

# **Watch out!**

Dokumentacja projektu gry typu endelss runner

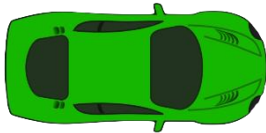
autor: Karolina Kołek

## 1. Instrukcja gry

Celem gry jest zdobycie jak największej liczby punktów.  
Gracz wciela się w kierowcę białego samochodu:



Musi omijać napotkane przeszkody, jeśli tego nie zrobi – odejmowane są punkty życia gracza. Użytkownik może napotkać następujące przeszkody:



- 50 HP



- 20 HP



- 25 HP



- 5 HP



Potrącenie pieszego powoduje natychmiastowe zakończenie gry.



Jeśli gracz trafi na ten symbol, zostanie mu dodanych 10 punktów życia.

Gra kończy się, jeżeli gracz potrąci pieszego lub utraci wszystkie punkty życia.

Sterowanie:

W – jazda w przód

S – jazda w tył

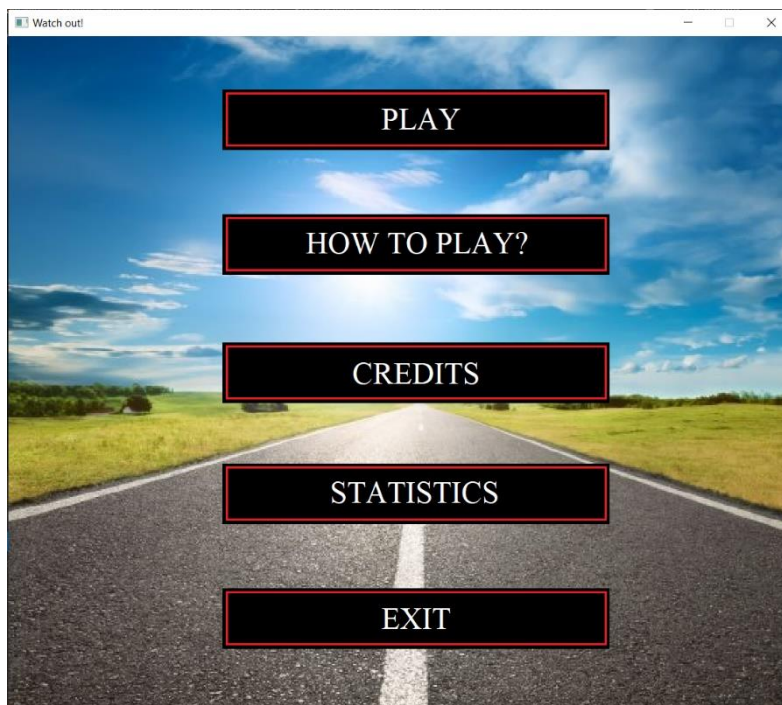
D – jazda w prawo

A – jazda w lewo.

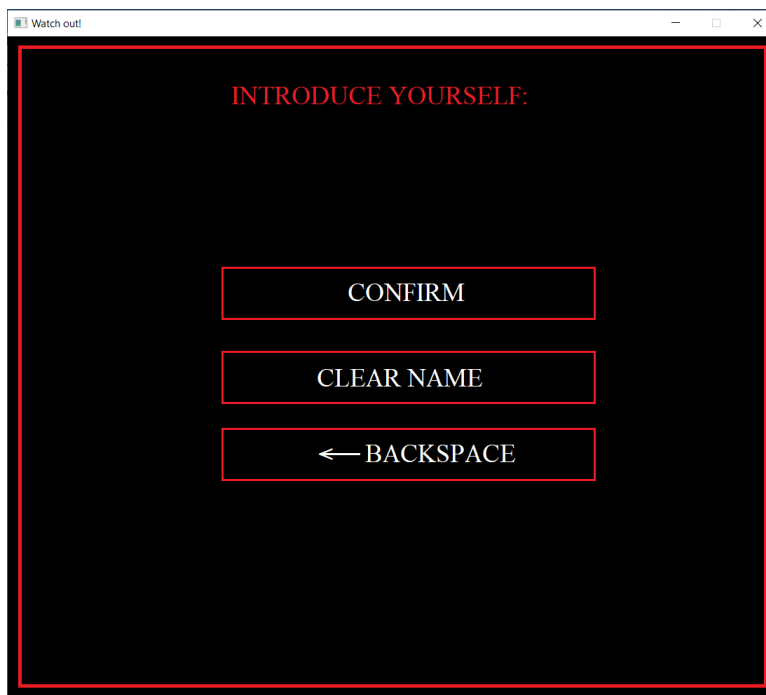
Punkty za ominięcie przeszkody są naliczane, gdy przeszkoda zniknie ze sceny.

## 2. Specyfikacja zewnętrzna

Gra rozpoczyna się wyświetleniem menu głównego:

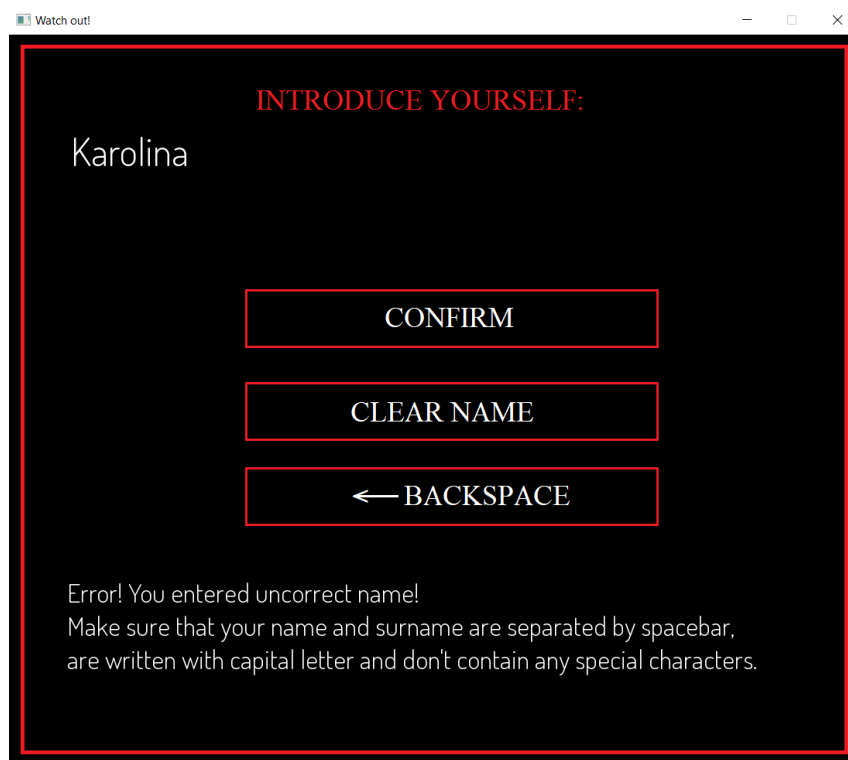


Jeśli gracz wybierze opcję „PLAY”, wyświetla się prośba o przedstawienie się gracza:



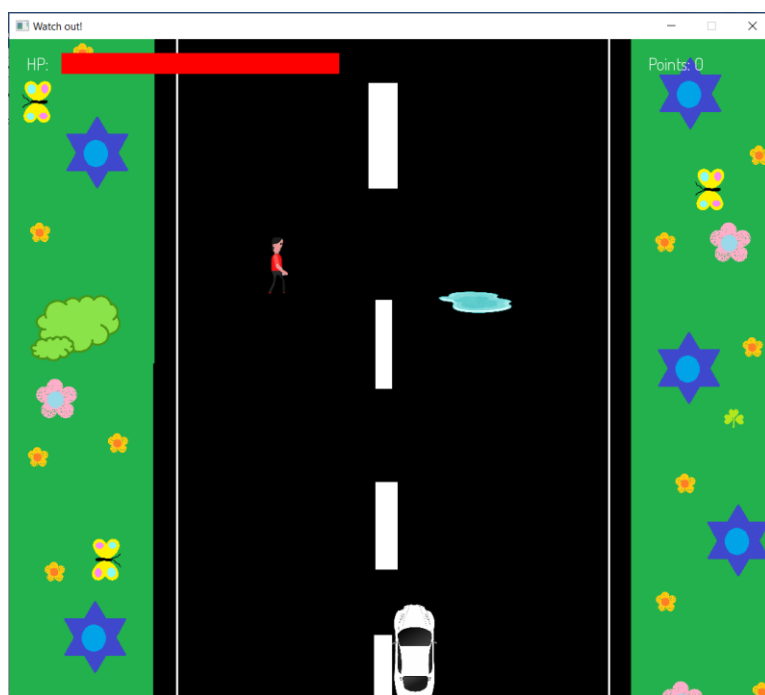
The screenshot shows a window titled "Watch out!" with a black background and a red border. At the top, the text "INTRODUCE YOURSELF:" is displayed in red. Below it, there are three buttons with white text and red borders: "CONFIRM", "CLEAR NAME", and "← BACKSPACE".

Jeśli gracz poda nazwę niezgodną z szablonem (jeśli nie poda imienia i nazwiska rozpoczynających się wielką literą), pojawia się informacja o błędzie:

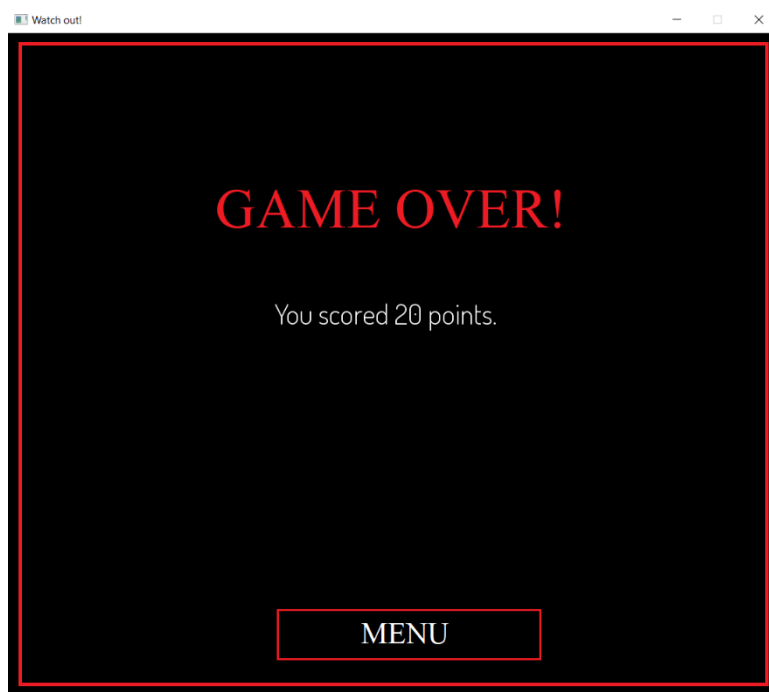


The screenshot shows the same "Watch out!" window. The name "Karolina" has been entered in the input field. Below the input field, an error message is displayed in white text: "Error! You entered uncorrect name! Make sure that your name and surname are separated by spacebar, are written with capital letter and don't contain any special characters." The three buttons remain visible below the error message.

