygotowanie sprite'a bohatera gry 7h— Marcel Kasprzycki
3. interakcje z kartami (wyświetlenie ich oraz wybranie karty atakującej, oraz przycisk (ikona) do wylosowania kart) 8h — Piotr Kociołek
4. — poprawa bieżących kodów w celu optymalizacji 5h - Paweł Myjak-Zasadni
5. Rozwój dokumentacji — Wszyscy

10.0.6 Sprint 6 ""od 07.11 Do 14.11.2023

1. Przygotowanie animacji ataku przeciwnika — Jan Stachura 10h
2. Przygotowanie nowego typu przeciwnika — Marcel Kasprzycki 7h
3. Dodanie kart, oraz ich mechaniki. - Piotr Kociołek 4h
4. Wprowadzenie systemu walki turowej—Paweł Myjak-Zasadni 12h
5. Rozwój dokumentacji — Wszyscy 4h

10.0.7 Sprint 7 ""od 14.11 Do 21.11.2023

1. Przygotowanie spraita Bossa — Jan Stachura 8h
2. Nieskończone przesuwanie mapy i poprawki kamery — Marcel Kasprzycki 4h
3. Dobieranie kart — Piotr Kociołek 3h
4. Rozwój świata gry — Paweł Myjak-Zasadni 10h
5. Rozwój dokumentacji — Wszyscy 3h

10.0.8 Sprint 8 ""od 21.11 Do 28.11.2023

1. Zliczanie punktów do tabeli i zapis ich do bazy - Jan Stachura 8h
2. Randomizacja kart w "ręce"jak i losowanie nowych - Piotr Kociołek 5h
3. Rozwój przeciwników, dodanie specialnych efektów - Paweł Myjak-Zasadni 8h
4. Wprowadzenie bazy danych, nadanie id kartom oraz przypisanie efektów (m.in. zadawanie obrażeń) - Marcel Kasprzycki
5. Rozwój dokumentacji — Wszyscy 3h

10.0.9 Sprint 9""od 28.11 Do 05.12.2023

1. Poprawa i rozwój pola rozgrywki wraz z tłem- Jan Stachura 10h
2. Dodanie animacji głównego bohatera - Paweł Myjak-Zasadni 10h
3. Rozwój działania kart o dodatkowe efekty - Piotr Kociołek i Marcel Kasprzycki - 5h
4. spięcie całości projektu (branch merging) - 1h Piotr Kociołek
5. Rozwój dokumentacji — Wszyscy 3h

10.0.10 Sprint 10 ""od 05.12 Do 12.12.2023

1. dokończenie animacji - Paweł Myjak-Zasadni 5h
2. poprawki wizualne - Marcel Kasprzycki 3h
3. optymalizacja kodu- Piotr Kociołek 4h
4. dokończenie assetów - Jan Stachura
5. Rozwój dokumentacji — Wszyscy 3h

10.0.11 Sprint 11 ""od 12.12 Do 16.01.2024

1. test aplikacji - Jan Stachura 4h
2. test aplikacji - Marcel Kasprzycki 4h
3. test aplikacji - Paweł Myjak-Zsadni 4h
4. test aplikacji - Piotr Kociołek 4h
5. poprawa kodu i błędów - wszyscy 16h
6. Rozwój dokumentacji — Wszyscy 3h

11 Estymacja czasowa

11.0.1 Sprint 2

- Przygotowanie systemu obrażeń i przeciwników 7h
- Przygotowanie klasy postaci 7h
- przygotowanie szkieletu ruchu i przemieszczania się po mapie 10h
- Obróbka graficzna assetów 12h
  Zakładany czas wykonania wynosił 42h
  Rzeczywisty czas poświęcony to: 41h
  Data wykonania zadań 10.10.2023-17.10.2023r

11.0.2 Sprint 3

- Przygotowanie systemu obrażeń i przeciwników 7h
- Menu główne i ekran porażki 7h
- System Losowania kart 10h
- Obróbka graficzna assetów 10h
- Scena walki po wejściu w przeciwnika, z opcją ucieczki 6h
  Zakładany czas wykonania wynosił 45h
  Rzeczywisty czas poświęcony to: 40h
  Data wykonania zadań 17.10.2023-24.10.2023r

