The PyHero

Szymon Kublin

Streszczenie

Dokumentacja do gry `The PyHero`, zrealizowanej jako projekt na koniec przedmiotu `Programowanie 2 (Python)`. Zawiera opis rozgrywki, diagram UML, spis klas i metod, wykorzystane narzędzia oraz spis treści z zewnątrz.

1 Wstęp

1.1 Pomysł

Po początkowym odrzuceniu innych idei na projekt, a pozostaniu przy stworzeniu gry, dosyć szybko zdecydowałem się na pomysł stworzenia gry inspirowanej serią gier `Heroes of Might and Magic`. Są to strategiczne gry turowe w świecie fantasy. Taką też grą w założeniu miał stać się mój projekt.

Poza elementami strategicznymi oraz systemem turowym, `The PyHero` zawiera drobne elementy RPG (system umiejętności bohatera, doświadczenie i poziomy) i także jest (luźno, ponieważ nie zawiera fabuły) osadzony w świecie fantasy.

1.2 Rozgrywka

Celem gry jest dominacja nad światem (mapą).

Gracz, poruszając się po mapie może odwiedzać różne pola, zdobywając złoto, doświadczenie, czy dodatkowe ruchy. Po trafieniu razem z przeciwnikiem na to samo pole rozpoczynamy bitwę, w której bierze udział nasza armia, złożona z siedmiu rodzajów wojowników. Kolejne jednostki atakują się nawzajem, aż do momentu pokonania wszystkich wojsk jednej ze stron lub ucieczki jednego z bohaterów.

Gracz może rozwijać swoje umiejętności, zdobywając wyższe poziomy, a także kupować wojowników do swojej armii.

Więcej szczegółów dotyczących rozgrywki można znaleźć w rozdziale System gry.