Po podaniu poprawnych danych rozpoczyna się gra:

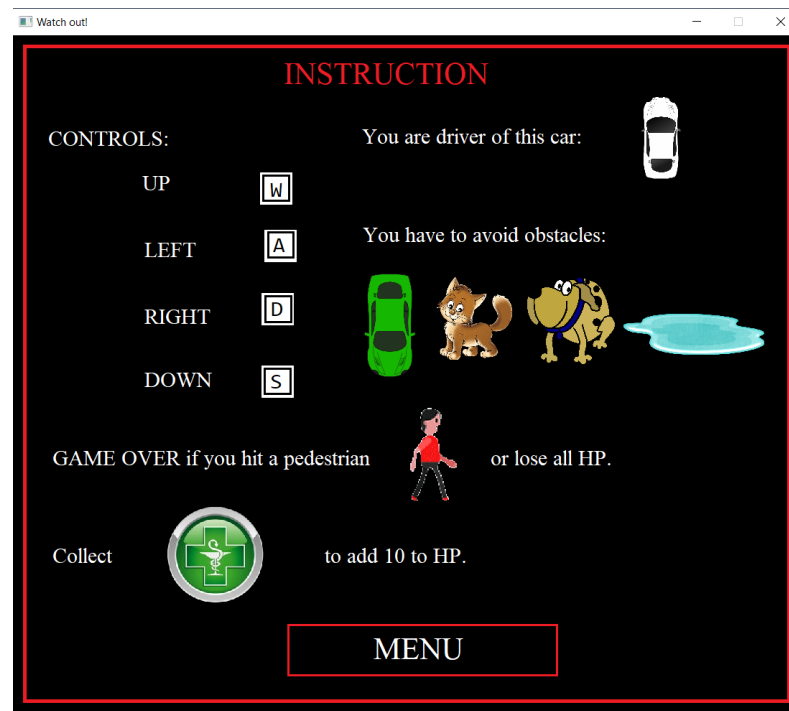


Po utraceniu wszystkich punktów życia pojawia się ekran końca gry:

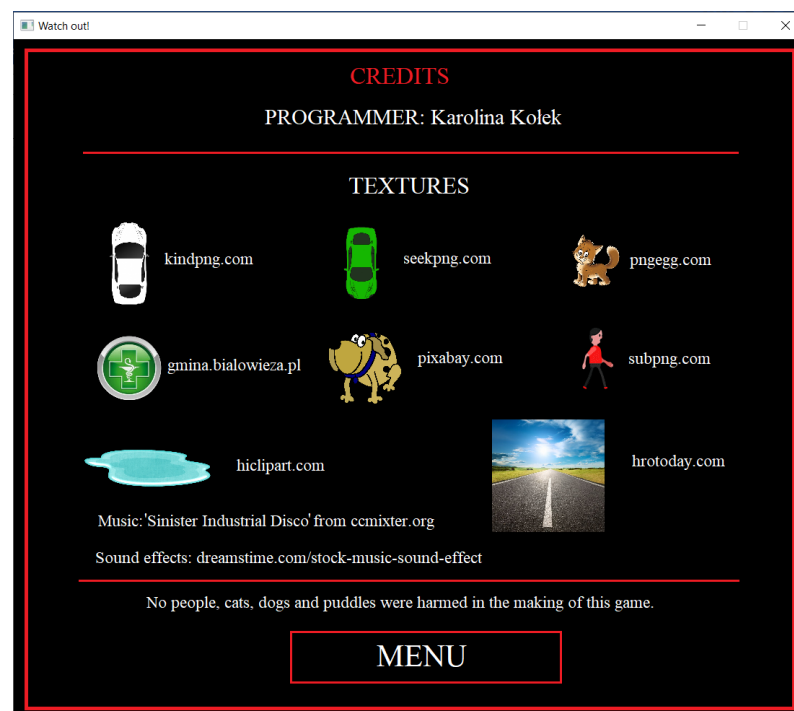


Po czym ponownie wyświetla się menu główne.

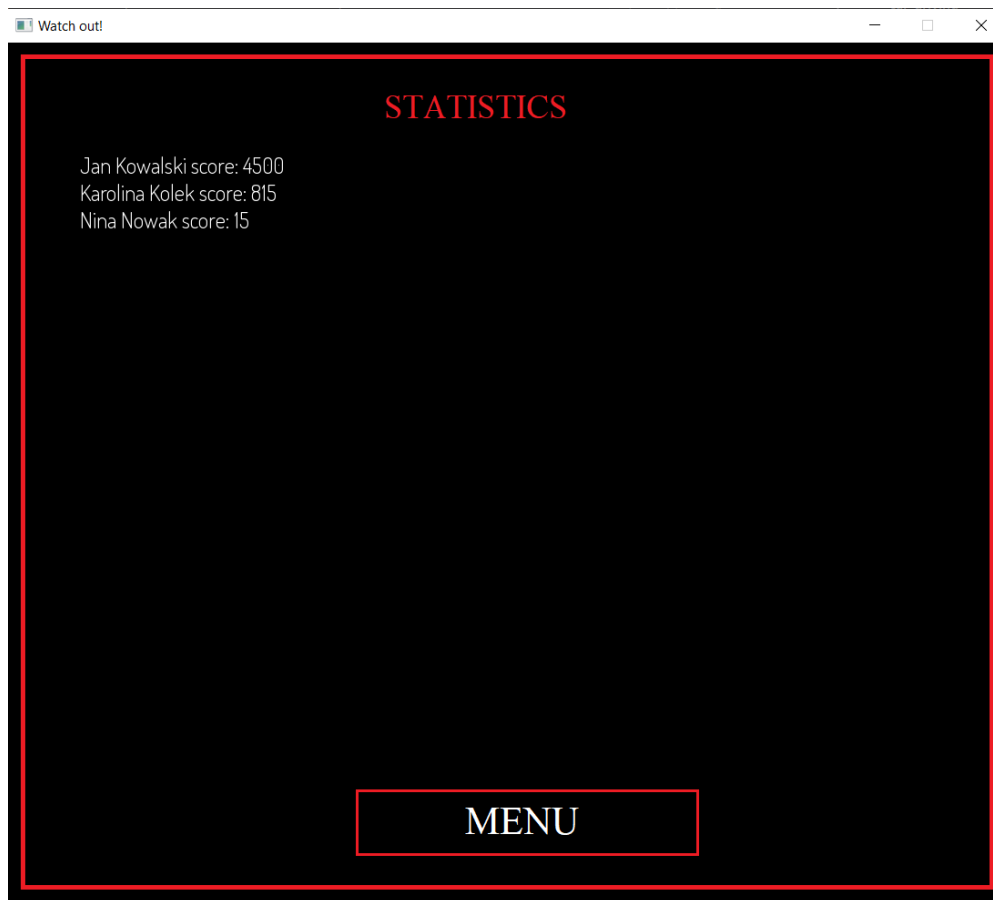
Jeśli gracz wybierze opcję „HOW TO PLAY”, wyświetli się instrukcja gry:



Jeśli gracz wybierze opcję „CREDITS”, wyświetlą się przypisy:



Jeśli gracz wybierze opcję „STATISTICS”, wyświetlą się statystyki:



Wybór opcji „EXIT”, powoduje zakończenie działania programu.

### 3. Specyfikacja wewnętrzna

- Struktury danych:

W projekcie zaimplementowałam listę jednokierunkową do przechowywania danych ze statystyk. Rolę listy pełni klasa *Stats*. Operacje na tej liście wykonuje klasa *list*. Na dane statystyczne składają się nazwa gracza (tzn. imię i nazwisko) oraz liczba punktów przez niego uzyskanych.

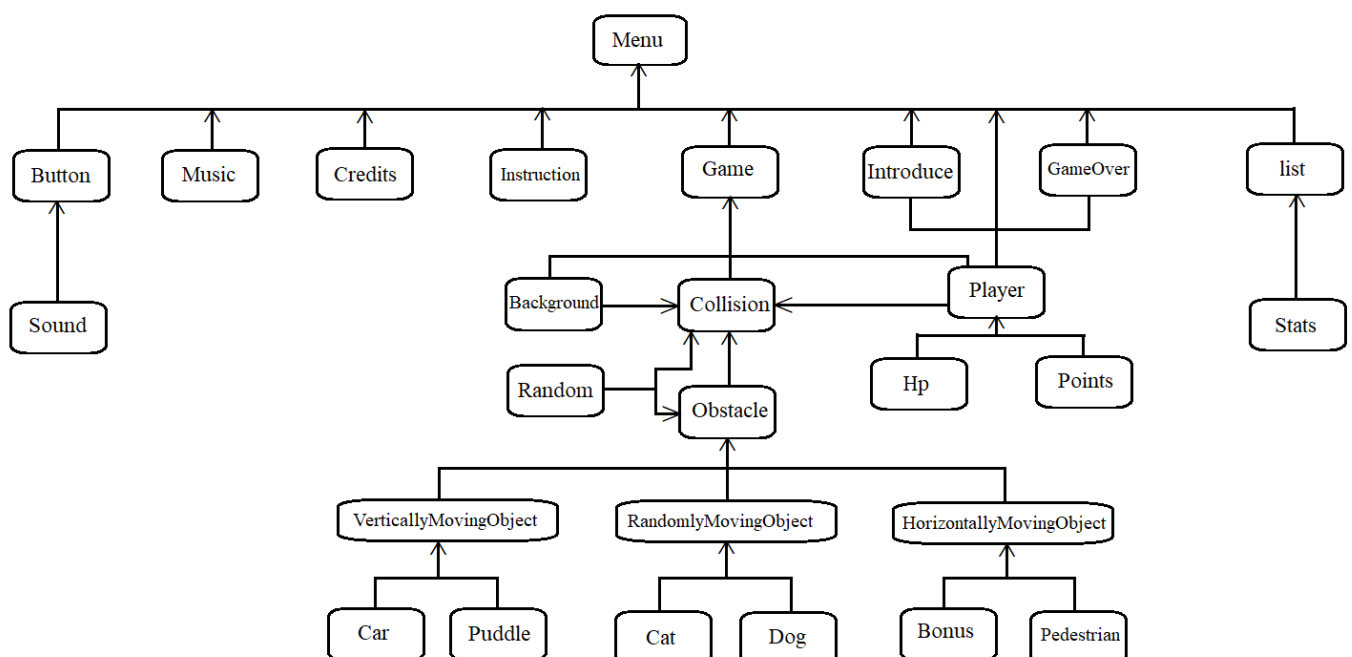
- Klasy:

- Menu – klasa kierująca przebiegiem całej gry. Wyświetla menu główne i odpowiada za wywoływanie funkcji odpowiedzialnych za wyświetlanie statystyk, przypisów, instrukcji oraz uruchomienie gry.
- Button – klasa odpowiadająca za inicjalizację przycisków pojawiających się w programie.
- Sound – klasa odpowiadająca za odtwarzanie dźwięku.
- Music – klasa odpowiadająca za odtwarzanie muzyki.
- Credits – klasa wyświetlająca przypisy.
- Instruction – klasa wyświetlająca przypisy.
- Introduce – klasa odpowiedzialna za pobranie od użytkownika imienia i nazwiska.
- GameOver – klasa odpowiadająca za wyświetlenie okna końca gry.
- Player – klasa odpowiadająca za przechowywanie danych związanych z graczem.
- Points – klasa przechowująca punkty gracza.
- Hp – klasa przechowująca punkty życia gracza.
- Game – klasa odpowiadająca za przeprowadzenie gry.
- Background – klasa odpowiadająca za inicjalizację, wyświetlanie i przesuwanie tła.
- Collision – klasa odpowiadająca za wykrywanie kolizji gracza z przeszkodami oraz za inicjalizację przeszkód i kontrolę ich liczby na scenie.
- Random – klasa odpowiedzialna za generowanie liczb losowych.
- Obstacle – klasa, po której dziedziczą klasy odpowiedzialne za generowanie przeszkód.
- VerticallyMovingObject – klasa, po której dziedziczą przeszkody poruszające się wertykalnie. Klasa ta przechowuje liczbę punktów, jakie zdobędzie gracz, gdy ominie przeszkodę tego typu.



- **HorizontallyMovingObject** – klasa, po której dziedziczą przeszkody poruszające się horyzontalnie. Klasa ta przechowuje liczbę punktów, jakie zdobędzie gracz, gdy ominie przeszkodę tego typu.
- **RandomlyMovingObject** – klasa, po której dziedziczą przeszkody poruszające się chaotycznie. Klasa ta przechowuje liczbę punktów, jakie zdobędzie gracz, gdy ominie przeszkodę tego typu.
- **Car** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie przeszkody w postaci samochodu.
- **Cat** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie przeszkody w postaci kota.
- **Dog** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie przeszkody w postaci psa.
- **Puddle** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie przeszkody w postaci kałuży.
- **Pedestrian** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie przeszkody w postaci pieszego.
- **Bonus** – klasa odpowiadająca za inicjalizację i wyświetlanie na scenie obiektu, po zetknięciu z którym graczowi zostają dodane punkty życia.

Dokładny opis typów i klas dostępny jest w załączniku. Diagram z hierarchią klas:



- Wykorzystane techniki obiektowe

W projekcie skorzystałam z pięciu technik obiektowych: RTTI (zrealizowane w klasie `Collision`), mechanizmu wyjątków (zaimplementowanego w każdej metodzie, która pobiera dane z pliku – jeśli pliku nie da się odnaleźć, program natychmiast kończy swoje działanie), kontenerów STL (a dokładniej wektora, w którym są przechowywane przeszkody. Dzięki temu zabiegowi w prosty sposób mogłam kontrolować liczbę przeszkód na scenie, ich dodawanie i usuwanie), inteligentnych wskaźników, dzięki którym nie musiałam pamiętać o zwalnianiu pamięci, oraz z wyrażenia regularnego, którego użyłam w celu sprawdzenia, czy gracz podaje swoje dane zgodnie z przyjętym szablonem (tzn. imię i nazwisko oddzielone spacją i zaczynające się wielkimi literami).