11.0.3 Sprint 4

- Przygotowanie szkieletu bazy danych w SQL wraz z poznaniem się technologi 7h
- tworzenie bazy kart oraz assetów 8h
- dodanie typów kart, rzucania kart oraz pola dropzone 5h
- Obróbka graficzna assetów 2h
  Zakładany czas wykonania wynosił 20h
  Rzeczywisty czas poświęcony to: 22h
  Data wykonania zadań 24.10.2023-31.10.2023r

11.0.4 Sprint 5

- Podpięcie bazy danych do głównej aplikacji 10h — Jan Stachura
- Poprawki graficzne assetów kart, przygotowanie sprite'a bohatera gry 7h— Marcel Kasprzycki

- interakcje z kartami (wyświetlenie ich oraz wybranie karty atakującej, oraz przycisk (ikona) do wylosowania kart) 8h — Piotr Kociołek
- — poprawa bieżących kodów w celu optymalizacji 5h - Paweł Myjak-Zasadni
- Rozwój dokumentacji — Wszyscy 3h
  Zakładany czas wykonania wynosił 39h
  Rzeczywisty czas poświęcony to: 30h
  Data wykonania zadań 31.10.2023-07.10.2023r

11.0.5 Sprint 6

- Przygotowanie animacji ataku przeciwnika — Jan Stachura 8h
- Przygotowanie nowego typu przeciwnika — Marcel Kasprzycki 6h
- Bieżące poprawy kodu i poprawa grafiki kart — Piotr Kociołek 7h
- Wprowadzenie systemu walki turowej—Paweł Myjak-Zasadni 7h
- Rozwój dokumentacji — Wszyscy 2h
  Zakładany czas wykonania wynosił 37h
  Rzeczywisty czas poświęcony to: 30 h
  Data wykonania zadań 07.10.2023-14.10.2023r

11.0.6 Sprint 7

- Przygotowanie i wdrożenie finalnego Bossa — Jan Stachura 9h
- Nieskończone przesuwanie mapy i poprawki kamery — Marcel Kasprzycki 3h
- Dodanie neutralnych npc — Piotr Kociołek 5h
- Rozwój świata gry — Paweł Myjak-Zasadni 12h
- Rozwój dokumentacji — Wszyscy 3h
  Zakładany czas wykonania wynosił 32h
  Rzeczywisty czas poświęcony to: 30h
  Data wykonania zadań 14.10.2023-21.10.2023r

11.0.7 Sprint 8 od 21.11 Do 28.11.2023

1. Zliczanie punktów do tabeli i zapis ich do bazy - Jan Stachura 8h
2. Randomizacja kart w "ręce"jak i losowanie nowych - Piotr Kociołek 5h
3. Rozwój przeciwników, dodanie specjalnych efektów - Paweł Myjak-Zasadni 8h
4. Wprowadzenie bazy danych, nadanie id kartom oraz przypisanie efektów (m.in. zadawanie obrażeń).- Marcel Kasprzycki 7h

5. Rozwój dokumentacji — Wszyscy 3h
   Zakładany czas wykonania wynosił 31h
   Rzeczywisty czas poświęcony to: 35h
   Data wykonania zadań 21.11.2023-28.11.2023r

11.0.8 Sprint 9 od 28.11 Do 05.12.2023

1. Poprawa i rozwój pola rozgrywki wraz z tłem- Jan Stachura 10h
2. Dodanie animacji głównego bohatera - Paweł Myjak-Zasadni 10h
3. Dodanie kosztu karty w turze. Rozwój kart o kolejne efekty- Piotr Kociołek i Marcel Kasprzycki - 5h
4. spięcie całości projektu (branch merging) - 1h Piotr Kociołek
5. Rozwój dokumentacji — Wszyscy 3h
   Zakładany czas wykonania wynosił 29h
   Rzeczywisty czas poświęcony to: 29h
   Data wykonania zadań 28.11.2023 05.12.2023r

11.0.9 Sprint 10 od 05.12 Do 12.12.2023

1. dokończenie animacji - Paweł Myjak-Zasadni 5h
2. poprawki wizualne - Marcel Kasprzycki 3h
3. optymalizacja kodu- Piotr Kociołek 4h
4. dokończenie assetów - Jan Stachura 5h
5. Rozwój dokumentacji — Wszyscy 3h
   Zakładany czas wykonania wynosił 20h
   Rzeczywisty czas poświęcony to: 25h
   Data wykonania zadań 05.12.2023-12.12.2023r