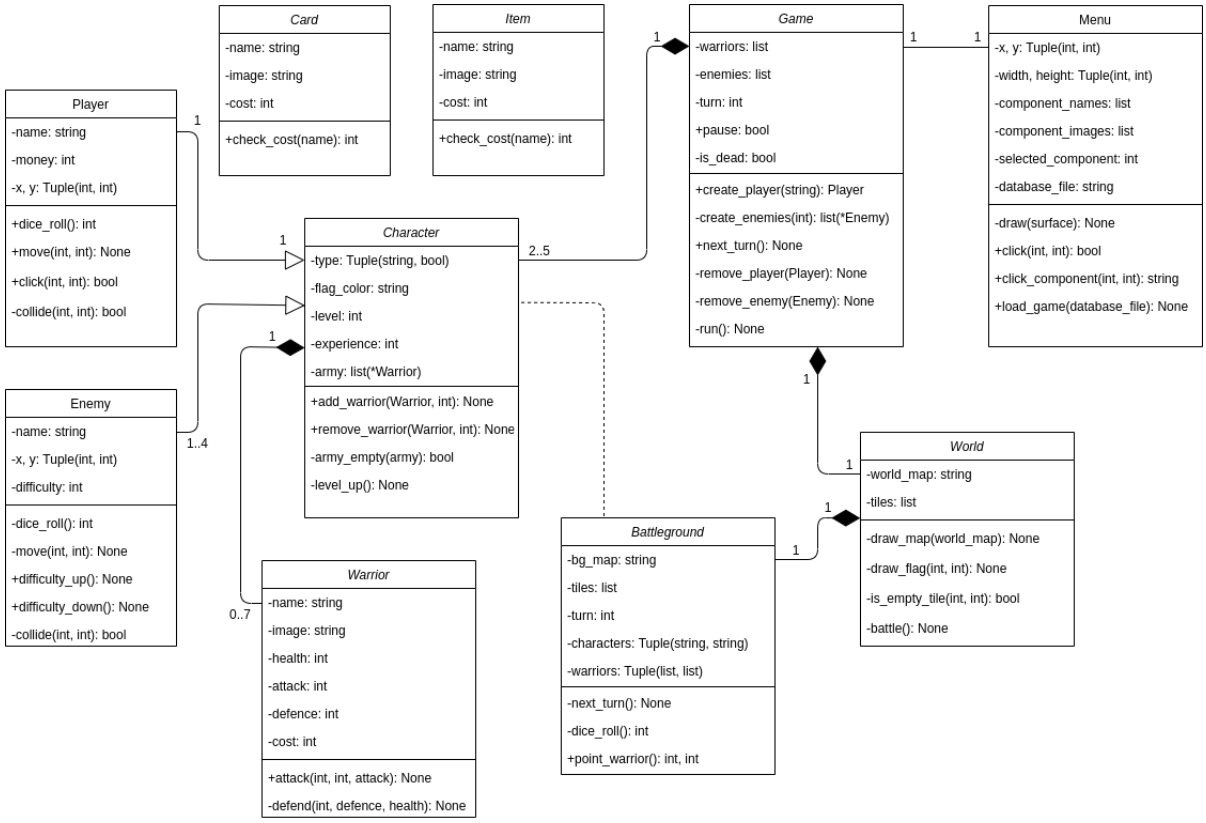
2 Założenia

2.1 Diagram UML

W tym podrozdziale możemy porównać diagramy UML, pierwotny z aktualnym.

Najpierw zobaczmy pierwszy diagram, który powstał przed tworzeniem gry, a następnie diagram przedstawiający obecną sytuację.