- Wykorzystane biblioteki zewnętrzne:

W projekcie użyłam biblioteki SFML, dzięki której mogłam dodać do gry grafikę oraz dźwięk.

- Dodatkowa uwaga:

Aby gra była ciekawsza, zaimplementowałam funkcję, która utrudnia rozgrywkę:

- jeśli gracz zdobył liczbę punktów podzielną przez 200, ruch samochodu gracza zostaje spowolniony, a droga zaczyna przesuwać się szybciej.
- jeśli gracz zdobył liczbę punktów podzielną przez 100, która nie jest jednocześnie podzielna przez 200, dopuszczalna liczba przeszkód na scenie zostaje zinkrementowana.
- jeśli gracz zdobył liczbę punktów podzielną przez 50, która nie jest jednocześnie podzielna przez 100 ani 200, przeszkody zaczynają poruszać się szybciej.

#### 4. Testowanie i uruchamianie

Program został przetestowany w następujący sposób:

- Jeśli gracz poda nazwę użytkownika niezgodną z przyjętym szablonem, zmienna przechowująca tę nazwę zostanie wyczyszczona, a na ekranie wyświetli się informacja:

```
Error! You entered uncorrect name!  
Make sure that your name and surname are separated by spacebar,  
are written with capital letter and don't contain any special characters.
```

- Jeśli któregoś z potrzebnych do uruchomienia gry plików nie uda się odnaleźć, program kończy swoje działanie, a w konsoli wyświetlana jest przyczyna błędu.

- Dzięki funkcji `limit()` w klasie `Player` gracz nie ma możliwości wyjścia poza okno gry oraz nie może wyjechać poza jezdnię.

## 5. Uwagi i wnioski

- Początkowo podczas zderzenia gracza z przeszkodą chciałam odtwarzać efekt dźwiękowy adekwatny do typu przeszkody. Niestety, przez to, że obiekt był usuwany przed zakończeniem odtwarzania dźwięku, po zakończeniu działania programu w konsoli pojawiała się informacja o tym, że urządzenie odtwarzające dźwięk nie zostało poprawnie zamknięte:

```
AL lib: (EE) alc_cleanup: 1 device not closed
```

Powodowało to mnóstwo wycieków pamięci. Podjęłam kilka prób pozbycia się tego problemu:

- wprowadziłam wątek kontrolujący odtwarzanie dźwięku i usuwałam obiekty dopiero, gdy wątek odtwarzania dźwięku zakończy swoje działanie
- wprowadziłam zegar dostępny w bibliotece SFML, który nie pozwoliłby usunąć obiektu dopóki dźwięk nadal byłby odtwarzany (niestety, po zastosowaniu tej metody po każdej kolizji gra zawieszała swoje działanie na czas odtwarzania dźwięku, a problem nadal występował)
- jawne wywoływałam destruktor klasy `Sound`

Niestety, żadna z zastosowanych przeze mnie metod nie zadziałała, więc zdecydowałam się na usunięcie efektów dźwiękowych występujących w czasie kolizji. W zamian za to dodałam funkcję wydającą dźwięk podczas gdy gracz klika przycisk.

- Program został przetestowany pod kątem wycieków pamięci. Występuje jeden wyciek, jednakże nie jest on wynikiem mojego zaniedbania, a skutkiem zastosowania biblioteki graficznej. W projekcie używam grafiki dużego rozmiaru, która powoduje, że programy wykrywające wycieki pamięci interpretują pamięć przydzieloną tej grafice jako pamięć zaalokowaną dynamicznie, która nie została zwolniona.

# Załącznik

Szczegółowy opis typów i funkcji

Dokumentacja projektu z przedmiotu Programowanie Komputerow

Generated by Doxygen 1.8.18



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Background Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 Background()	7
4.1.2.2 ~Background()	7
4.1.3 Member Function Documentation	8
4.1.3.1 faster_background()	8
4.1.3.2 render()	8
4.1.3.3 update()	8
4.2 Bonus Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	9
4.2.2.1 Bonus()	9
4.2.2.2 ~Bonus()	9
4.2.3 Member Function Documentation	9
4.2.3.1 getSprite()	9
4.2.3.2 move()	9
4.2.3.3 render()	10
4.2.3.4 update()	10
4.3 Button Class Reference	10
4.3.1 Detailed Description	10
4.3.2 Constructor & Destructor Documentation	10
4.3.2.1 Button()	10
4.3.2.2 ~Button()	11
4.3.3 Member Function Documentation	11
4.3.3.1 change_activity()	11
4.3.3.2 getSprite()	11
4.3.3.3 is_active()	11
4.3.3.4 playSound()	11
4.4 Car Class Reference	12
4.4.1 Detailed Description	12
4.4.2 Constructor & Destructor Documentation	12
4.4.2.1 Car()	12

4.4.2.2 ~Car()	12
4.4.3 Member Function Documentation	13
4.4.3.1 getSprite()	13
4.4.3.2 move()	13
4.4.3.3 render()	13
4.4.3.4 update()	13
4.5 Cat Class Reference	13
4.5.1 Detailed Description	14
4.5.2 Constructor & Destructor Documentation	14
4.5.2.1 Cat()	14
4.5.2.2 ~Cat()	14
4.5.3 Member Function Documentation	14
4.5.3.1 getSprite()	14
4.5.3.2 limit()	15
4.5.3.3 move()	15
4.5.3.4 render()	15
4.5.3.5 update()	15
4.6 Collision Class Reference	15
4.6.1 Detailed Description	16
4.6.2 Constructor & Destructor Documentation	16
4.6.2.1 Collision()	16
4.6.2.2 ~Collision()	16
4.6.3 Member Function Documentation	16
4.6.3.1 collision()	16
4.6.3.2 delete_from_scene()	16
4.6.3.3 make_harder()	17
4.6.3.4 render()	17
4.6.3.5 spawnObstacles()	17
4.6.3.6 update()	17
4.6.3.7 updateObstacles()	17
4.7 Credits Class Reference	17
4.7.1 Detailed Description	18
4.7.2 Constructor & Destructor Documentation	18
4.7.2.1 Credits()	18
4.7.2.2 ~Credits()	18
4.7.3 Member Function Documentation	18
4.7.3.1 render()	18
4.8 Dog Class Reference	19
4.8.1 Detailed Description	19
4.8.2 Constructor & Destructor Documentation	19
4.8.2.1 Dog()	19
4.8.2.2 ~Dog()	19



4.8.3 Member Function Documentation . . . . .	20
4.8.3.1 getSprite() . . . . .	20
4.8.3.2 limit() . . . . .	20
4.8.3.3 move() . . . . .	20
4.8.3.4 render() . . . . .	20
4.8.3.5 update() . . . . .	20
4.9 Game Class Reference . . . . .	20
4.9.1 Detailed Description . . . . .	21
4.9.2 Constructor & Destructor Documentation . . . . .	21
4.9.2.1 Game() . . . . .	21
4.9.2.2 ~Game() . . . . .	21
4.9.3 Member Function Documentation . . . . .	21
4.9.3.1 end() . . . . .	21
4.9.3.2 render() . . . . .	22
4.9.3.3 set_was_played() . . . . .	22
4.9.3.4 update() . . . . .	22
4.9.3.5 was_played() . . . . .	22
4.10 GameOver Class Reference . . . . .	22
4.10.1 Detailed Description . . . . .	22
4.10.2 Constructor & Destructor Documentation . . . . .	22
4.10.2.1 GameOver() . . . . .	22
4.10.2.2 ~GameOver() . . . . .	23
4.10.3 Member Function Documentation . . . . .	23
4.10.3.1 render() . . . . .	23
4.11 HorizontallyMovingObject Class Reference . . . . .	23
4.11.1 Detailed Description . . . . .	24
4.11.2 Constructor & Destructor Documentation . . . . .	24
4.11.2.1 HorizontallyMovingObject() . . . . .	24
4.11.2.2 ~HorizontallyMovingObject() . . . . .	24
4.11.3 Member Function Documentation . . . . .	24
4.11.3.1 get_points() . . . . .	24
4.11.3.2 set_points() . . . . .	24
4.12 HorizontalMovingObstacle Class Reference . . . . .	25
4.13 Hp Class Reference . . . . .	25
4.13.1 Detailed Description . . . . .	25
4.13.2 Constructor & Destructor Documentation . . . . .	25
4.13.2.1 Hp() . . . . .	25
4.13.2.2 ~Hp() . . . . .	25
4.13.3 Member Function Documentation . . . . .	25
4.13.3.1 getHp() . . . . .	26
4.13.3.2 render() . . . . .	26
4.13.3.3 subtractHp() . . . . .	26

4.13.3.4 update()	26
4.14 Instruction Class Reference	26
4.14.1 Detailed Description	26
4.14.2 Constructor & Destructor Documentation	27
4.14.2.1 Instruction()	27
4.14.2.2 ~Instruction()	27
4.14.3 Member Function Documentation	27
4.14.3.1 render()	27
4.15 Introduce Class Reference	27
4.15.1 Detailed Description	28
4.15.2 Constructor & Destructor Documentation	28
4.15.2.1 Introduce()	28
4.15.2.2 ~Introduce()	28
4.15.3 Member Function Documentation	28
4.15.3.1 check()	28
4.15.3.2 is_correct()	28
4.15.3.3 pollEvents()	29
4.15.3.4 render()	29
4.15.3.5 set_backspace()	29
4.15.3.6 set_clear()	29
4.15.3.7 set_confirm()	29
4.15.3.8 set_player_name()	29
4.15.3.9 update()	30
4.16 list Class Reference	30
4.16.1 Detailed Description	30
4.16.2 Constructor & Destructor Documentation	31
4.16.2.1 list() [1/4]	31
4.16.2.2 ~list()	31
4.16.2.3 list() [2/4]	31
4.16.2.4 list() [3/4]	31
4.16.2.5 list() [4/4]	31
4.16.3 Member Function Documentation	31
4.16.3.1 actual_save()	32
4.16.3.2 add()	32
4.16.3.3 delete_record()	32
4.16.3.4 delete_stats()	32
4.16.3.5 find()	32
4.16.3.6 load()	32
4.16.3.7 open_close_file()	33
4.16.3.8 operator=() [1/2]	33
4.16.3.9 operator=() [2/2]	33
4.16.3.10 prepare_to_print()	33

4.16.3.11 print()	33
4.16.3.12 save()	33
4.16.3.13 try_to_add()	34
4.16.4 Member Data Documentation	34
4.16.4.1 pHead	34
4.17 Menu Class Reference	34
4.17.1 Constructor & Destructor Documentation	35
4.17.1.1 Menu()	35
4.17.1.2 ~Menu()	35
4.17.2 Member Function Documentation	35
4.17.2.1 game_over()	35
4.17.2.2 introduce_player()	35
4.17.2.3 is_running()	35
4.17.2.4 make_the_game()	35
4.17.2.5 pollEvents()	36
4.17.2.6 render()	36
4.17.2.7 render_credits()	36
4.17.2.8 render_game()	36
4.17.2.9 render_instruction()	36
4.17.2.10 render_menu()	36
4.17.2.11 render_stats()	36
4.17.2.12 update()	37
4.18 Music Class Reference	37
4.18.1 Detailed Description	37
4.18.2 Constructor & Destructor Documentation	37
4.18.2.1 Music()	37
4.18.2.2 ~Music()	37
4.18.3 Member Function Documentation	38
4.18.3.1 playMusic()	38
4.19 Obstacle Class Reference	38
4.19.1 Detailed Description	39
4.19.2 Constructor & Destructor Documentation	39
4.19.2.1 Obstacle()	39
4.19.2.2 ~Obstacle()	39
4.19.3 Member Function Documentation	39
4.19.3.1 accelerate()	39
4.19.3.2 addObstacles()	39
4.19.3.3 get_damage()	39
4.19.3.4 get_speed()	40
4.19.3.5 getMaxObstacles()	40
4.19.3.6 move()	40
4.19.3.7 set_damage()	40