11.0.10 Sprint 11 od 12.12 Do 16.01.2024

1. test aplikacji - Jan Stachura 4h
2. test aplikacji - Marcel Kasprzycki 4h
3. test aplikacji - Paweł Myjak-Zsadni 4h
4. test aplikacji - Piotr Kociołek 4h
5. poprawa kodu i błędów - wszyscy 16h
6. Rozwój dokumentacji — Wszyscy 3h
   Zakładany czas wykonania wynosił 35h
   Rzeczywisty czas poświęcony to: 42h
   Data wykonania zadań 12.12.2023-16.01.2024r

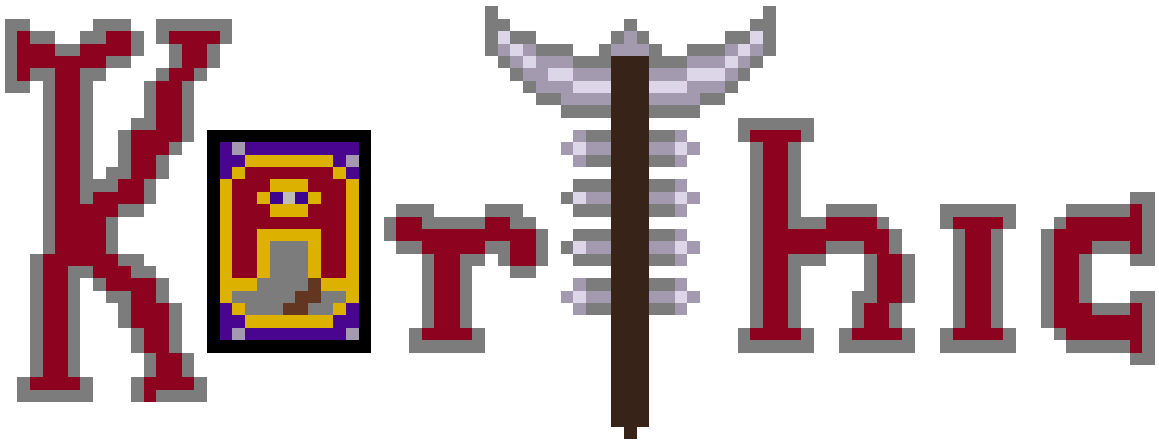
12 Podsumowanie zadań

12.0.1 Sprint 2.

Zostało wykonane następujące zadania

- Stworzone logo gry.

12.0.2 Logo gry



Rysunek 4: Logo gry

- Stworzenie assetu pierwszego przeciwnika wraz z animacją poklatkową.

12.0.3 Pierwszy przeciwnik

Rysunek 5: Pierwszy przeciwnik

- System zdrowia postaci graczy został aktywowany. Po utworzeniu węzła gracza przypisano mu skrypt zawierający zmienne HP, manę, tarczę oraz funkcje odpowiedzialne za przyjmowanie obrażeń i leczenie. Ponadto zmienna HP jest powiązana z paskiem postępu, który służy jako wskaźnik aktualnego poziomu zdrowia.

```
        extends CharacterBody2D

var max_hp = 100
var hp = max_hp
var shild = 0
var health_progress_bar
const SPEED = 300.0
const JUMP_VELOCITY = -400.0
var gravity = ProjectSettings.get_setting("physics/2d/default_gravity
    ")
var direction
func _ready():
  health_progress_bar = $ProgressBar
  var currentScene = get_tree().get_current_scene()
  if currentScene.get_name() =="world":
    canWalk = true
func _process(delta):

  health_progress_bar.max_value = max_hp
  health_progress_bar.value = hp
```

```
func take_damage(damage):
  hp -= damage
  if hp <=0:
    hp =0
    if hp <= 0:
      hp=0
func heal(amount):
  hp += amount
  if hp > max_hp:
    hp = max_hp
.

```

Listing 1: Kod odpowiedzialny za zdrowie

- Działający system chodzenia postaci. Funkcja odpowiadająca za chodzenie został dodany do skryptu węzła gracz.

```
        func _physics_process(delta):
  if canWalk:
    if not is_on_floor():
      velocity.y += gravity * delta

    direction = Input.get_axis("ui_left", "ui_right")
    if direction:
      velocity.x = direction * SPEED
    else:
      velocity.x = move_toward(velocity.x, 0, SPEED)
    move_and_slide()

```

Listing 2: Kod funkcji odpowiedzialnej za chodzenie

- Stworzenie szkieletu mapy.