2.1.1 Pierwotny diagram

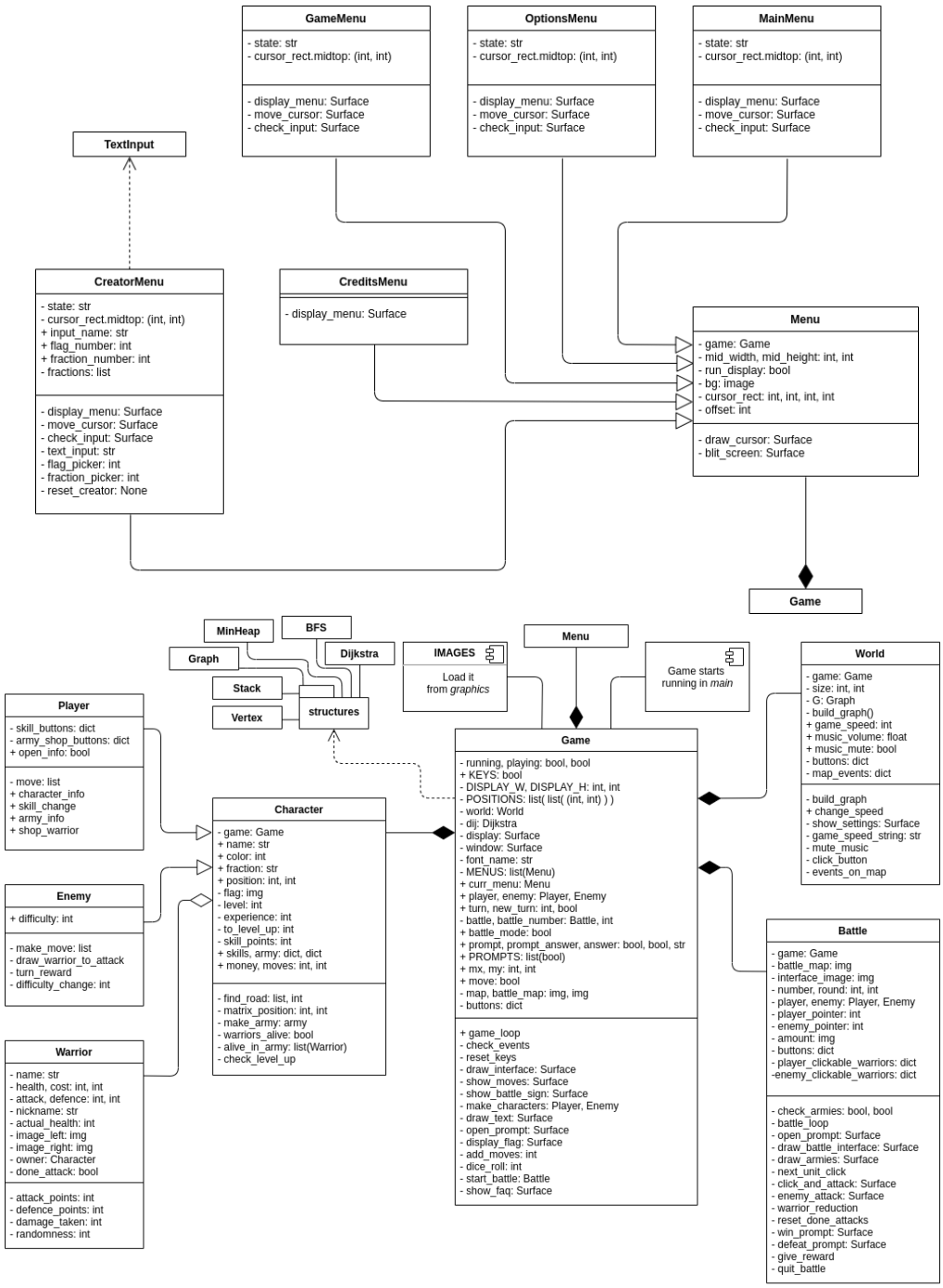


Rysunek 1: Pierwszy diagram UML projektu gry.

Powyższy diagram uwzględnia klasy `Card` oraz `Item`, które nie pojawiły się w ostatecznej wersji (nie zostały zaimplementowane), a także klasę `Battleground`, która miała służyć do tworzenia bitwy, a obecnie jest zastąpiona klasą `Battle`. Pozostałe klasy, przynajmniej w głównych założeniach i celach są w niezmienionej formie.

2.1.2 Aktualny diagram

Poniższy diagram uwzględnia wszystkie zmiany, które zaszły w trakcie tworzenia projektu oraz został podzielony na dwie sekcje (ze względu na wielkość).



Rysunek 2: Górny diagram, to pierwsza część (menu gry), dolny to część rozgrywki.

2.1.3 Porównanie

Przede wszystkim dostrzegalną różnicą jest wielkość obu diagramów. Z powodu rozrostu całego projektu obecny diagram także stał się dużo większy niż początkowo. Dużą zmianą jest rozłożenie klasy `Menu` na podklasy, które tworzą osobne okna w menu gry. W części związanej z rozgrywką uwzględniłem obrazki (`IMAGES`), ładowane w grze (nie tylko w klasie `Game`, na co może wskazywać diagram), struktury stworzone na wykładzie, a także informację o uruchamianiu gry przez skrypt `main.py`. Pozostałe klasy zostały rozbudowane, bez zmian w głównych założeniach (szczegóły w kolejnym rozdziale).

2.2 Budowa

W tej sekcji można znaleźć spis stworzonych klas, ich opis oraz zawartość (metody), a także pakiety zewnętrzne, które zostały użyte.

2.2.1 Pakiety

Ten projekt używa pakietów:

- `PyGame`
- `random` (*choice, randint*)
- `math` (*inf*)

A także kod z zewnątrz:

- `structures` (**Grzegorz Jagiella** - wykład)
- `TextInput` (**Silas Gyger**)

2.2.2 Main

```
    """
    This is the main loop to run the game.
    """
```

Nie jest to klasa, ale krótki skrypt (`main.py`). W nim uruchamiamy główną pętlę i rozpoczynamy program, tworząc obiekt klasy `Game`.

```
from game import Game

g = Game()

while g.running:
    g.curr_menu.display_menu()
    g.game_loop()
```

2.2.3 Klasa Game

```
1     """
2     This class handle the game, display screen and make game loop.
3     """
```

Główna klasa, która przechowuje najważniejsze operacje, tworzy obiekty i rozgrywkę.