4.19.3.8 set_factor()	40
4.19.3.9 set_speed()	40
4.19.4 Member Data Documentation	40
4.19.4.1 random	41
4.20 Pedestrian Class Reference	41
4.20.1 Detailed Description	41
4.20.2 Constructor & Destructor Documentation	41
4.20.2.1 Pedestrian()	42
4.20.2.2 ~Pedestrian()	42
4.20.3 Member Function Documentation	42
4.20.3.1 getSprite()	42
4.20.3.2 move()	42
4.20.3.3 pedestrianMovement()	42
4.20.3.4 render()	42
4.20.3.5 update()	43
4.21 Player Class Reference	43
4.21.1 Detailed Description	43
4.21.2 Constructor & Destructor Documentation	43
4.21.2.1 Player()	43
4.21.2.2 ~Player()	44
4.21.3 Member Function Documentation	44
4.21.3.1 getHp()	44
4.21.3.2 getName()	44
4.21.3.3 getPoints()	44
4.21.3.4 getSprite()	44
4.21.3.5 input()	44
4.21.3.6 limit()	44
4.21.3.7 move()	45
4.21.3.8 render()	45
4.21.3.9 setName()	45
4.21.3.10 slowTheCar()	45
4.21.3.11 update()	45
4.22 Points Class Reference	45
4.22.1 Detailed Description	46
4.22.2 Constructor & Destructor Documentation	46
4.22.2.1 Points()	46
4.22.2.2 ~Points()	46
4.22.3 Member Function Documentation	46
4.22.3.1 addPoints()	46
4.22.3.2 getPoints()	46
4.22.3.3 render()	47
4.22.3.4 update()	47

4.23 Puddle Class Reference	47
4.23.1 Detailed Description	47
4.23.2 Constructor & Destructor Documentation	48
4.23.2.1 Puddle()	48
4.23.2.2 ~Puddle()	48
4.23.3 Member Function Documentation	48
4.23.3.1 getSprite()	48
4.23.3.2 move()	48
4.23.3.3 render()	48
4.23.3.4 update()	49
4.24 Random Class Reference	49
4.24.1 Detailed Description	49
4.24.2 Constructor & Destructor Documentation	49
4.24.2.1 Random()	49
4.24.2.2 ~Random()	49
4.24.3 Member Function Documentation	49
4.24.3.1 get_random_number()	50
4.25 RandomlyMovingObject Class Reference	50
4.25.1 Detailed Description	50
4.25.2 Constructor & Destructor Documentation	50
4.25.2.1 RandomlyMovingObject()	51
4.25.2.2 ~RandomlyMovingObject()	51
4.25.3 Member Function Documentation	51
4.25.3.1 get_points()	51
4.25.3.2 set_points()	51
4.26 Sound Class Reference	51
4.26.1 Detailed Description	52
4.26.2 Constructor & Destructor Documentation	52
4.26.2.1 Sound()	52
4.26.2.2 ~Sound()	53
4.26.3 Member Function Documentation	53
4.26.3.1 playSound()	53
4.27 Stats Class Reference	53
4.27.1 Detailed Description	54
4.27.2 Constructor & Destructor Documentation	54
4.27.2.1 Stats() [1/3]	54
4.27.2.2 ~Stats()	54
4.27.2.3 Stats() [2/3]	54
4.27.2.4 Stats() [3/3]	54
4.27.3 Member Function Documentation	54
4.27.3.1 getName()	55
4.27.3.2 getScore()	55

4.27.3.3 operator=() [1/2]	55
4.27.3.4 operator=() [2/2]	55
4.27.3.5 render()	55
4.27.3.6 setVariables()	55
4.27.4 Member Data Documentation	55
4.27.4.1 pNext	56
4.28 VerticallyMovingObject Class Reference	56
4.28.1 Detailed Description	56
4.28.2 Constructor & Destructor Documentation	56
4.28.2.1 VerticallyMovingObject()	56
4.28.2.2 ~VerticallyMovingObject()	57
4.28.3 Member Function Documentation	57
4.28.3.1 get_points()	57
4.28.3.2 set_points()	57
<b>5 File Documentation</b>	<b>59</b>
5.1 Background.cpp File Reference	59
5.2 Background.h File Reference	59
5.3 Bonus.cpp File Reference	59
5.4 Bonus.h File Reference	59
5.5 Button.cpp File Reference	60
5.6 Button.h File Reference	60
5.7 Car.cpp File Reference	60
5.8 Car.h File Reference	60
5.9 car_driving.cpp File Reference	60
5.9.1 Function Documentation	61
5.9.1.1 main()	61
5.10 Cat.cpp File Reference	61
5.11 Cat.h File Reference	61
5.12 Collision.cpp File Reference	61
5.13 Collision.h File Reference	61
5.14 Credits.cpp File Reference	62
5.15 Credits.h File Reference	62
5.16 Dog.cpp File Reference	62
5.17 Dog.h File Reference	62
5.18 Game.cpp File Reference	62
5.19 Game.h File Reference	63
5.20 GameOver.cpp File Reference	63
5.21 GameOver.h File Reference	63
5.22 HorizontallyMovingObject.cpp File Reference	63
5.23 HorizontallyMovingObject.h File Reference	63
5.24 HorizontalMovingObstacle.h File Reference	64

5.25 Hp.cpp File Reference . . . . .	64
5.26 Hp.h File Reference . . . . .	64
5.27 Instruction.cpp File Reference . . . . .	64
5.28 Instruction.h File Reference . . . . .	64
5.29 Introduce.cpp File Reference . . . . .	65
5.30 Introduce.h File Reference . . . . .	65
5.31 list.cpp File Reference . . . . .	65
5.32 list.h File Reference . . . . .	65
5.33 Menu.cpp File Reference . . . . .	66
5.34 Menu.h File Reference . . . . .	66
5.35 Music.cpp File Reference . . . . .	66
5.36 Music.h File Reference . . . . .	66
5.37 Obstacle.cpp File Reference . . . . .	66
5.38 Obstacle.h File Reference . . . . .	67
5.39 Pedestrian.cpp File Reference . . . . .	67
5.40 Pedestrian.h File Reference . . . . .	67
5.41 Player.cpp File Reference . . . . .	67
5.42 Player.h File Reference . . . . .	67
5.43 Points.cpp File Reference . . . . .	68
5.44 Points.h File Reference . . . . .	68
5.45 Puddle.cpp File Reference . . . . .	68
5.46 Puddle.h File Reference . . . . .	68
5.47 Random.cpp File Reference . . . . .	68
5.48 Random.h File Reference . . . . .	69
5.49 RandomlyMovingObject.cpp File Reference . . . . .	69
5.50 RandomlyMovingObject.h File Reference . . . . .	69
5.51 resource.h File Reference . . . . .	69
5.52 Sound.cpp File Reference . . . . .	69
5.53 Sound.h File Reference . . . . .	69
5.54 statistics.txt File Reference . . . . .	70
5.55 Stats.cpp File Reference . . . . .	70
5.56 Stats.h File Reference . . . . .	70
5.57 VerticallyMovingObject.cpp File Reference . . . . .	70
5.58 VerticallyMovingObject.h File Reference . . . . .	70
<b>Index</b>	<b>71</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Background . . . . .	7
Button . . . . .	10
Collision . . . . .	15
Credits . . . . .	17
Game . . . . .	20
GameOver . . . . .	22
HorizontalMovingObstacle . . . . .	25
Hp . . . . .	25
Instruction . . . . .	26
Introduce . . . . .	27
list . . . . .	30
Menu . . . . .	34
Music . . . . .	37
Obstacle . . . . .	38
HorizontallyMovingObject . . . . .	23
Bonus . . . . .	8
Pedestrian . . . . .	41
RandomlyMovingObject . . . . .	50
Cat . . . . .	13
Dog . . . . .	19
VerticallyMovingObject . . . . .	56
Car . . . . .	12
Puddle . . . . .	47
Player . . . . .	43
Points . . . . .	45
Random . . . . .	49
Sound . . . . .	51
Stats . . . . .	53



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Background	7
Bonus	8
Button	10
Car	12
Cat	13
Collision	15
Credits	17
Dog	19
Game	20
GameOver	22
HorizontallyMovingObject	23
HorizontalMovingObstacle	25
Hp	25
Instruction	26
Introduce	27
list	30
Menu	34
Music	37
Obstacle	38
Pedestrian	41
Player	43
Points	45
Puddle	47
Random	49
RandomlyMovingObject	50
Sound	51
Stats	53
VerticallyMovingObject	56



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

Background.cpp	59
Background.h	59
Bonus.cpp	59
Bonus.h	59
Button.cpp	60
Button.h	60
Car.cpp	60
Car.h	60
car_driving.cpp	60
Cat.cpp	61
Cat.h	61
Collision.cpp	61
Collision.h	61
Credits.cpp	62
Credits.h	62
Dog.cpp	62
Dog.h	62
Game.cpp	62
Game.h	63
GameOver.cpp	63
GameOver.h	63
HorizontallyMovingObject.cpp	63
HorizontallyMovingObject.h	63
HorizontalMovingObstacle.h	64
Hp.cpp	64
Hp.h	64
Instruction.cpp	64
Instruction.h	64
Introduce.cpp	65
Introduce.h	65
list.cpp	65
list.h	65
Menu.cpp	66
Menu.h	66
Music.cpp	66

Music.h	66
Obstacle.cpp	66
Obstacle.h	67
Pedestrian.cpp	67
Pedestrian.h	67
Player.cpp	67
Player.h	67
Points.cpp	68
Points.h	68
Puddle.cpp	68
Puddle.h	68
Random.cpp	68
Random.h	69
RandomlyMovingObject.cpp	69
RandomlyMovingObject.h	69
resource.h	69
Sound.cpp	69
Sound.h	69
Stats.cpp	70
Stats.h	70
VerticallyMovingObject.cpp	70
VerticallyMovingObject.h	70

## Chapter 4

# Class Documentation

### 4.1 Background Class Reference

```
#include <Background.h>
```

#### Public Member Functions

- [Background](#) ()
- [~Background](#) ()
- void [faster\\_background](#) ()
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

#### 4.1.1 Detailed Description

Klasa odpowiadająca za inicjalizację, wyświetlanie i przesuwanie tła podczas gry.

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 Background()

```
Background::Background ( )
```

Konstruktor klasy [Background](#)

##### 4.1.2.2 ~Background()

```
Background::~~Background ( )
```

Destruktor klasy [Background](#)

### 4.1.3 Member Function Documentation

#### 4.1.3.1 faster\_background()

```
void Background::faster_background ( )
```

Metoda przyspieszająca ruch tła (skrócony zostaje czas zegara wykorzystywany w funkcji [update\(\)](#)).

#### 4.1.3.2 render()

```
void Background::render (
    sf::RenderTarget * target )
```

Metoda odpowiadająca za wyświetlenie tła na ekranie.

#### 4.1.3.3 update()

```
void Background::update ( )
```

Metoda aktualizująca tło (tutaj zaimplementowane jest przesuwanie się tła).

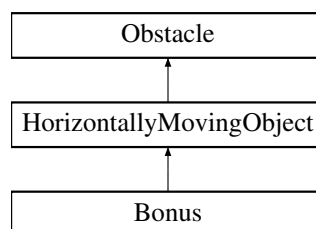
The documentation for this class was generated from the following files:

- [Background.h](#)
- [Background.cpp](#)

## 4.2 Bonus Class Reference

```
#include <Bonus.h>
```

Inheritance diagram for Bonus:



### Public Member Functions

- [Bonus](#) ()
- [~Bonus](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)



## Additional Inherited Members

### 4.2.1 Detailed Description

Klasa dziedziczaca po klasie [HorizontallyMovingObject](#), inicjujaca i wyswietlajaca obiekt, po zderzeniu z ktorym gracz otrzymuje dodatkowe punkty zycia.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Bonus()

```
Bonus::Bonus ( )
```

Konstruktor klasy [Bonus](#).

#### 4.2.2.2 ~Bonus()

```
Bonus::~~Bonus ( )
```

Destruktor klasy [Bonus](#).

### 4.2.3 Member Function Documentation

#### 4.2.3.1 getSprite()

```
const sf::Sprite & Bonus::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.2.3.2 move()

```
void Bonus::move ( ) [override], [virtual]
```

Metoda odpowiadajaca za ruch obiektu.

Reimplemented from [Obstacle](#).

#### 4.2.3.3 render()

```
void Bonus::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca obiekt na scenie.

#### 4.2.3.4 update()

```
void Bonus::update ( )
```

Metoda aktualizujaca pozycje obiektu na scenie.

The documentation for this class was generated from the following files:

- [Bonus.h](#)
- [Bonus.cpp](#)

## 4.3 Button Class Reference

```
#include <Button.h>
```

### Public Member Functions

- [Button](#) (std::string path, sf::Vector2f pos)
- [~Button](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [playSound](#) ()
- bool [is\\_active](#) ()
- void [change\\_activity](#) (bool activity)

#### 4.3.1 Detailed Description

Klasa odpowiedzialna za dzialanie przyciskow w grze.

#### 4.3.2 Constructor & Destructor Documentation

##### 4.3.2.1 Button()

```
Button::Button (
    std::string path,
    sf::Vector2f pos )
```

Konstruktor klasy [Button](#).

