Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 4

Pokemon™ Battler

---

autor	Michał Rabsztyn
prowadzący	dr inż. Anna Gorawska
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	SSI
semestr	4
termin laboratorium	środa, 13:00 – 15:00
sekcja	1A
termin oddania sprawozdania	2021-05-23

---

1 Treść zadania

Napisać program będący symulacją komponentu walki z serii gier Pokemon w formule endless (Battle Tower). Gra polega na prowadzeniu pojedynków z symulowanym przeciwnikiem. Każdy z uczestników ma 6 Pokemonów, runda trwa póki wszystkie Pokemony oponenta nie będą niezdolne do walki - wygrana - lub dopóki wszystkie Pokemony gracza nie będą niezdolne do walki - przegrana.

Program składa się z 24 autorskich klas i wykorzystuje 4 tematy z laboratorium - mechanizm wyjątków, kontenery STL, inteligentne wskaźniki oraz wyrażenia regularne <regex>.

2 Analiza zadania

2.1 Rozszerzony opis działania

Użytkownik rozpoczyna grę w głównym menu, gdzie może rozpocząć nową rozgrywkę lub kontynuować poprzednią, pod warunkiem, że plik zapisu istnieje i zawiera poprawne dane. Wybranie opcji “New Game” otwiera okno wyboru drużyny. Po wybraniu sześciu Pokemonów i potwierdzeniu swojego wyboru pokazywany jest ekran wyboru pierwszego Pokemona do walki, a następnie właściwa scena pojedynku. Gracz ma do wyboru zaatakować (Fight), zmienić Pokemona (Pokemon), wyleczyć aktualnie walczącego Pokemona o 50 hp (Heal), bądź przejść do menu opcji (Options) oferujące możliwość zapisania gry. Każda z akcji ma swój priorytet opisany w punkcie *4.3*. Atak powoduje obniżenie punktów życia Pokemona oponenta. Jeżeli wartość obrażeń jest tak niska, że wyniosłaby 0 podciągana jest do 1, chyba, że zerowe obrażenia wynikają ze złożenia typów. Jeżeli poziom życia spadnie do 0 musi zostać wystawiony do walki kolejny Pokemon. Jeżeli nie ma więcej kończy się runda. Wygrana oznacza podniesienie poziomu postaci, a więc i wszystkich Pokemonów oraz wykonanie autozapisu. Jeżeli poziom ewolucji jest zgodny z aktualnym poziomem Pokemona, przechodzi on ewolucję na początku kolejnej rundy, a więc zmienia się jego nazwa, statystyki, wygląd, może zmienić się też typ i zestaw ruchów. Z poziomu ekranu wygranej bądź przegranej gracz może kontynuować rozgrywkę (Continue), co w przypadku wygranej oznacza wylosowanie drużyny dla przeciwnika lub w przypadku przegranej przywrócenie stanu obu drużyn do stanu z początku rundy.

2.2 Wybór bibliotek

Do oprawy graficznej wykorzystana została biblioteka SFML ze względu na prostotę obsługi i działania.

2.3 Uwagi dodatkowe

Grafiki poszczególnych Pokemonów narysowane przez r/JordieBo https://i.imgur.com/4QgFYas.png

Ekran menu głównego wykorzystuje oryginalne logo stworzone przez Game Freak - Satoshi Tajiri Ken Sugimori Junichi Masuda

3 Specyfikacja zewnętrzna

3.1 Instrukcja dla użytkownika

3.1.1 Sterowanie

Po wszystkich menu gracz porusza się przy pomocy strzałek. Aktualnie obsługiwane pole oznaczone jest czarną ramką.

Zatwierdzanie odbywa się za pomocą klawisza Enter.

Klawiszem Esc lub krzyżykiem można zamknąć aplikację. Nie jest wtedy jednak wykonywany zapis.