Składa się z metod:

- `game_loop` - główna pętla gry, prowadzi rozgrywkę
- `check_events` - sprawdza strumienie wejścia (klawiatura i mysz)
- `reset_keys` - wraca klawisze do stanu pierwotnego
- `draw_interface` - wyświetla mapę i interfejs (przyciski, flagi)
- `show_moves` - przesuwa flagę na mapie do wybranego celu
- `show_battle_sign` - dźwięk fanfar i wyświetlenie znaku bitwy
- `make_characters` - tworzy postać gracza i wroga z bazowych danych
- `draw_text` - wyświetla tekst na ekranie
- `open_prompt` - otwiera okno z podanym tekstem i opcją TAK/NIE
- `display_flag` - wyświetla flagę na pozycji bohatera
- `add_moves` - dodaje losową ilość ruchów
- `dice_roll` - symuluje rzut kostką (statyczna)
- `start_battle` - tworzy i rozpoczyna nową bitwę
- `show_faq` - wyświetla okno z informacjami o grze

2.2.4 Klasa Menu

```
1     """
2     This class creates basic menu for subclasses.
3     """
```

Klasa, rozbudowana przez podklasy, razem tworząc system menu gry.

Składa się z metod:

- `draw_cursor` - rysuje kursor na ekranie
- `blit_screen` - wyświetla aktualny ekran i odświeża widok

A jej podklasy to:

- `MainMenu` - odpowiada za okno menu startowego (głównego)
- `GameMenu` - okno menu gry (nowa gra, wczytanie gry - bez implementacji)
- `CreatorMenu` - odpowiada za okno tworzenia postaci
- `OptionsMenu` - okno ustawień (nie zaimplementowane, ustawienia w grze)
- `CreditsMenu` - wyświetla twórcę i pochodzenie materiałów

Powyższe składają się z metod (zmodyfikowanych w zależności od podklasy):

- `display_menu` - wyświetla aktualne okno menu - każda podklasa
- `move_cursor` - zmienia położenie kursora - każda, poza `CreditsMenu`
- `check_input` - sprawdza, która opcja została wybrana - każda, poza `CreditsMenu`
- `text_input` - pozwala na wpisanie imienia gracza (statyczna) - tylko `CreatorMenu`
- `flag_picker` - służy do wyboru flagi gracza - tylko `CreatorMenu`
- `fraction_picker` - pozwala wybrać frakcję gracza - tylko `CreatorMenu`
- `reset_creator` - resetuje stan kreatora postaci - tylko `CreatorMenu`

2.2.5 Klasa Character

```
1     """
2     This class creates basic hero, with attributes and methods for both
      Player and Enemy.
3     """
```

Główna klasa, która pozwala na tworzenie postaci w grze.

Jej podklasami są `Player` oraz `Enemy`.

Zbudowana jest z metod:

- `find_road` - przygotowuje graf do algorytmu Dijsktry i wyznacza najlepszą drogę
- `matrix_position` - oblicza pozycję postaci z podanych współrzędnych (statyczna)
- `make_army` - tworzy armię postaci, wypełnioną odpowiednimi ilościami wojowników
- `warriors_alive` - sprawdza, czy wojowników z podanym indeksem żyje (ilość > 0)
- `alive_in_army` - zwraca listę z żyjącymi wojownikami (statyczna)
- `check_level_up` - sprawdza, czy postać powinna awansować na wyższy poziom

Metody dla podklasy `Player`:

- `move` - ogarnia operację ruchu gracza (także wyświetla okna), zwraca drogę
- `character_info` - wyświetla kartę postaci i tworzy przyciski umiejętności
- `skill_change` - pozwala na zmianę umiejętności gracza
- `army_info` - wyświetla okno ze stanem wojska (karta z armią)
- `shop_warrior` - pozwala kupować i sprzedawać wojowników

Metody dla podklasy `Enemy`:

- `make_move` - oblicza ruch przeciwnika, zwraca drogę
- `draw_warrior_to_attack` - losuje wojownika do zaatakowania (w bitwie)
- `turn_reward` - przyznaje przeciwnikowi doświadczenie, ruchy i wojsko (po turze)

2.2.6 Klasa Warrior

```
1     """
2     This class makes warriors and calculates damage.
3     """
```

Klasa ta tworzy wojowników, przechowuje ich statystyki oraz wylicza zadawane obrażenia.

Zbudowana jest z metod:

- `attack_points` - oblicza punkty zadawane w ataku przez wojownika
- `defence_points` - oblicza punkty zatrzymywane w obronie przez wojownika
- `damage_taken` - wylicza zadane obrażenia wobec innego wojownika
- `randomness` - wprowadza element losowości do obliczeń

2.2.7 Klasa World

```
1     """
2     This class helps in world configuration (graph) and makes settings.
3     """
```

Odpowiada za stworzenie grafu dla mapy gry, przechowuje ustawienia (szybkość, głośność) i pozwala na ich zmiany.

Zawiera metody:

- `build_graph` - tworzy graf reprezentujący mapę gry
- `change_speed` - pozwala na zmianę szybkości gry
- `show_settings` - wyświetla ustawienia gry, tworzy przyciski

- `game_speed_string` - zamienia wartość szybkości na tekst
- `mute_music` - wyłącza/włącza muzykę
- `click_button` - obsługuje przyciski, korzystając z podanego
- `events_on_map` - przyznaje nagrody w zależności od wydarzenia na mapie

2.2.8 Klasa Battle

```
1     """
2     This class makes battle loop and all operations in battle.
3     """
```

Obsługuje bitwę i operacje wykonywane w trakcie (atak, wyświetlanie informacji).

Zbudowana jest z metod:

- `battle_loop` - główna metoda bitwy, obsługuje resztę
- `open_prompt` - otwiera okno z tekstem i przyciskami TAK/NIE
- `draw_battle_interface` - wyświetla mapę, interfejs i jednostki
- `draw_armies` - wyświetla wojowników (w `draw_battle_interface`)
- `next_unit_click` - dokonuje zmian po wciśnięciu przycisku kolejnej jednostki
- `click_and_attack` - wyświetla obrażenia, dokonuje zmian w wojownikach
- `enemy_attack` - obsługuje atak wykonywany przez przeciwnika
- `warrior_reduction` - według obrażeń zmniejsza ilość wojowników (statyczna)
- `reset_done_attack` - resetuje stan ataku wszystkich jednostek
- `win_prompt` - wyświetla okno po wygranej bitwie
- `defeat_prompt` - wyświetla okno po przegranej bitwie
- `quit_battle` - pozwala uciec z pola bitwy
- `give_reward` - przyznaje nagrodę po wygranej bitwie

Po zakończeniu bitwy możemy wrócić do rozgrywki na mapie.

2.3 System gry

W tym rozdziale opiszę przebieg oraz sposób rozgrywki.

2.3.1 Menu

Po uruchomieniu `main.py` znajdujemy się w głównym menu (klasa `MainMenu`), możemy poruszać się klawiszami strzałek, do góry i w dół. Wybrania zaznaczonej opcji dokonujemy przez klawisz **enter**, a wrócić do poprzedniego okna możemy klawiszem **esc** lub wybierając opcję *Back*.

Wybierając ostatnią opcję zamkniemy grę – *Quit Game* (duh).

Po wybraniu opcji *Credits* przejdziemy do wyświetlenia twórcy i źródeł (klasa `CreditsMenu`).