## Parameters

<i>path</i>	ścieżka do pliku z grafika, która ma się znaleźć na przycisku
<i>pos</i>	pozycja przycisku na scenie

#### 4.3.2.2 ~Button()

```
Button::~~Button ( )
```

Destruktor klasy [Button](#).

### 4.3.3 Member Function Documentation

#### 4.3.3.1 change\_activity()

```
void Button::change_activity (
    bool activity )
```

Metoda zmieniająca aktywność przycisku.

#### 4.3.3.2 getSprite()

```
const sf::Sprite & Button::getSprite ( ) const
```

Metoda zwracająca sprite.

#### 4.3.3.3 is\_active()

```
bool Button::is_active ( )
```

Metoda zwracająca wartość true, jeśli przycisk jest aktualnie aktywny i false, jeżeli nie jest aktywny.

#### 4.3.3.4 playSound()

```
void Button::playSound ( )
```

Metoda wywołująca funkcję klasy [Sound](#), która odtwarza dźwięk.

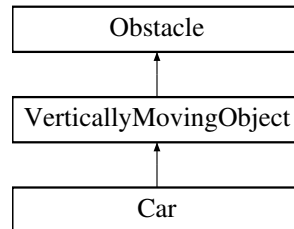
The documentation for this class was generated from the following files:

- [Button.h](#)
- [Button.cpp](#)

## 4.4 Car Class Reference

```
#include <Car.h>
```

Inheritance diagram for Car:



### Public Member Functions

- [Car](#) ()
- [~Car](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### Additional Inherited Members

#### 4.4.1 Detailed Description

Klasa dziedziczaca po klasie [VerticallyMovingObject](#), inicjujaca i wyswietlajaca przeszkode w postaci samochodu.

#### 4.4.2 Constructor & Destructor Documentation

##### 4.4.2.1 Car()

```
Car::Car ( )
```

Konstruktor klasy [Car](#).

##### 4.4.2.2 ~Car()

```
Car::~~Car ( )
```

Destruktor klasy [Car](#).

### 4.4.3 Member Function Documentation

#### 4.4.3.1 getSprite()

```
const sf::Sprite & Car::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.4.3.2 move()

```
void Car::move ( ) [override], [virtual]
```

Metoda odpowiadajaca za ruch obiektu.

Reimplemented from [Obstacle](#).

#### 4.4.3.3 render()

```
void Car::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca obiekt na scenie.

#### 4.4.3.4 update()

```
void Car::update ( )
```

Metoda aktualizujaca pozycje obiektu na scenie.

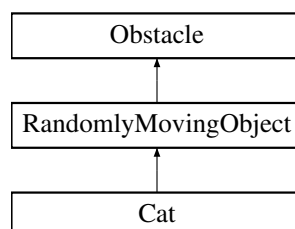
The documentation for this class was generated from the following files:

- [Car.h](#)
- [Car.cpp](#)

## 4.5 Cat Class Reference

```
#include <Cat.h>
```

Inheritance diagram for Cat:



## Public Member Functions

- [Cat](#) ()
- [~Cat](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [limit](#) ()
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

## Additional Inherited Members

### 4.5.1 Detailed Description

Klasa dziedziczaca po klasie [RandomlyMovingObject](#), inicjujaca i wyswietlajaca przeszkode w postaci kota.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 Cat()

```
Cat::Cat ( )
```

Konstruktor klasy [Cat](#).

#### 4.5.2.2 ~Cat()

```
Cat::~~Cat ( )
```

Destruktor klasy [Cat](#).

### 4.5.3 Member Function Documentation

#### 4.5.3.1 getSprite()

```
const sf::Sprite & Cat::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.5.3.2 limit()

```
void Cat::limit ( )
```

Metoda odpowiadająca za to, by obiekt nie pojawiał się poza polem ruchu gracza.

#### 4.5.3.3 move()

```
void Cat::move ( ) [override], [virtual]
```

Metoda odpowiadająca za ruch obiektu.

Reimplemented from [Obstacle](#).

#### 4.5.3.4 render()

```
void Cat::render (
    sf::RenderTarget * target )
```

Metoda wyświetlająca obiekt na scenie.

#### 4.5.3.5 update()

```
void Cat::update ( )
```

Metoda aktualizująca pozycje obiektu na scenie.

The documentation for this class was generated from the following files:

- [Cat.h](#)
- [Cat.cpp](#)

## 4.6 Collision Class Reference

```
#include <Collision.h>
```

### Public Member Functions

- [Collision](#) ()
- [~Collision](#) ()
- void [spawnObstacles](#) ()
- void [collision](#) (std::shared\_ptr< [Player](#) > player)
- void [delete\\_from\\_scene](#) (std::shared\_ptr< [Player](#) > player)
- void [update](#) (std::shared\_ptr< [Player](#) > player, std::shared\_ptr< [Background](#) > background, sf::RenderTarget \*target)
- void [make\\_harder](#) (std::shared\_ptr< [Player](#) > player, std::shared\_ptr< [Background](#) > background)
- void [updateObstacles](#) ()
- void [render](#) (sf::RenderTarget \*target)

### 4.6.1 Detailed Description

Klasa odpowiadająca za inicjalizację przeszkód, wykrywanie kolizji przeszkód z graczem oraz usuwanie obiektów ze sceny.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 Collision()

```
Collision::Collision ( )
```

Konstruktor klasy [Collision](#).

##### Parameters

<i>o</i>	shared pointer do obiektu klasy <a href="#">Obstacle</a>
----------	--

#### 4.6.2.2 ~Collision()

```
Collision::~~Collision ( )
```

Destruktor klasy [Collision](#).

### 4.6.3 Member Function Documentation

#### 4.6.3.1 collision()

```
void Collision::collision (
    std::shared_ptr< Player > player )
```

Metoda wykrywająca kolizje przeszkód z graczem. Jeśli nastąpi kolizja, graczowi odbierane są punkty życia, a przeszkoda usuwana jest ze sceny.

#### 4.6.3.2 delete\_from\_scene()

```
void Collision::delete_from_scene (
    std::shared_ptr< Player > player )
```

Metoda usuwająca obiekty, które znalazły się poza polem gry.



#### 4.6.3.3 make\_harder()

```
void Collision::make_harder (
    std::shared_ptr< Player > player,
    std::shared_ptr< Background > background )
```

Metoda utrudniajaca gre. Jesli gracz zbierze liczbe punktow podzielna przez 200, samochod gracza zwalnia, a ruch tla staje sie szybszy. Jesli gracz zbierze liczbe punktow podzielna przez 100, liczba przeszkod na scenie zostaje zinkrementowana. Jesli gracz zbierze liczbe punktow podzielna przez 50, przeszkody przyspieszaja. Naraz moze zostac wprowadzone tylko jedno utrudnienie (np. gdy gracz zbierze liczbe punktow rown<sup>1</sup> 300, zostaje zinkrementowana liczba przeszkod, ale przeszkody nie przyspieszaja).

#### 4.6.3.4 render()

```
void Collision::render (
    sf::RenderTarget * target )
```

Metoda wywolujaca funkcje [render\(\)](#) dla wszystkich przeszkod na scenie.

#### 4.6.3.5 spawnObstacles()

```
void Collision::spawnObstacles ( )
```

Metoda tworzaca nowe przeszkody.

#### 4.6.3.6 update()

```
void Collision::update (
    std::shared_ptr< Player > player,
    std::shared_ptr< Background > background,
    sf::RenderTarget * target )
```

Metoda aktualizuje stan przeszkod, tzn. sprawdza, czy nastapila kolizja, czy obiekt znalazl sie poza polem gry oraz czy nalezy dodac nowe przeszkody.

#### 4.6.3.7 updateObstacles()

```
void Collision::updateObstacles ( )
```

Metoda wywolujaca funkcje [update\(\)](#) dla wszystkich przeszkod na scenie.

The documentation for this class was generated from the following files:

- [Collision.h](#)
- [Collision.cpp](#)

## 4.7 Credits Class Reference

```
#include <Credits.h>
```

## Public Member Functions

- [Credits](#) ()
- [~Credits](#) ()
- void [render](#) (sf::RenderTarget \*target)

### 4.7.1 Detailed Description

Klasa odpowiadająca za inicjalizację i wyświetlanie przypisów.

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 Credits()

```
Credits::Credits ( )
```

Konstruktor klasy [Credits](#).

#### 4.7.2.2 ~Credits()

```
Credits::~Credits ( )
```

Destruktor klasy [Credits](#).

### 4.7.3 Member Function Documentation

#### 4.7.3.1 render()

```
void Credits::render (
    sf::RenderTarget * target )
```

Metoda wyświetlająca przypisy.

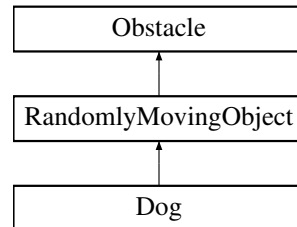
The documentation for this class was generated from the following files:

- [Credits.h](#)
- [Credits.cpp](#)

## 4.8 Dog Class Reference

```
#include <Dog.h>
```

Inheritance diagram for Dog:



### Public Member Functions

- [Dog](#) ()
- [~Dog](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [limit](#) ()
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### Additional Inherited Members

#### 4.8.1 Detailed Description

Klasa dziedziczaca po klasie [RandomlyMovingObject](#), inicjujaca i wyswietlajaca przeszkode w postaci psa.

#### 4.8.2 Constructor & Destructor Documentation

##### 4.8.2.1 Dog()

```
Dog::Dog ( )
```

Konstruktor klasy [Dog](#).

##### 4.8.2.2 ~Dog()

```
Dog::~~Dog ( )
```

Destruktor klasy [Dog](#).

### 4.8.3 Member Function Documentation

#### 4.8.3.1 `getSprite()`

```
const sf::Sprite & Dog::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.8.3.2 `limit()`

```
void Dog::limit ( )
```

Metoda odpowiadajaca za to, by obiekt nie pojawial sie poza polem ruchu gracza.

#### 4.8.3.3 `move()`

```
void Dog::move ( ) [override], [virtual]
```

Metoda odpowiadajaca za ruch obiektu.

Reimplemented from [Obstacle](#).

#### 4.8.3.4 `render()`

```
void Dog::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca obiekt na scenie.

#### 4.8.3.5 `update()`

```
void Dog::update ( )
```

Metoda aktualizujaca pozycje obiektu na scenie.

The documentation for this class was generated from the following files:

- [Dog.h](#)
- [Dog.cpp](#)

## 4.9 Game Class Reference

```
#include <Game.h>
```

## Public Member Functions

- [Game](#) (std::shared\_ptr< [Player](#) > player)
- [~Game](#) ()
- const bool [end](#) () const
- const bool [was\\_played](#) () const
- void [set\\_was\\_played](#) (bool was\_p)
- void [update](#) (sf::RenderTarget \*target)
- void [render](#) (sf::RenderTarget \*target)

### 4.9.1 Detailed Description

Klasa odpowiadająca za przeprowadzenie gry.

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 Game()

```
Game::Game (
    std::shared_ptr< Player > player )
```

Konstruktor klasy [Game](#).

##### Parameters

<i>player</i>	shared pointer do gracza
---------------	--------------------------

#### 4.9.2.2 ~Game()

```
Game::~~Game ( )
```

Destruktor klasy [Game](#).

### 4.9.3 Member Function Documentation

#### 4.9.3.1 end()

```
const bool Game::end ( ) const
```

Metoda zwracająca true, jeśli nastąpił koniec gry. W przeciwnym przypadku zwraca wartość false.

#### 4.9.3.2 render()

```
void Game::render (
    sf::RenderTarget * target )
```

Metoda wywołująca funkcje [render\(\)](#) dla obiektów klas [Player](#), [Collision](#) i [Background](#).

#### 4.9.3.3 set\_was\_played()

```
void Game::set_was_played (
    bool was_p )
```

Metoda pozwalająca ustawić wartość zmiennej `was_played`.

#### 4.9.3.4 update()

```
void Game::update (
    sf::RenderTarget * target )
```

Metoda sprawdzająca, czy nastąpił koniec gry. Jeżeli nie, zostają wywołane funkcje [update\(\)](#) dla obiektów klas [Player](#), [Collision](#) i [Background](#).

#### 4.9.3.5 was\_played()

```
const bool Game::was_played ( ) const
```

Metoda zwracająca `true`, jeśli gra była już wcześniej rozegrana. W przeciwnym przypadku zwraca wartość `false`.

The documentation for this class was generated from the following files:

- [Game.h](#)
- [Game.cpp](#)

## 4.10 GameOver Class Reference

```
#include <GameOver.h>
```

### Public Member Functions

- [GameOver](#) (std::shared\_ptr< [Player](#) > player)
- [~GameOver](#) ()
- void [render](#) (sf::RenderTarget \*target)

#### 4.10.1 Detailed Description

Klasa odpowiadająca za inicjalizację i wyświetlanie okna końca gry.

#### 4.10.2 Constructor & Destructor Documentation

##### 4.10.2.1 GameOver()

```
GameOver::GameOver (
    std::shared_ptr< Player > player )
```

Konstruktor klasy [GameOver](#).

## Parameters

<i>player</i>	shared pointer do gracza
---------------	--------------------------

## 4.10.2.2 ~GameOver()

```
GameOver::~GameOver ( )
```

Destruktor klasy [GameOver](#).

## 4.10.3 Member Function Documentation

## 4.10.3.1 render()

```
void GameOver::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca ekran konca gry.