3.1.2 Zależności typów

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	1	1	1	0.5	2	2	1	0.5	0.5	1	1	1	1	2	1	1
2	2	0.5	1	1	2	2	1	1	0.5	1	1	1	1	1	0	1	1	1
3	1	1	2	0.5	2	1	0.5	1	1	0.5	1	2	0	1	1	1	1	0.5
4	1	1	1	0.5	1	1	1	0.5	1	1	2	1	1	1	1	1	0.5	1
5	0.5	.05	0	1	1	0.5	1	1	1	1	1	1	1	2	1	1	2	1
6	0.5	0.5	1	1	2	1	1	2	1	1	1	1	1	1	2	0.5	1	1
7	0.5	1	1	1	0.5	1	0.5	1	1	0.5	2	0.5	1	1	1	2	0.5	2
8	0.5	1	1	2	1	0.5	1	1	1	0.5	0	2	1	1	1	2	1	1
9	0.5	2	1	1	1	0	1	1	2	1	1	1	0	0.5	1	1	1	1
10	2	1	1	0.5	1	1	2	2	1	0.5	0.5	2	1	2	1	1	1	0.5
11	1	1	1	0	1	1	1	1	1	2	1	2	1	0.5	1	0.5	1	2
12	1	1	1	1	1	2	2	1	1	1	1	0.5	1	1	1	2	2	1
13	1	1	1	1	1	2	1	1	0	1	1	1	1	1	1	1	1	1
14	0.5	1	1	1	0.5	0.5	1	1	1	0.5	2	1	1	0.5	2	1	1	1
15	2	2	1	1	1	0.5	1	1	2	1	1	1	1	1	0.5	1	1	1
16	1	1	1	1	1	2	0.5	0.5	1	2	2	1	0.5	0.5	1	1	2	2
17	0.5	1	0.5	1	0.5	2	2	0.5	1	0.5	2	0.5	0.5	0	0.5	0.5	0.5	1
18	1	1	1	2	1	1	0.5	1	1	2	1	0.5	0	1	1	1	0.5	0.5

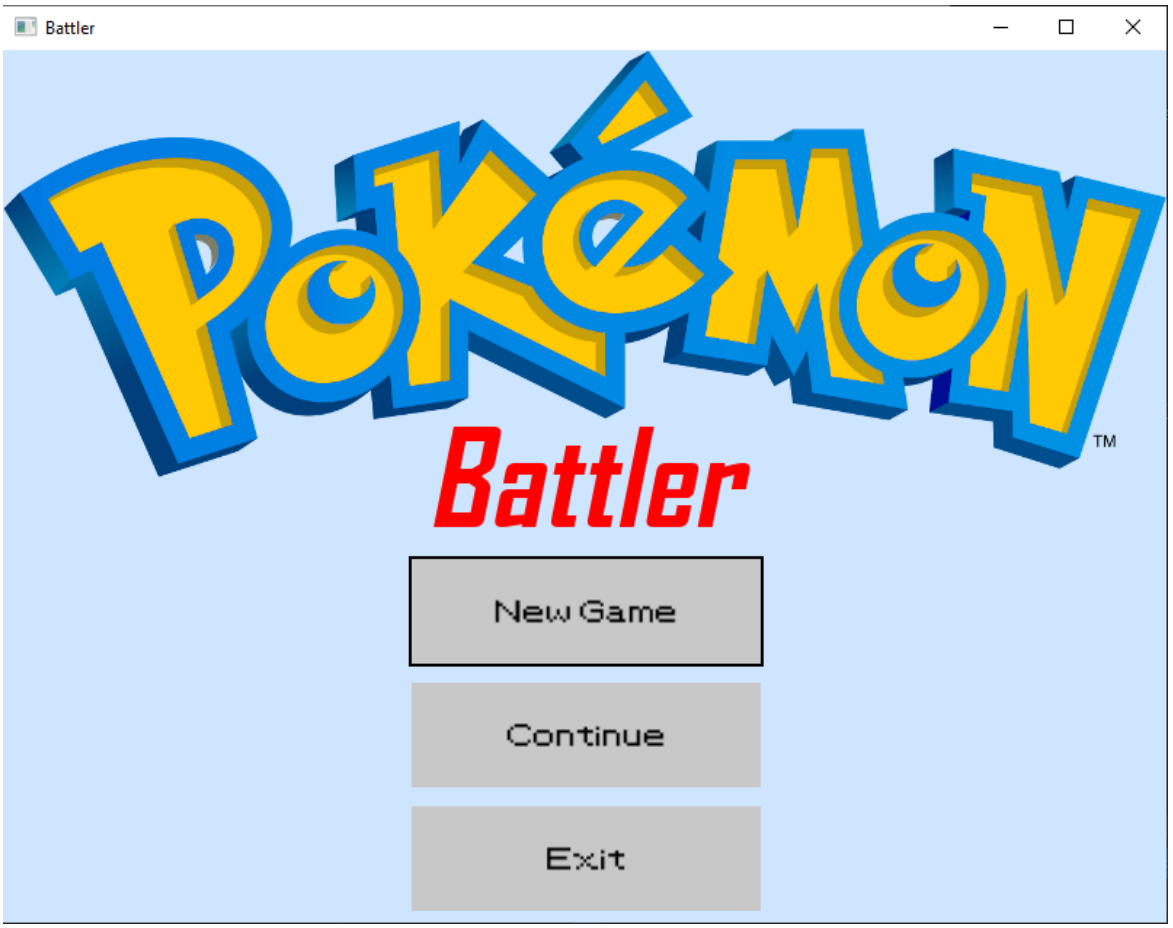
Powyższa tabela prezentuje zależności obrażeń pomiędzy poszczególnymi typami np. jeżeli typ [10] trawiasty zaatakuje typ [7] ognisty to mnożnikiem obrażeń będzie 0.5.