Opcja *Options* zaprowadzi nas do okna ustawień (`OptionsMenu`), w którym docelowo możemy zmienić prędkość gry i poziom głośności (nie zaimplementowane).

Start Game prowadzi do menu gry (`GameMenu`). Mamy w nim do wyboru nową grę – *New Game* lub załadowanie gry – *Load Game* (obecnie niedostępne).

Po wybraniu *Start Game* znajdziemy się w oknie menu tworzenia postaci – `CreatorMenu`. Możemy w nim wpisać imię swojej postaci, wybrać kolor flagi i frakcję, a także rozpocząć rozgrywkę (klasa `Game` metoda `game_loop`).

2.3.2 Tworzenie

Zanim wyruszymy w podróż możemy stworzyć swoją postać w menu tworzenia (`CreatorMenu`).

Imię naszego bohatera może składać się z dowolnych znaków, ale nie może być puste, ani dłuższe niż 12 znaków.

Flagę możemy wybrać spośród dostępnych kolorów, korzystając ze strzałek LEWO/PRAWO.

Rysunek 3: Wszystkie dostępne flagi.

Na koniec wybieramy frakcję, korzystając ze strzałek jak przy wyborze flagi.

Dostępne frakcje, to `Demony`, `Elfy`, `Ludzie`, `Nieumarli`.

Każda z frakcji posiada unikalne grafiki wojowników, ale każdy posiada ich 7.

2.3.3 Poruszanie

Po stworzeniu postaci i wyruszeniu w podróż (opcja *Let's Go!*) pojawia się trochę większe okno, które będzie domem dla naszych przygód. Na górnej belce możemy znaleźć imię naszego bohater, jego poziom, doświadczenie i pieniądze, a także przyciski (karty postaci, karty armii, informacji oraz ustawień). Prawa strona tej belki to numer tury, dostępna ilość ruchów i przycisk przechodzący do kolejnej tury.

W tym oknie głównym urządzeniem jest dla nas mysz. Nasz bohater oraz przeciwnik znajdują się w wylosowanych miejscach, a poruszać się możemy naciskając wybraną płytkę na mapie. W zależności od potrzebnych i dostępnych ruchów może pojawić się okno z pytaniem, w którym możemy wybrać TAK/NIE lub skorzystać z klawiatury **enter**/**esc**.

Ważną uwagą jest możliwość trafienia na wydarzenie na mapie, więc lepiej wykonywać niewielkie podróże (droga wyznaczana jest automatycznie za pomocą algorytmu Dijkstry).

Wydarzenia na mapie są jednorazowe na płytkach:

- `Kopalnie` i `ruiny domostw` - dodatkowe złoto dla bohatera
- `Obeliski` - dodawane jest dodatkowe doświadczenie
- `Wieże strażnicze` - możliwość zwiększenia poziomu losowej umiejętności
- `Lasy` i `góry` - otrzymujemy dodatkowe ruchy do wykorzystania

2.3.4 Rozwój

Na dobry początek przygód otrzymujemy niewielkie oddziały wojowników, które możemy powiększyć w sklepie (także sprzedać wybranych i zyskać pieniądze na inne zakupy).

Stan początkowy naszej skrzyni ze złotem to całe 200 sztabek, natomiast doświadczenie naszego bohatera wynosi 20 punktów, gdzie do awansu na kolejny poziom potrzebujemy zdobyć 100 punktów doświadczenia (zawsze tyle samo).

Kolejnym darem przygotowującym do wyruszenia jest 10 punktów umiejętności do wydania wedle uznania!