12.0.4 Sprint 3

- ładowanie sceny walki, scena ładuje się w momencie wejścia w przeciwnika, assety wczytują się, button leave battle wprowadza w pętle, przez którą scena walki ładuję się cały czas.

- przygotowanie funkcji, która po wejściu w przeciwnika losuje 3 cyfry (potem będą to karty ataków) i wyświetlenie ich na ekranie (label)

- Menu główne oraz ekran porażki działają jako osobne sceny. Menu główne wyświetla się po uruchomieniu gry i umożliwia nam rozpoczęcie nowej rozgrywki, sprawdzenie tabeli wyników oraz wyjście z gry. Ekran porażki pojawia się w momencie, gdy zdrowie postaci spada do zera. Na tym ekranie, oprócz napisu "Game Over", znajdują się również dwa przyciski, które pozwalają zresetować grę do pierwszej planszy oraz wrócić do menu głównego. W przyszłości zostanie dodana informacja o liczbie zdobytych punktów podczas rozgrywki.

Rysunek 6: Menu główne

Rysunek 7: Ekran porażki

- dodanie grafik ataku i otrzymywania obrażeń pierwszego przeciwnika, niezaimplementowane jeszcze w grze.
-

12.0.5 Sprint 4

- stworzenie bazy danych z kartami oraz stworzenie assetów kart, które będą losowane podczas walki z przeciwnikami
- Przygotowanie bazy danych w języku SQL, nauka i testy wykorzystywania zapytań SQL w środowisku Godot zakończyła się niepowodzeniem spowodowany złym dopasowaniem technologii opartej na SQLite.
- przygotowania pola na rzucane karty w scenie walki.Przygotowanie rzucania kart z ogólną mechaniką (karta rzucana nie zadaje obrażeń/nie ma podpiętego efektu)

12.0.6 sprint 5

- Dobieranie decku kart, karty bez podpiętej grafiki. Problem z losowym dobieraniem.

```
      func _ready():
  CardActions.connect("card_played", was_i_played)
  id = randi() % 999999
  texture = card_texture

func _get_drag_data(_at_position):
  var data = {}
  data["id"] = id
  data["texture"] = card_texture

  var drag_preview = duplicate()
  drag_preview.modulate.a = .5

  set_drag_preview(drag_preview)

  return data


func was_i_played(card_id):
  if card_id == id:
    queue_free()

```

Listing 3: Kod przedstawiajacy mechanike kart

- połączenie bazy danych z aplikacją nie powiódł się z powodów braków kompatybilności, rozwiązaniem jest przejście na PostgreSQL wraz z Dockerem do symulacji serwera.
- Stworzenie spirtu bohatera i wzoru kart. Karta zawiera miejsce na koszt many oraz opis działania danej karty.

Rysunek 8: Sprite postaci gracza

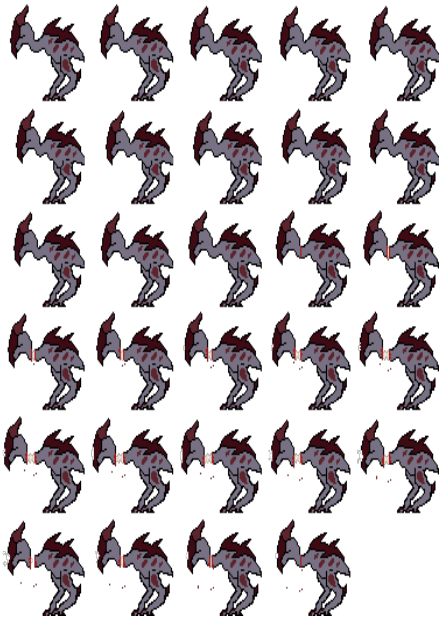
Rysunek 9: Wzór karty

12.0.7 Sprint 6

1. Przeciwnik "Ścierwojad"dostał w pełną animację ataku jak i animację przyjęcia obrażeń od gracza. Animacja ataku składa się z 28 klatek animacyjnych podobnie jak animacją podstawowa kiedy nie wykonuje ruchu, zapewnia to płyną i w miare spujną całość przedstawienia przeciwnika gdyż jego animację są mniejwięcje tej samej długości.

2. W secnie walki zostały dodane karty, poprawiono mechanike dropzone. Mechanika kart jest do poprawy i obecnie nie dziala.