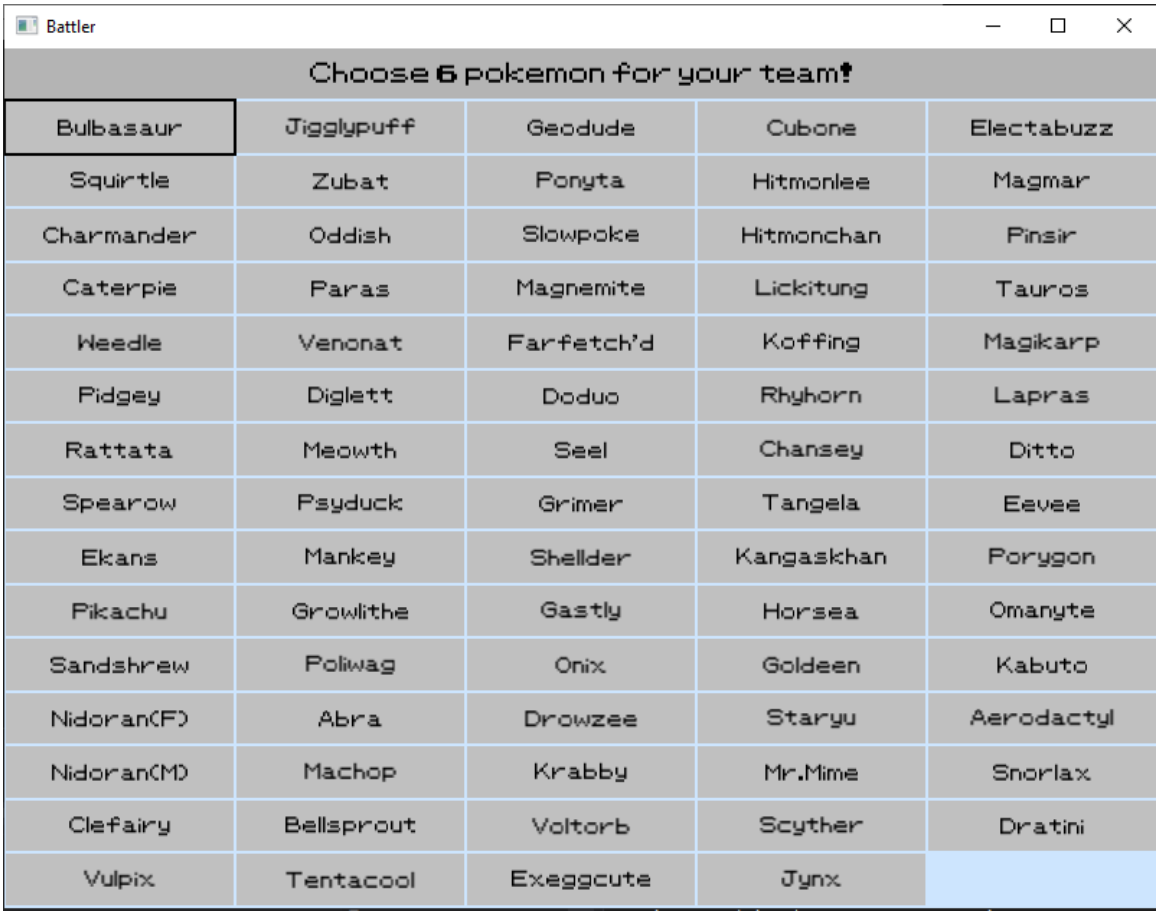
Numeracja typów:

[1] Bug	[10] Grass
[2] Dark	[11] Ground
[3] Dragon	[12] Ice
[4] Electric	[13] Normal
[5] Fairy	[14] Poison
[6] Fighting	[15] Psychic
[7] Fire	[16] Rock
[8] Flying	[17] Steel
[9] Ghost	[18] Water

3.1.3 Komunikaty obrażeń
"It's not very effective..." - modyfikator obrażeń to x0.25 lub x0.5
"It's super effective!" - modyfikator obrażeń to x2 lub x4
"It doesn't affect" - modyfikator obrażeń to x0

3.2 Prezentacja działania




Battler
Choose 6 pokemon for your team!
Bulbasaur
Jigglypuff
Geodude
Cubone
Electabuzz
Squirtle
Zubat
Ponyta
Hitmonlee
Magmar
Charmander
Oddish
Slowpoke
Hitmonchan
Pinsir
Caterpie
Paras
Magnemite
Lickitung
Tauros
Weedle
Venonat
Farfetch'd
Koffing
Magikarp
Pidgey
Diglett
Doduo
Rhyhorn
Lapras
Rattata
Meowth
Seel
Chansey
Ditto
Spearow
Psyduck
Grimer
Tangela
Eevee
Ekans
Mankey
Shellder
Kangaskhan
Porygon
Pikachu
Growlithe
Gastly
Horsea
Omanyte
Sandshrew
Poliwag
Onix
Goldeen
Kabuto
Nidoran(F)
Abra
Drowzee
Staryu
Aerodactyl
Nidoran(M)
Machop
Krabby
Mr.Mime
Snorlax
Clefairy
Bellsprout
Voltorb
Scyther
Dratini
Vulpix
Tentacool
Exeggcute
Jynx


Battler
Grimer
12/12
Bulbasaur
11/11
Megahorn 10/10
Leaf Storm 5/5
Tackle 35/35
Gunk Shot 5/5
Back

4 Specyfikacja wewnętrzna

4.1 Opis klas

4.1.1 klasy z *TextBox* w nazwie - prefaby bloczków z napisami. Posiadają kilka podstawowych funkcji dotyczących wyświetlania.

- TextBoxBase
- TextBoxSmall
- TextBoxMedium
- TextBoxLong

4.1.2 klasy z *PlayerMenu* w nazwie - zestawy bloków lub obiektów do wyrysowania. Zawierają podstawowe funkcje do obsługi poszczególnych zestawów. Na początku pełniły rolę scen.

- PlayerMenu
- PlayerMenuAction
- PlayerMenuBattleScene
- PlayerMenuMoves
- PlayerMenuOutput
- PlayerMenuYesNo

4.1.3 klasy z *Scene* w nazwie - właściwe sceny, między którymi przełącza się gracz. Posiadają schemat działania oparty na pollEvents - update - render: [obsłuż wydarzenia z klawiatury podane przez użytkownika - głównie poruszanie się po menu] - [zareaguj na zmiany spowodowane inputem] - [wyrysuj obiekty na ekranie]. “-1” jest standardowym sygnałem, że nic się nie wydarzyło.

- ChooseActionScene
- ChooseMoveScene
- ChoosePokemonScene
- MainMenuScene
- OptionsScene
- SetupScene
- WinLooseScene

4.1.4 Pozostałe klasy:

- App - klasa globalna. Posiada zmienne, które powinny być widoczne globalnie np. okno aplikacji. Zapisana jako inline w pliku App.h dzięki standardowi C++17.
- Enemy - klasa przeciwnika. Posiada funkcje potrzebne do funkcjonowania oponenta dla gracza takie jak losowanie ruchu czy wysłanie kolejnego Pokemona z listy.
- Entity - klasa-rodzic dla Player oraz Entity. Posiada elementy wspólne dla swoich klas-dzieci. Najważniejszy jest vector pokemonList reprezentujący Pokemony, które ma się aktualnie w posiadaniu. Zawiera również funkcje odpowiedzialne za pilnowanie i resetowanie różnego rodzaju zmiennych dotyczących więcej niż jednego Pokemona np. zbiorowe leczenie albo sprawdzanie czy któryś z Pokemonów może ewoluować.
- GameSystem - klasa systemu walki. Odpowiada za przełączanie pomiędzy scenami, obsługę algorytmu głównego (4.3) oraz kalkulowanie obrażeń i ich modyfikatorów.
- Move - klasa ruchu. Każdy Pokemon posiada cztery ruchy do wyboru. Każdy z tych ruchów ma opisujące go statystyki: nazwa, PP(Power Points = liczba użyć), celność, moc, status (ujęty w projektowaniu, pominięty w implementacji - do możliwego rozwinięcia), typ oraz rodzaj (fizyczny lub specjalny).
- Player - klasa gracza. Na jej podstawie tworzony jest obiekt gracza. Niewiele różni się od klasy Entity, więc służy głównie jako identyfikator.