Stan początkowy to poziom 5 każdej z umiejętności:

- `Strength` - siła, odpowiada za moc ataku naszych wojowników
- `Stamina` - wytrzymałość, zwiększa obronę wojowników i ilość dodawanych ruchów
- `Wisdom` - mądrość, zwiększa ilość zdobywanych ruchów i punktów doświadczenia
- `Intellect` - inteligencja, zwiększa moc ataku i zdobywanie punktów doświadczenia
- `Agility` - zręczność, zwiększa moc obrony
- `Charm` - urok, zwiększa ilość zdobywanego doświadczenia i złota
- `Fortune` - szczęście, wpływa w niedużym stopniu na wszystko

2.3.5 Walka

Gdy spotkasz się z przeciwnikiem na tym samym polu dojdzie do starcia!

Na przebieg bitwy mają wpływ umiejętności bohatera, atrybuty poszczególnych wojowników, ich ilość i odrobina szczęścia.

Każda bitwa rozpoczyna się od ruchu gracza i wojownika z indeksem 1. Aby zaatakować musisz kliknąć LPM na wrogiego wojownika, po czym zobaczysz na krótki czas (w zależności od ustawionej prędkości) wartość zadanych obrażeń. Jeśli obrażenia będą duże uda się pokonać kilku wojowników z grupy! Po wykonaniu ataku możesz nacisnąć przycisk kolejnej jednostki – `NEXT UNIT`. Gdy wykonasz atak każdą z jednostek, nadejdzie kolej na ruch przeciwnika, który następuje automatycznie, ale pozwala oglądać zadawane obrażenia. Po atakach przeciwnych wojowników rozpocznie się kolejna runda.

Jeśli wygrasz otrzymasz sławę, złoto, doświadczenie i... świeżego przeciwnika!

Jeśli jednak przegrasz, stracisz wszystko...

Po przegranej bitwie możesz wybrać, czy zakończyć grę, czy rozpocząć rozgrywkę od nowa.

2.3.6 Zakończenie

W obecnym stanie gra nie posiada implementacji wygranej rozgrywki.

2.4 Instalacja

Wszystkie pliki dostępne są do pobrania tutaj.

Na początek wystarczy pobrać archiwum z plikami lub sklonować repozytorium.

Oczywiście musimy zacząć od sprawdzenia/zainstalowania wersji `Pythona` (najlepiej `3.9`).

Poza pakietem `PyGame` musimy posiadać pakiety zawarte na liście `requirements.txt`.

Po otworzeniu terminala w katalogu z plikami gry wystarczy wpisać **python main.py** i cieszyć się grą!

Uwaga! Gra może nie wyświetlać się odpowiednio na systemie innym niż wybrana dystrybucja `Linuxa`, a także mogą wystąpić problemy z dźwiękiem. W systemie `Windows` można rozwiązać problem odtwarzania dźwieku przez uprzednie wpisanie w wierszu poleceń komendy **set SDL_AUDIODRIVER=directsound**.

3 Wykonanie

Postaram się w tym rozdziale przybliżyć sposób pracy nad projektem.

3.1 Narzędzia

Podstawowy warsztat składa się z:

- `Ubuntu 21.04`
- `Python 3.9`
- `PyCharm Community Edition`
- `drawio` - diagramy UML
- `GitHub` - system kontroli wersji – `Git`
- `GIMP` - edycja grafik, tworzenie szablonów
- `Todoist` - zarządzanie zadaniami, ramy czasowe

3.2 Planowanie

Prace nad projektem rozbiłem na 5 bloków tygodniowych, starając się wykonywać zaplanowane elementy w danym tygodniu, co piątek kontrolując postępy i otrzymując uwagi od pani **Katarzyny Giniewicz**.