3. Dodany sytem walki turowej "na sucho", po porawieniu mechaniki kart przetestujemy to rozwiązanie.



Rysunek 10: Ścierowjad atak

Animacja jak widać zawiera nie typowy sposób ataku, jest to spowodowane tym iż podczas produkcji przeciwnika zmienił się jego koncpet i został on przedstawiony jak "żywy trup"i w ten sposób postanowiliśmy to przedstawić.

4. Animacja optrzymywania obrażeń została wykonana w podobny sposób jak animacją podstawowa bez dodania motywów "zombie"aby unikatowość atak tej postaci nabrała charakteru unikatowości czy też wywołała zaksoczenie u przyszłych graczy

Rysunek 11: Scierwojad dmg

5. W naszej grze pojawił się nowy rodzaj przeciwnika, którego nazwaliśmy "krwiopijcami". Są to małe, potencjalnie niebezpieczne stworzenia podobne do ważek Aby wprowadzić je do rozgrywki, zaimplementowaliśmy kilka elementów, takich jak zdrowie,animacja postojowa, system otrzymywania obrażeń, oraz kolizje z graczem.

Rysunek 12: Pojedyńczy krwiopijca

```
        extends CharacterBody2D
@onready var sprite = $AnimatedSprite2D
@onready var sprite1 = $AnimatedSprite2D2
@onready var sprite2 = $AnimatedSprite2D3
enum EnemyType { Type1, Type2, Type3 }
var max_hp = 100
var hp = max_hp
var is_in_combat : bool = false
var attack_values : Array=[10,25,50]
var player : CharacterBody2D
func _ready():
  hp = max_hp
  sprite.play("default")
  sprite1.play("default")
  sprite2.play("default")
func _process(delta):
  sprite.play("default")
  sprite1.play("default")
  sprite2.play("default")
func _on_detection_area_body_entered(body):
  if body is CharacterBody2D and not is_in_combat:
    is_in_combat = true
    print("Combat started with enemy of type", EnemyType)
    start_combat()
func _input(event):
  if event is InputEventMouseButton and event.pressed:
    var mouse_pos = get_global_mouse_position()
    var entities = get_tree().get_nodes_in_group("enemies")
    for entity in entities:
      if entity.global_position.distance_to(mouse_pos) < entity.
   texture.get_width() / 2:
        is_in_combat = true
        print("Combat started with enemy of type", EnemyType)
        start_combat()
func start_combat():
  while is_in_combat and hp > 0:
    await get_tree().create_timer(15.0).timeout
func take_damage(damage):
  hp -= damage
  if hp <= 0:
    hp = 0
    is_in_combat = false
    queue_free()

```

Listing 4: Kod przedstawiajacy grupe przeciwników krwiopijców

6. Bieżące poprawy kodu i poprawa grafiki kart — Piotr Kociołek 4h
7. Wprowadzenie systemu walki turowej—Paweł Myjak-Zasadni 12h
8. Rozwój dokumentacji — Wszyscy 4h

12.0.8 Sprint 7

1. Dodaliśmy efekt parallaxu do tła w świecie gry.Zablokowaliśmy lewy krawędzie mapy za pomocą klifu, co sprawia, że niedasie wypaść poza mapę w fazie eksploracji. Zmieniliśmy również system kamery - teraz kamera jest zawsze wierna postaci gracza podczas fazy eksploracji, a podczas etapu walki, została sztywnie osadzą.

2. Przygotowaliśmy poruszanie się kart. Interakcja z dropzone nastąpi po podpięciu bazy danych. Obecnie występuje problem, że podnoszą się wszystkie karty jednocześnie.

```
    extends Node2D

var is_dragging = false
var drag_offset = Vector2()

func _process(delta):
  if is_dragging:
    var mouse_pos = get_global_mouse_position()
    set_global_position(mouse_pos - drag_offset)

func _input(event):
  if event is InputEventMouseButton:
    if event.button_index == MOUSE_BUTTON_LEFT:
      if event.is_pressed():
        start_drag(event.global_position)
      else:
        stop_drag()

func start_drag(mouse_position):
  is_dragging = true
  drag_offset = mouse_position - global_position

func stop_drag():
  is_dragging = false
  drag_offset = Vector2()


```

Listing 5: Kod pozwalający na poruszanie kartami po scenie

3. dodano grunt, gdzie porusza się postać oraz gdzie jest przeciwnik rozwój mapy gdzie w przyszłości będą kolejni przeciwnicy

4. Dodano nowy asset protagonisty, na razie statyczny zostanie zamieniony animacją poruszania.

Rysunek 13: Protagonista

12.1 Sprite 8

1. Zliczenie punktów i wpisywanie ich do bazy danych nie zostało wykonane. Powodem jest złe zarządzanie czasem osoby wykonującej zadanie (Jan Stachura), konsekwencje są przewidziane.