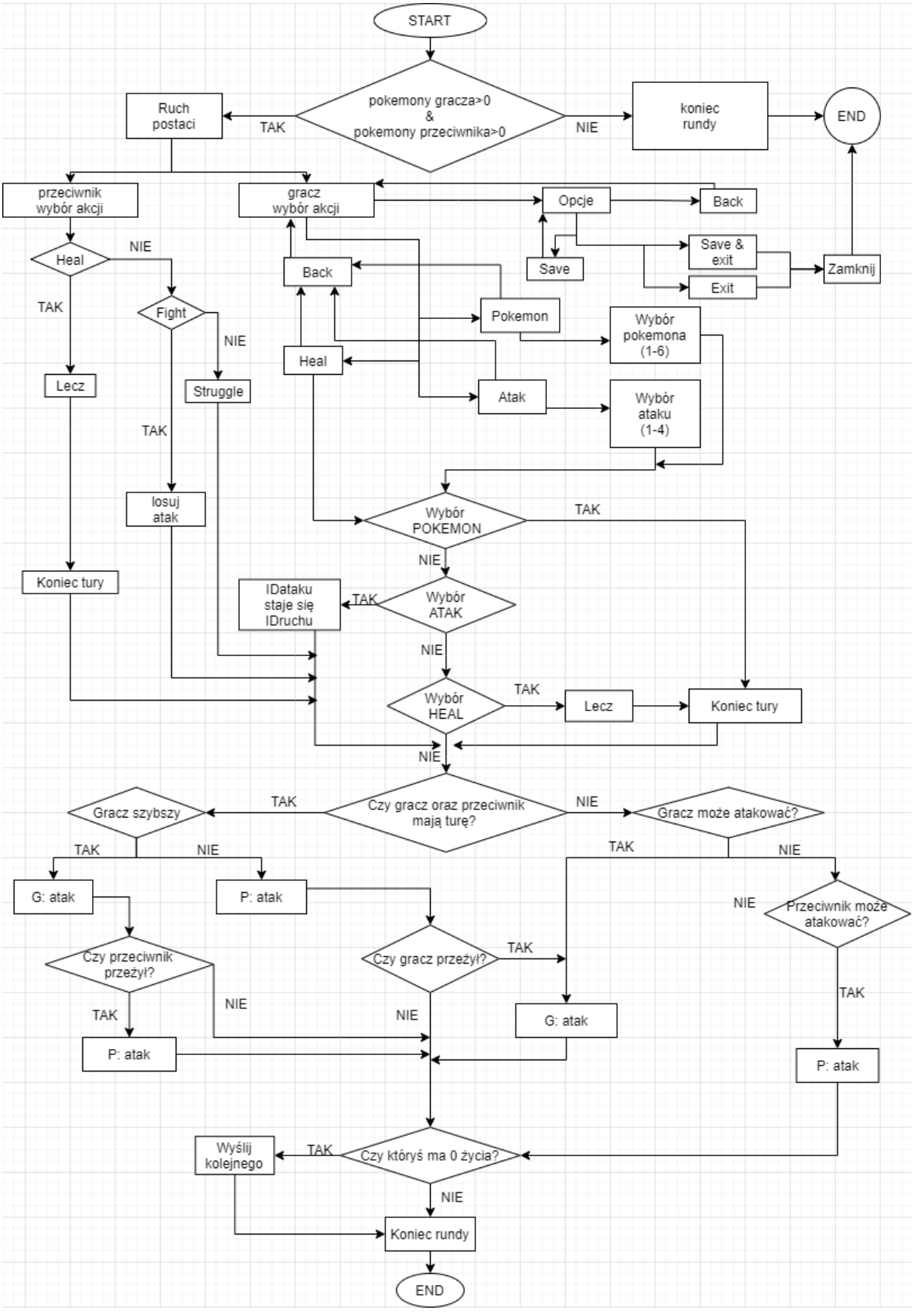
- Pokemon - klasa Pokemona. Każdy z walczących posiada 6 Pokemonów. Każdy z nich ma opisujące go statystyki: nazwa, punkty życia, atak, defensywa, specjalny atak, specjalna defensywa oraz szybkość, ID czterech ruchów, status sprawności do walki, liczba pozostałych ruchów poziom aktualny oraz poziom, na którym może ewoluować. W jej skład wchodzą głównie gettery i settery, ale posiada też mechanizmy ewolucji oraz kalkulowania statystyk rzeczywistych na podstawie statystyk bazowych.