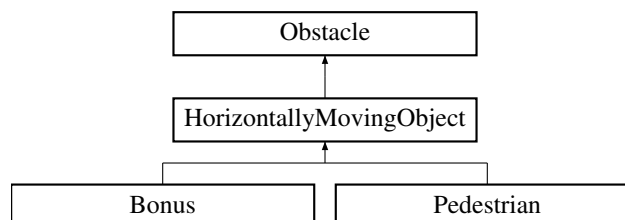
The documentation for this class was generated from the following files:

- [GameOver.h](#)
- [GameOver.cpp](#)

## 4.11 HorizontallyMovingObject Class Reference

```
#include <HorizontallyMovingObject.h>
```

Inheritance diagram for HorizontallyMovingObject:



## Public Member Functions

- [HorizontallyMovingObject](#) ()
- [~HorizontallyMovingObject](#) ()
- void [set\\_points](#) ()
- int [get\\_points](#) ()

## Additional Inherited Members

### 4.11.1 Detailed Description

Klasa dziedziczaca po klasie [Obstacle](#). Po klasie tej dziedzicza obiekty poruszajace sie horyzontalnie.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 HorizontallyMovingObject()

```
HorizontallyMovingObject::HorizontallyMovingObject ( )
```

Konstruktor klasy [HorizontallyMovingObject](#).

#### 4.11.2.2 ~HorizontallyMovingObject()

```
HorizontallyMovingObject::~~HorizontallyMovingObject ( )
```

Destruktor klasy [HorizontallyMovingObject](#).

### 4.11.3 Member Function Documentation

#### 4.11.3.1 get\_points()

```
int HorizontallyMovingObject::get_points ( )
```

Metoda zwracajaca liczbe punktow, ktore zyska gracz, jesli ominie przeszkode.

#### 4.11.3.2 set\_points()

```
void HorizontallyMovingObject::set_points ( )
```

Metoda pozwalajaca ustawic liczbe punktow, ktore zyska gracz, jesli ominie przeszkode.

The documentation for this class was generated from the following files:

- [HorizontallyMovingObject.h](#)
- [HorizontallyMovingObject.cpp](#)



## 4.12 HorizontalMovingObstacle Class Reference

```
#include <HorizontalMovingObstacle.h>
```

The documentation for this class was generated from the following file:

- [HorizontalMovingObstacle.h](#)

## 4.13 Hp Class Reference

```
#include <Hp.h>
```

### Public Member Functions

- [Hp](#) ()
- [~Hp](#) ()
- int [getHp](#) ()
- void [subtractHp](#) (int subtract)
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### 4.13.1 Detailed Description

Klasa odpowiadająca za przechowywanie i operacje na punktach życia gracza.

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 Hp()

```
Hp::Hp ( )
```

Konstruktor klasy [Hp](#).

#### 4.13.2.2 ~Hp()

```
Hp::~Hp ( )
```

Destruktor klasy [Hp](#).

### 4.13.3 Member Function Documentation

#### 4.13.3.1 getHp()

```
int Hp::getHp ( )
```

Metoda zwracająca liczbę punktów życia gracza.

#### 4.13.3.2 render()

```
void Hp::render (
    sf::RenderTarget * target )
```

Metoda odpowiedzialna za wyświetlenie poziomu punktów życia gracza.

#### 4.13.3.3 subtractHp()

```
void Hp::subtractHp (
    int subtract )
```

Metoda odejmująca punkty życia gracza po kolizji z przeszkodą.

#### 4.13.3.4 update()

```
void Hp::update ( )
```

Metoda aktualizująca kształt odpowiadający poziomowi punktów życia gracza.

The documentation for this class was generated from the following files:

- [Hp.h](#)
- [Hp.cpp](#)

## 4.14 Instruction Class Reference

```
#include <Instruction.h>
```

### Public Member Functions

- [Instruction](#) ()
- [~Instruction](#) ()
- void [render](#) (sf::RenderTarget \*target)

#### 4.14.1 Detailed Description

Klasa odpowiadająca za inicjalizację i wyświetlanie instrukcji.

## 4.14.2 Constructor & Destructor Documentation

### 4.14.2.1 Instruction()

```
Instruction::Instruction ( )
```

Konstruktor klasy [Instruction](#).

### 4.14.2.2 ~Instruction()

```
Instruction::~~Instruction ( )
```

Destruktor klasy [Instruction](#).

## 4.14.3 Member Function Documentation

### 4.14.3.1 render()

```
void Instruction::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca instrukcje gry.

The documentation for this class was generated from the following files:

- [Instruction.h](#)
- [Instruction.cpp](#)

## 4.15 Introduce Class Reference

```
#include <Introduce.h>
```

### Public Member Functions

- [Introduce](#) (std::shared\_ptr< [Player](#) > player)
- [~Introduce](#) ()
- void [pollEvents](#) (sf::Event event, sf::RenderTarget \*target)
- void [update](#) (sf::Event event, sf::RenderTarget \*target)
- void [render](#) (sf::RenderTarget \*target)
- bool [check](#) ()
- bool [is\\_correct](#) ()
- void [set\\_player\\_name](#) ()
- void [set\\_confirm](#) (bool active)
- void [set\\_clear](#) (bool active)
- void [set\\_backspace](#) (bool active)

### 4.15.1 Detailed Description

Klasa odpowiadająca za pobranie imienia i nazwiska gracza oraz przekazanie tych danych do klasy [Player](#).

### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 Introduce()

```
Introduce::Introduce (
    std::shared_ptr< Player > player )
```

Konstruktor klasy [Introduce](#).

##### Parameters

<i>player</i>	shared pointer do gracza
---------------	--------------------------

#### 4.15.2.2 ~Introduce()

```
Introduce::~~Introduce ( )
```

Destruktor klasy [Introduce](#).

### 4.15.3 Member Function Documentation

#### 4.15.3.1 check()

```
bool Introduce::check ( )
```

Metoda sprawdzająca, czy wprowadzone przez użytkownika dane są zgodne z szablonem.

#### 4.15.3.2 is\_correct()

```
bool Introduce::is_correct ( )
```

Metoda zwracająca wartość true, jeżeli nazwa gracza została podana poprawnie.

#### 4.15.3.3 pollEvents()

```
void Introduce::pollEvents (
    sf::Event event,
    sf::RenderTarget * target )
```

Metoda odpowiadająca za zamianę znaków z klawiatury w tekst oraz uruchamianiu odpowiednich funkcji w zależności od przycisku wybranego przez użytkownika.

#### 4.15.3.4 render()

```
void Introduce::render (
    sf::RenderTarget * target )
```

Metoda wyświetlająca na ekranie tło, zawartość zmiennej `name` oraz ewentualne ostrzeżenie o wprowadzeniu niepoprawnej wartości zmiennej `name`.

#### 4.15.3.5 set\_backspace()

```
void Introduce::set_backspace (
    bool active )
```

Metoda ustawiająca wartość zmiennej `backspace`.

#### 4.15.3.6 set\_clear()

```
void Introduce::set_clear (
    bool active )
```

Metoda ustawiająca wartość zmiennej `clear`.

#### 4.15.3.7 set\_confirm()

```
void Introduce::set_confirm (
    bool active )
```

Metoda ustawiająca wartość zmiennej `confirm`.

#### 4.15.3.8 set\_player\_name()

```
void Introduce::set_player_name ( )
```

Metoda ustawiająca nazwę gracza na tę podaną przez użytkownika.

#### 4.15.3.9 update()

```
void Introduce::update (
    sf::Event event,
    sf::RenderTarget * target )
```

Metoda wywołująca funkcję [pollEvents\(\)](#) oraz dopisująca do zmiennej name znak wprowadzony przez użytkownika.

The documentation for this class was generated from the following files:

- [Introduce.h](#)
- [Introduce.cpp](#)

## 4.16 list Class Reference

```
#include <list.h>
```

### Public Member Functions

- [list](#) ()
- [~list](#) ()
- [list](#) (Stats \*p)
- [list](#) (const [list](#) &)
- [list](#) ([list](#) &&) noexcept
- [list](#) & [operator=](#) (const [list](#) &s)
- [list](#) & [operator=](#) ([list](#) &&s) noexcept
- void [load](#) (Stats \*&pHead)
- void [try\\_to\\_add](#) (Stats \*&pHead, const std::string &name, const int &score)
- void [add](#) (Stats \*&pHead, const std::string &name, const int &score)
- Stats \* [find](#) (Stats \*&pHead, const std::string &name)
- void [delete\\_stats](#) (Stats \*&pHead)
- void [prepare\\_to\\_print](#) (Stats \*&pHead, std::string &s, int counter)
- void [print](#) (sf::RenderTarget \*target)
- void [save](#) (Stats \*&pHead)
- void [open\\_close\\_file](#) (Stats \*&pHead)
- void [delete\\_record](#) (Stats \*&pHead, Stats \*&s)
- void [actual\\_save](#) (Stats \*&pHead, std::fstream &file, int counter)

### Public Attributes

- Stats \* [pHead](#)

#### 4.16.1 Detailed Description

Klasa odpowiadająca za wykonywanie operacji na klasie [Stats](#) pełniacej role listy jednokierunkowej.

## 4.16.2 Constructor & Destructor Documentation

### 4.16.2.1 list() [1/4]

```
list::list ( )
```

Konstruktor bezargumentowy klasy list.

### 4.16.2.2 ~list()

```
list::~~list ( )
```

Destruktor klasy list.

### 4.16.2.3 list() [2/4]

```
list::list (
    Stats * p )
```

Konstruktor jednoargumentowy klasy list.

### 4.16.2.4 list() [3/4]

```
list::list (
    const list & s )
```

Konstruktor kopiujacy klasy list.

### 4.16.2.5 list() [4/4]

```
list::list (
    list && s ) [noexcept]
```

Konstruktor przenoszacy klasy list.

## 4.16.3 Member Function Documentation

#### 4.16.3.1 actual\_save()

```
void list::actual_save (
    Stats *& pHead,
    std::fstream & file,
    int counter )
```

Metoda zapisuje co najwyzej 20 rekordow z listy ze statystykami.

#### 4.16.3.2 add()

```
void list::add (
    Stats *& pHead,
    const std::string & name,
    const int & score )
```

Metoda dodaje gracza do statystyk.

#### 4.16.3.3 delete\_record()

```
void list::delete_record (
    Stats *& pHead,
    Stats *& s )
```

Metoda usuwa rekord ze statystyk.

#### 4.16.3.4 delete\_stats()

```
void list::delete_stats (
    Stats *& pHead )
```

Metoda usuwa liste ze statystykami.

#### 4.16.3.5 find()

```
Stats * list::find (
    Stats * pHead,
    const std::string & name )
```

Metoda wyszukuje gracza w statystykach i jesli go znajdzie, zwraca wskaznik na odpowiedni element listy. Jesli nie znajdzie, zwraca 0.

#### 4.16.3.6 load()

```
void list::load (
    Stats *& pHead )
```

Metoda odpowiadajaca za zaladowanie statystyk z pliku do struktury.



#### 4.16.3.7 open\_close\_file()

```
void list::open_close_file (
    Stats *& pHead )
```

Metoda odpowiada za otworzenie i wyczyszczenie pliku ze statystykami, wywołanie funkcji [actual\\_save\(\)](#) i zamknięcie pliku.

#### 4.16.3.8 operator=() [1/2]

```
list & list::operator= (
    const list & s )
```

Operator przypisania klasy list.

#### 4.16.3.9 operator=() [2/2]

```
list & list::operator= (
    list && s ) [noexcept]
```

Operator przeniesienia klasy list.

#### 4.16.3.10 prepare\_to\_print()

```
void list::prepare_to_print (
    Stats *& pHead,
    std::string & s,
    int counter )
```

Metoda pobiera z listy maksymalnie 20 elementów, zapisuje je do zmiennej typu string.

#### 4.16.3.11 print()