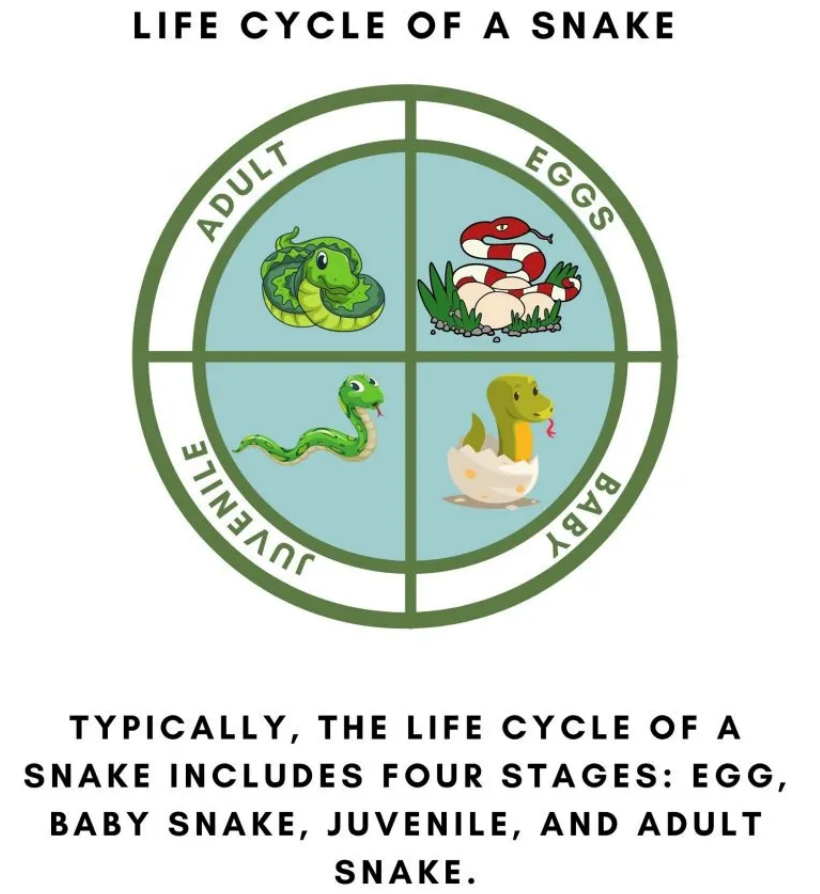
3.3 Wersje

Korzystając z kontroli wersji oraz platformy `GitHub` mogłem tworzyć zapisy kolejnych wersji i stąd do tej pory powstało ich sześć (z obecną):

- `eggPy.one (v0.1)` - 15.05
- `eggPy.two (v0.2)` - 28.05
- `eggPy.three (v0.3)` - 04.06
- `babyPy.four (v0.4)` - 15.06
- `babyPy.five (v0.5)` - 24.06
- `babyPy.six (v0.6)` - 25.06
- `juvenilePy.seven (v0.7)` - nadchodzi
- `juvenilePy.eight (v0.8)` - nadchodzi
- `juvenilePy.nine (v0.9)` - nadchodzi
- `adultPy.zero (v1.0)` - nadchodzi (pełna wersja)

Każda z powyższych wersji dostępna jest do pobrania w repozytorium gry.

Nazewnictwo wersji inspirowane jest fazami rozwoju węży (*egg - baby - juvenile - adult*) oraz proponowanym poprawnym nazewnictwem wersji oprogramowania (`v0.1`).



Rysunek 4: Obrazek ilustrujący rozwój węży, znaleziony tutaj.

3.4 Źródła

Grafiki użyte w grze zostały zakupione na stronie CraftPix.

Muzyka wykorzystana w projekcie pochodzi ze strony **Free Music Archive**, wykonywana przez *Pictures of the Floating World* oraz *Dee Yan-Key*, na licencji zawartej tutaj.

Dźwięk fanfar pochodzi ze strony **MixKit**, na licencji zawartej tutaj.

Czcionka Fredericka the Great pobrana z **Google Fonts**, na licencji zawartej tutaj.

Skrypt do wpisywania tekstu jest autorstwa **Nearoo**, znajdujący się tutaj.

Wykorzystywane struktury (`structures`) pochodzą z wykładu (Grzegorz Jagiella).