4.2 Ogólny schemat działania programu

W funkcji *main tworzony* jest obiekt klasy *GameSystem*. Posiada ona w swoim konstruktorze funkcję *checkFiles.* Sprawdza wstępnie czy pliki, z których będzie korzystać program istnieją i zawierają sensowne dane. Jeżeli wystąpi błąd rzucany jest wyjątek i program się kończy bez potrzeby tworzenia pozostałych obiektów. Jeżeli pliki okazały się poprawne menedżer scen uruchomi Setup lub Continue w zależności od wyboru gracza w MainMenu. Następnie program przechodzi do części głównej, czyli algorytmu walki (4.3). Podczas swojego działania przechodzi do kolejnych scen: wyboru akcji, wyboru ruchu lub wyboru opcji. Sceny były tworzone z zamiarem jak największego rozdzielenia grafiki od logiki dlatego mają strukturę pollEvent-update-render (4.1.4). Po skończonej rundzie uaktualniane są zmienne i wykonywany jest autozapis do pliku save.sv. Możliwe jest wtedy kontynuowanie gry na dwa sposoby - bezpośrednio przez Continue lub Exit to main menu -> continue.

4.3 Opis algorytmu głównego

Poniższy schemat opisuje algorytm będący kręgosłupem aplikacji. Opisuje zachowanie kolejności akcji w zależności od działań gracza i przeciwnika.


START
pokemony gracza>0 & pokemony przeciwnika>0
TAK
NIE
Ruch postaci
koniec rundy
END
przeciwnik wybór akcji
gracz wybór akcji
Opcje
Back
Save
Save & exit
Exit
Zamknij
Heal
NIE
TAK
Fight
NIE
Back
Pokemon
Wybór pokemona (1-6)
Heal
Atak
Wybór ataku (1-4)
Lecz
Struggle
TAK
losuj atak
Wybór POKEMON
TAK
NIE
Koniec tury
IDataku staje się IDruchu
TAK
Wybór ATAK
NIE
Wybór HEAL
TAK
Lecz
Koniec tury
NIE
Gracz szybszy
TAK
Czy gracz oraz przeciwnik mają turę?
NIE
Gracz może atakować?
TAK
NIE
TAK
NIE
G: atak
P: atak
NIE
Przeciwnik może atakować?
Czy przeciwnik przeżył?
Czy gracz przeżył?
TAK
TAK
NIE
NIE
G: atak
TAK
P: atak
P: atak
Wyślij kolejnego
TAK
Czy któryś ma 0 życia?
NIE
Koniec rundy
END

4.4 Struktury plików:
pokemon.dat
name hp atk def sp_atk sp_def spd type1 type2 mv1 mv2 mv3 mv4 evoName evoLvl
moves.dat
name name PP accuracy power status type category
save.sv
characterLvl player_pkmn1 player_pkmn2 player_pkmn3 player_pkmn4 player_pkmn5 player_pkmn6 enemy_pkmn1 enemy_pkmn2 enemy_pkmn3 enemy_pkmn4 enemy_pkmn5 enemy_pkmn6

5 Testowanie i uruchamianie

5.1 Wycieki pamięci
Program korzysta z technologii inteligentnych wskaźników w każdym miejscu, w którym obiekty są tworzone z pomocą *new* i są odpowiednio dealokowane.

5.2 Ogólne działanie
Rozegrane zostało kilkanaście rund i nie stwierdzono problemów w działaniu.

Początkowo problematyczne było dopilnowanie stanu wszystkich zmiennych podczas zmiany scen np. brak możliwości powrotu do sceny walki jeśli Pokemon jest niezdolny do walki i nie został wybrany nowy lub resetowanie ilości ruchów które pozostały do wykorzystania po wybraniu opcji Continue po rozegranej rundzie.

6 Uwagi

Końcowy produkt nie zdążył nabrać ostatecznego porządku w budowie kodu oraz jego optymalności z powodu ograniczonych ram czasowych względem rozbudowania projektu.