```
void list::print (
    sf::RenderTarget * target )
```

Metoda wyświetla na ekranie zawartość łańcucha znakowego powstałego w metodzie [prepare\\_to\\_print\(\)](#).

#### 4.16.3.12 save()

```
void list::save (
    Stats *& pHead )
```

Metoda odpowiada za przeprowadzenie operacji zapisu statystyk do pliku.

#### 4.16.3.13 try\_to\_add()

```
void list::try_to_add (
    Stats *& pHead,
    const std::string & name,
    const int & score )
```

Metoda probujaca dodac rekord do statystyk. Jezeli gracz nie uzyskal zadnych punktow lub byl juz wzczesniej wpisany w statystykach i uzyskal liczbe punktow mniejsza od tej w statystykach zapisanych, gracz nie zostaje dodany do statystyk. W przeciwnym przypadku, jesli uzytkownik gral juz wzczesniej i jego nazwa figuruje w liscie ze statystykami, usuwany jest zapis z mniejszym wynikiem koncowym, a nastepnie wywoływana jest funkcja [add\(\)](#). Jesli gracz ma byc wpisany do statystyk po raz pierwszy, od razu wywoływana jest funkcja [add\(\)](#).

### 4.16.4 Member Data Documentation

#### 4.16.4.1 pHead

```
Stats* list::pHead
```

The documentation for this class was generated from the following files:

- [list.h](#)
- [list.cpp](#)

## 4.17 Menu Class Reference

```
#include <Menu.h>
```

### Public Member Functions

- [Menu](#) ()
- [~Menu](#) ()
- void [pollEvents](#) ()
- void [render\\_menu](#) ()
- void [render\\_stats](#) ()
- void [render\\_credits](#) ()
- void [render\\_instruction](#) ()
- void [render\\_game](#) ()
- void [introduce\\_player](#) ()
- void [game\\_over](#) ()
- void [make\\_the\\_game](#) ()
- void [render](#) ()
- void [update](#) ()
- bool [is\\_running](#) ()

## 4.17.1 Constructor & Destructor Documentation

### 4.17.1.1 Menu()

```
Menu::Menu ( )
```

Konstruktor klasy [Menu](#).

### 4.17.1.2 ~Menu()

```
Menu::~~Menu ( )
```

Destruktor klasy [Menu](#).

## 4.17.2 Member Function Documentation

### 4.17.2.1 game\_over()

```
void Menu::game_over ( )
```

Metoda odpowiedzialna za wyswietlenie okna konca gry.

### 4.17.2.2 introduce\_player()

```
void Menu::introduce_player ( )
```

Metoda odpowiedzialna za wyswietlenie okna przedstawienia sie gracza.

### 4.17.2.3 is\_running()

```
bool Menu::is_running ( )
```

Metoda zwracajaca wartosc true, jesli program dziala.

### 4.17.2.4 make\_the\_game()

```
void Menu::make_the_game ( )
```

Metoda odpowiedzialna za wyswietlenie gry.

#### 4.17.2.5 pollEvents()

```
void Menu::pollEvents ( )
```

Metoda uruchamiająca odpowiednie funkcje w zaleznosci od tego, na co kliknie lub jaki klawisz nacisnie gracz.

#### 4.17.2.6 render()

```
void Menu::render ( )
```

Metoda wyswietlajaca odpowiednie obiekty na ekranie.

#### 4.17.2.7 render\_credits()

```
void Menu::render_credits ( )
```

Metoda odpowiedzialna za wyswietlenie przypisow.

#### 4.17.2.8 render\_game()

```
void Menu::render_game ( )
```

Metoda odpowiedzialna za dzialanie gry.

#### 4.17.2.9 render\_instruction()

```
void Menu::render_instruction ( )
```

Metoda odpowiedzialna za wyswietlenie instrukcji.

#### 4.17.2.10 render\_menu()

```
void Menu::render_menu ( )
```

Metoda odpowiedzialna za wyswietlenie menu.

#### 4.17.2.11 render\_stats()

```
void Menu::render_stats ( )
```

Metoda odpowiedzialna za wyswietlenie statystyk.

#### 4.17.2.12 update()

```
void Menu::update ( )
```

Metoda wywołująca funkcje [pollEvents\(\)](#) oraz funkcje [update\(\)](#) dla aktualnie wyświetlanych obiektów.

The documentation for this class was generated from the following files:

- [Menu.h](#)
- [Menu.cpp](#)

## 4.18 Music Class Reference

```
#include <Music.h>
```

### Public Member Functions

- [Music](#) (std::string file)
- [~Music](#) ()
- void [playMusic](#) ()

#### 4.18.1 Detailed Description

Klasa odpowiadająca za odtwarzanie muzyki.

#### 4.18.2 Constructor & Destructor Documentation

##### 4.18.2.1 Music()

```
Music::Music (
    std::string file )
```

Konstruktor klasy [Music](#).

##### Parameters

<i>file</i>	ścieżka do pliku z muzyką
-------------	---------------------------

##### 4.18.2.2 ~Music()

```
Music::~Music ( )
```

Destruktor klasy [Music](#).

### 4.18.3 Member Function Documentation

#### 4.18.3.1 playMusic()

```
void Music::playMusic ( )
```

Metoda odpowiadajaca za odtwarzanie muzyki.

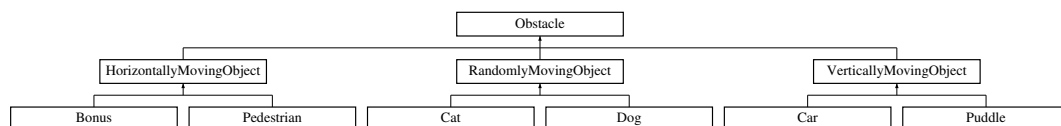
The documentation for this class was generated from the following files:

- [Music.h](#)
- [Music.cpp](#)

## 4.19 Obstacle Class Reference

```
#include <Obstacle.h>
```

Inheritance diagram for Obstacle:



### Public Member Functions

- [Obstacle](#) ()
- [~Obstacle](#) ()
- void [set\\_damage](#) (int damage)
- void [set\\_speed](#) ()
- void [set\\_factor](#) ()
- void [accelerate](#) ()
- float [get\\_speed](#) ()
- int [get\\_damage](#) ()
- int [getMaxObstacles](#) ()
- void [addObstacles](#) ()
- virtual void [move](#) ()

### Public Attributes

- [Random](#) random

### 4.19.1 Detailed Description

Klasa wirtualna, po której dziedziczą klasy odpowiedzialne za tworzenie poszczególnych przeszkód w grze.

### 4.19.2 Constructor & Destructor Documentation

#### 4.19.2.1 Obstacle()

```
Obstacle::Obstacle ( )
```

Konstruktor klasy [Obstacle](#).

#### 4.19.2.2 ~Obstacle()

```
Obstacle::~~Obstacle ( )
```

Destruktor klasy [Obstacle](#).

### 4.19.3 Member Function Documentation

#### 4.19.3.1 accelerate()

```
void Obstacle::accelerate ( )
```

Metoda przyspieszająca ruch przeszkody.

#### 4.19.3.2 addObstacles()

```
void Obstacle::addObstacles ( )
```

Metoda inkrementująca liczbę przeszkód na scenie.

#### 4.19.3.3 get\_damage()

```
int Obstacle::get_damage ( )
```

Metoda zwracająca szkodę wyrządzoną przez przeszkodę.

#### 4.19.3.4 `get_speed()`

```
float Obstacle::get_speed ( )
```

Metoda zwracająca predkosc przeszkody.

#### 4.19.3.5 `getMaxObstacles()`

```
int Obstacle::getMaxObstacles ( )
```

Metoda zwracająca maksymalna liczbe przeszkod na scenie.

#### 4.19.3.6 `move()`

```
void Obstacle::move ( ) [virtual]
```

Metoda wirtualna odpowiadająca za ruch przeszkody.

Reimplemented in [Pedestrian](#), [Cat](#), [Dog](#), [Bonus](#), [Car](#), and [Puddle](#).

#### 4.19.3.7 `set_damage()`

```
void Obstacle::set_damage (
    int damage )
```

Metoda umożliwiajaca ustawienie szkod wyrzadzanych przez przeszkode.

#### 4.19.3.8 `set_factor()`

```
void Obstacle::set_factor ( )
```

Metoda umożliwiajaca ustawienie mnoznika.

#### 4.19.3.9 `set_speed()`

```
void Obstacle::set_speed ( )
```

Metoda umożliwiajaca ustawienie predkosci poruszania sie przeszkody.

### 4.19.4 Member Data Documentation



#### 4.19.4.1 random

`Random` `Obstacle::random`

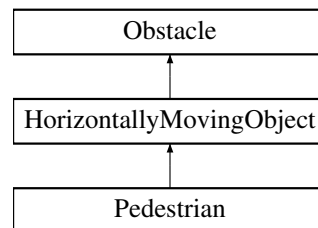
The documentation for this class was generated from the following files:

- [Obstacle.h](#)
- [Obstacle.cpp](#)

## 4.20 Pedestrian Class Reference

```
#include <Pedestrian.h>
```

Inheritance diagram for Pedestrian:



### Public Member Functions

- [Pedestrian](#) ()
- [~Pedestrian](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [pedestrianMovement](#) ()
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### Additional Inherited Members

#### 4.20.1 Detailed Description

Klasa dziedziczaca po klasie [HorizontallyMovingObject](#), inicjujaca i wyswietlajaca przeszkode w postaci pieszego.

#### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 Pedestrian()

```
Pedestrian::Pedestrian ( )
```

Konstruktor klasy [Pedestrian](#).

#### 4.20.2.2 ~Pedestrian()

```
Pedestrian::~~Pedestrian ( )
```

Destruktor klasy [Pedestrian](#).

### 4.20.3 Member Function Documentation

#### 4.20.3.1 getSprite()

```
const sf::Sprite & Pedestrian::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.20.3.2 move()

```
void Pedestrian::move ( ) [override], [virtual]
```

Metoda odpowiadajaca za ruch obiektu.

Reimplemented from [Obstacle](#).

#### 4.20.3.3 pedestrianMovement()

```
void Pedestrian::pedestrianMovement ( )
```

Metoda odpowiadajaca za animowany ruch pieszego.

#### 4.20.3.4 render()

```
void Pedestrian::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca obiekt na scenie.

#### 4.20.3.5 update()

```
void Pedestrian::update ( )
```

Metoda aktualizująca pozycje obiektu na scenie oraz jego animowany ruch.

The documentation for this class was generated from the following files:

- [Pedestrian.h](#)
- [Pedestrian.cpp](#)

## 4.21 Player Class Reference

```
#include <Player.h>
```

### Public Member Functions

- [Player](#) (sf::RenderTarget \*target)
- [~Player](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [setName](#) (const std::string &name)
- std::string [getName](#) ()
- void [slowTheCar](#) ()
- void [move](#) (float x, float y)
- void [input](#) (sf::Event event)
- void [limit](#) (const sf::RenderTarget \*target)
- void [update](#) (const sf::RenderTarget \*target, const sf::Event &event)
- void [render](#) (sf::RenderTarget \*target)
- std::shared\_ptr< [Hp](#) > [getHp](#) ()
- std::shared\_ptr< [Points](#) > [getPoints](#) ()

### 4.21.1 Detailed Description

Klasa przechowująca informacje na temat gracza.

### 4.21.2 Constructor & Destructor Documentation

#### 4.21.2.1 Player()

```
Player::Player (
    sf::RenderTarget * target )
```

Konstruktor klasy [Player](#).

#### 4.21.2.2 ~Player()

```
Player::~~Player ( )
```

Destruktor klasy [Player](#).

### 4.21.3 Member Function Documentation

#### 4.21.3.1 getHp()

```
std::shared_ptr< Hp > Player::getHp ( )
```

Metoda zwracajaca shared pointer do obiektu klasy [Hp](#).

#### 4.21.3.2 getName()

```
std::string Player::getName ( )
```

Metoda zwracajaca nazwe gracza.

#### 4.21.3.3 getPoints()

```
std::shared_ptr< Points > Player::getPoints ( )
```

Metoda zwracajaca shared pointer do obiektu klasy [Points](#).

#### 4.21.3.4 getSprite()

```
const sf::Sprite & Player::getSprite ( ) const
```

Metoda zwracajaca sprite.

#### 4.21.3.5 input()

```
void Player::input (
    sf::Event event )
```

Metoda odczytujaca, jakie klawisze nacisnal uzytkownik i sterujaca na tej podstawie samochodem gracza.

#### 4.21.3.6 limit()