2. Stożenie wzoru kart, oraz przypisanie ich parametrów.

Rysunek 14: karty

```
1               extends Node2D
2
3 class_name Card
4 var damage : int
5 var shield : int
6 var cost : int
7 var card_texture : Texture
8
9 func _init(damage, shield, cost, texture_path):
10     self.damage = damage
11     self.shield = shield
12     self.cost = cost
13     card_texture = preload(texture_path)
14
15 func _ready():
16     var sprite = Sprite.new()
17     sprite.texture = card_texture
18     add_child(sprite)
19
```

Listing 6: Kod zawierający klase Card

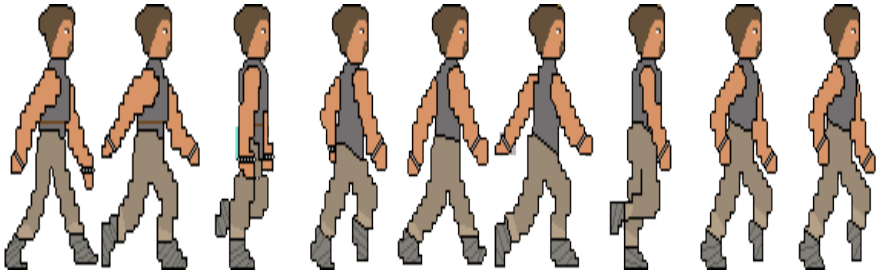
3. aktualizacja skryptów kart. dodanie obrażeń i efektów nakładanych na gracza.
4. nieudana próba podpięcia bazy danych.
5. aktualizacja problemów z kartami, dropzone działa kiedy wszystkie karty mają jedno id. Kiedy karty mają unikatowe id gra crash'uje się po wrzuceniu kart w dropzone.
6. dodano skrypt odpowiadający za randomizację kart w ręce (losowanie po id kart z tablicy)

12.2 Sprite 9

1. Dodanie pola walki.

Rysunek 15: tlo

2. Animacja hodzenia bohatera.



Rysunek 16: gracz-move

3. Karta o nazwie "Kula ognia"zadaje teraz obrażenia rozłożonew w czasie, Karta "Tarcza"daje nam oborone która najpierw musi być zniszczona przed zadaniem nam obrażeń.

12.3 Sprite 10

1. Dodanie animacji uderzenia, podpalenia i zatrucia.

Rysunek 17: fier-dmg

2.

Rysunek 18: poison-icon

3. Poprawienie umiejscowienia postaci wroga na mapie co usunęło błąd z ich pojawieniem się w ziemi.

4. Usunięcie "obcych"pikseli z już gotowych animacji.

12.4 Sprite 11

1. Błąd z bazą danych spowodowanym zaimplementowaniem niekompatybilnej wersji narzędzia do wersji Godot 4.1.

2. Brak ładowania się skryptu głównego z danymi globalnymi, Spowodowanych z powodu aktualizacji Godota. Naprawa problemu poprzez dodanie statycznego uruchamiania się skryptu.

3. Brak zapisu przypisania klawiszy myszy do akcji używania kart przyczyna nie znana występowała tylko u 2 osób.

4. Niekompatybilność funkcji macierzystych pomiędzy wersjami Godot 3.5 a Godot 4.1, błąd występuje w dokumentacji Programu

5. Aktualizacja projektu po wersji 3.5 powodowała błąd "Brak projektu w bazie plików". Powodem była do piska aktualizacji wersji Godota.

Spis tabel

Spis rysunków

Listings

Literatura

[1] Dokumentacji Godota: https://docs.godotengine.org/en/stable/

[2] GitHub użytkownika Marzin: https://github.com/Marzin-bot/PostgreSQLClient

[3] Dokumentacja Dockera: https://docs.docker.com

[4] Dokumentacja PostgreSQL: https://www.postgresql.org/docs/

[5] Strona z darmowymi assetami https://itch.io/game-assets/free/tag-pixel-art