```
void Player::limit (
    const sf::RenderTarget * target )
```

Metoda uniemozliwiajaca graczowi wyjście poza okno gry oraz poza jezdnie.

#### 4.21.3.7 move()

```
void Player::move (
    float x,
    float y )
```

Metoda odpowiedzialna za ruch gracza.

#### 4.21.3.8 render()

```
void Player::render (
    sf::RenderTarget * target )
```

Metoda wyswietlajaca gracza na scenie.

#### 4.21.3.9 setName()

```
void Player::setName (
    const std::string & name )
```

Metoda umozliwiajaca ustawienie nazwy gracza.

#### 4.21.3.10 slowTheCar()

```
void Player::slowTheCar ( )
```

Metoda odpowiedzialna za spowolnienie ruchu gracza.

#### 4.21.3.11 update()

```
void Player::update (
    const sf::RenderTarget * target,
    const sf::Event & event )
```

Metoda wywolujaca funkcje [input\(\)](#) oraz aktualizujaca pozycje gracza na scenie.

The documentation for this class was generated from the following files:

- [Player.h](#)
- [Player.cpp](#)

## 4.22 Points Class Reference

```
#include <Points.h>
```

## Public Member Functions

- [Points](#) ()
- [~Points](#) ()
- void [addPoints](#) (int p)
- int [getPoints](#) ()
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### 4.22.1 Detailed Description

Klasa odpowiadająca za przechowywanie i operacje na punktach gracza.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 [Points](#)()

```
Points::Points ( )
```

Konstruktor klasy [Points](#).

#### 4.22.2.2 [~Points](#)()

```
Points::~~Points ( )
```

Destruktor klasy [Points](#).

### 4.22.3 Member Function Documentation

#### 4.22.3.1 [addPoints](#)()

```
void Points::addPoints (
    int p )
```

Metoda dodająca punkty, jeśli gracz ominiął przeszkodę.

#### 4.22.3.2 [getPoints](#)()

```
int Points::getPoints ( )
```

Metoda zwracająca liczbę punktów gracza.

#### 4.22.3.3 render()

```
void Points::render (
    sf::RenderTarget * target )
```

Metoda odpowiedzialna za wyswietlenie liczby punktow gracza na scenie.

#### 4.22.3.4 update()

```
void Points::update ( )
```

Metoda aktualizujaca liczbe punktow gracza.

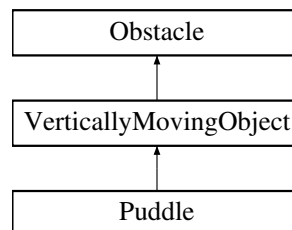
The documentation for this class was generated from the following files:

- [Points.h](#)
- [Points.cpp](#)

## 4.23 Puddle Class Reference

```
#include <Puddle.h>
```

Inheritance diagram for Puddle:



### Public Member Functions

- [Puddle](#) ()
- [~Puddle](#) ()
- const sf::Sprite & [getSprite](#) () const
- void [move](#) () override
- void [update](#) ()
- void [render](#) (sf::RenderTarget \*target)

### Additional Inherited Members

#### 4.23.1 Detailed Description

Klasa dziedziczaca po klasie [HorizontallyMovingObject](#), inicjujaca i wyswietlajaca przeszkode w postaci kaluzy.

## 4.23.2 Constructor & Destructor Documentation

### 4.23.2.1 Puddle()

```
Puddle::Puddle ( )
```

Konstruktor klasy [Puddle](#).

### 4.23.2.2 ~Puddle()

```
Puddle::~~Puddle ( )
```

Destruktor klasy [Puddle](#).

## 4.23.3 Member Function Documentation

### 4.23.3.1 getSprite()

```
const sf::Sprite & Puddle::getSprite ( ) const
```

Metoda zwracająca sprite.

### 4.23.3.2 move()

```
void Puddle::move ( ) [override], [virtual]
```

Metoda odpowiadająca za ruch obiektu.

Reimplemented from [Obstacle](#).

### 4.23.3.3 render()

```
void Puddle::render (
    sf::RenderTarget * target )
```

Metoda wyświetlająca obiekt na scenie.



#### 4.23.3.4 update()

```
void Puddle::update ( )
```

Metoda aktualizująca pozycje obiektu na scenie.

The documentation for this class was generated from the following files:

- [Puddle.h](#)
- [Puddle.cpp](#)

## 4.24 Random Class Reference

```
#include <Random.h>
```

### Public Member Functions

- [int get\\_random\\_number](#) (int begin, int end)
- [Random](#) ()
- [~Random](#) ()

#### 4.24.1 Detailed Description

Klasa odpowiedzialna za generowanie liczb losowych.

#### 4.24.2 Constructor & Destructor Documentation

##### 4.24.2.1 Random()

```
Random::Random ( )
```

Konstruktor klasy [Random](#).

##### 4.24.2.2 ~Random()

```
Random::~~Random ( )
```

Destruktor klasy [Random](#).

#### 4.24.3 Member Function Documentation

#### 4.24.3.1 `get_random_number()`

```
int Random::get_random_number (
    int begin,
    int end )
```

Metoda zwracająca liczbę losową.

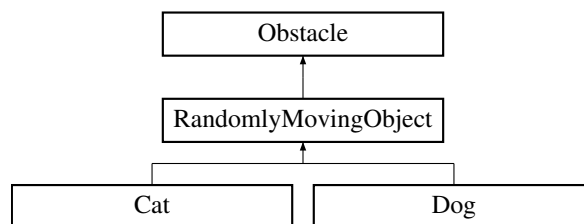
The documentation for this class was generated from the following files:

- [Random.h](#)
- [Random.cpp](#)

## 4.25 RandomlyMovingObject Class Reference

```
#include <RandomlyMovingObject.h>
```

Inheritance diagram for RandomlyMovingObject:



### Public Member Functions

- [RandomlyMovingObject \(\)](#)
- [~RandomlyMovingObject \(\)](#)
- void [set\\_points \(\)](#)
- int [get\\_points \(\)](#)

### Additional Inherited Members

#### 4.25.1 Detailed Description

Klasa dziedzicząca po klasie [Obstacle](#). Po klasie tej dziedziczą obiekty poruszające się w sposób losowy.

#### 4.25.2 Constructor & Destructor Documentation

#### 4.25.2.1 RandomlyMovingObject()

```
RandomlyMovingObject::RandomlyMovingObject ( )
```

Konstruktor klasy [RandomlyMovingObject](#).

#### 4.25.2.2 ~RandomlyMovingObject()

```
RandomlyMovingObject::~~RandomlyMovingObject ( )
```

Destruktor klasy [RandomlyMovingObject](#).

### 4.25.3 Member Function Documentation

#### 4.25.3.1 get\_points()

```
int RandomlyMovingObject::get_points ( )
```

Metoda zwracajaca liczbe punktow, ktore zyska gracz, jesli ominie przeszkode.

#### 4.25.3.2 set\_points()

```
void RandomlyMovingObject::set_points ( )
```

Metoda pozwalajaca ustawic liczbe punktow, ktore zyska gracz, jesli ominie przeszkode.

The documentation for this class was generated from the following files:

- [RandomlyMovingObject.h](#)
- [RandomlyMovingObject.cpp](#)

## 4.26 Sound Class Reference

```
#include <Sound.h>
```

### Public Member Functions

- [Sound](#) (std::string file)
- [~Sound](#) ()
- void [playSound](#) ()

### 4.26.1 Detailed Description

Klasa odpowiadająca za odtwarzanie efektów dźwiękowych.

### 4.26.2 Constructor & Destructor Documentation

#### 4.26.2.1 Sound()

```
Sound::Sound (
    std::string file )
```

Konstruktor klasy [Sound](#).

## Parameters

<i>file</i>	ściezka do pliku z dźwiękiem
-------------	------------------------------

#### 4.26.2.2 ~Sound()

```
Sound::~~Sound ( )
```

Destruktor klasy [Sound](#).

### 4.26.3 Member Function Documentation

#### 4.26.3.1 playSound()

```
void Sound::playSound ( )
```

Metoda odpowiadająca za odtworzenie dźwięku.

The documentation for this class was generated from the following files:

- [Sound.h](#)
- [Sound.cpp](#)

## 4.27 Stats Class Reference

```
#include <Stats.h>
```

### Public Member Functions

- [Stats](#) ()
- [~Stats](#) ()
- [Stats](#) (const [Stats](#) &)
- [Stats](#) ([Stats](#) &&) noexcept
- [Stats](#) & [operator=](#) ([Stats](#) &n)
- [Stats](#) & [operator=](#) ([Stats](#) &&n) noexcept
- void [setVariables](#) (std::string n, int s)
- int [getScore](#) ()
- std::string [getName](#) ()
- void [render](#) (sf::RenderTarget \*target)

## Public Attributes

- [Stats](#) \* [pNext](#)

### 4.27.1 Detailed Description

Klasa pelniaca role listy jednokierunkowej, przechowujacej dane ze statystyk.

### 4.27.2 Constructor & Destructor Documentation

#### 4.27.2.1 Stats() [1/3]

```
Stats::Stats ( )
```

Konstruktor klasy [Stats](#).

#### 4.27.2.2 ~Stats()

```
Stats::~~Stats ( )
```

Destruktor klasy [Stats](#).

#### 4.27.2.3 Stats() [2/3]

```
Stats::Stats (
    const Stats & s )
```

Konstruktor kopiujacy klasy [Stats](#).

#### 4.27.2.4 Stats() [3/3]

```
Stats::Stats (
    Stats && s ) [noexcept]
```

Konstruktor przenoszacy klasy [Stats](#).

### 4.27.3 Member Function Documentation

#### 4.27.3.1 getName()

```
std::string Stats::getName ( )
```

Metoda zwracająca nazwę gracza zapisanego w statystykach.

#### 4.27.3.2 getScore()

```
int Stats::getScore ( )
```

Metoda zwracająca liczbę punktów uzyskanych przez danego gracza zapisanego w statystykach.

#### 4.27.3.3 operator=() [1/2]

```
Stats & Stats::operator= (
    Stats && n ) [noexcept]
```

Operator przeniesienia klasy [Stats](#).

#### 4.27.3.4 operator=() [2/2]

```
Stats & Stats::operator= (
    Stats & n )
```

Operator przypisania klasy [Stats](#).

#### 4.27.3.5 render()

```
void Stats::render (
    sf::RenderTarget * target )
```

Metoda wyświetlająca okno statystyk.

#### 4.27.3.6 setVariables()

```
void Stats::setVariables (
    std::string n,
    int s )
```

Metoda umożliwiająca ustawienie wartości nazwy gracza i punktów.

### 4.27.4 Member Data Documentation

#### 4.27.4.1 pNext

```
Stats* Stats::pNext
```

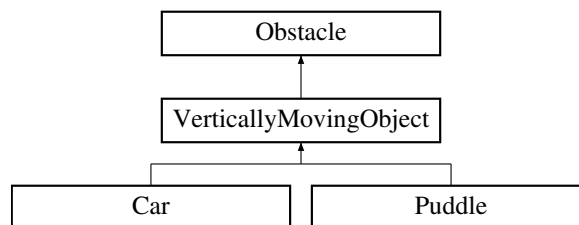
The documentation for this class was generated from the following files:

- [Stats.h](#)
- [Stats.cpp](#)

## 4.28 VerticallyMovingObject Class Reference

```
#include <VerticallyMovingObject.h>
```

Inheritance diagram for VerticallyMovingObject:



### Public Member Functions

- [VerticallyMovingObject](#) ()
- [~VerticallyMovingObject](#) ()
- void [set\\_points](#) ()
- int [get\\_points](#) ()

### Additional Inherited Members

#### 4.28.1 Detailed Description

Klasa dziedziczaca po klasie [Obstacle](#). Po klasie tej dziedzicza obiekty poruszajace sie wertykalnie.

#### 4.28.2 Constructor & Destructor Documentation

##### 4.28.2.1 VerticallyMovingObject()

```
VerticallyMovingObject::VerticallyMovingObject ( )
```

Konstruktor klasy [VerticallyMovingObject](#).



#### 4.28.2.2 ~VerticallyMovingObject()

```
VerticallyMovingObject::~VerticallyMovingObject ( )
```

Destruktor klasy [VerticallyMovingObject](#).

### 4.28.3 Member Function Documentation

#### 4.28.3.1 get\_points()

```
int VerticallyMovingObject::get_points ( )
```

Metoda zwracająca liczbę punktów, które zyska gracz, jeśli ominie przeszkodę.

#### 4.28.3.2 set\_points()

```
void VerticallyMovingObject::set_points ( )
```

Metoda pozwalająca ustawić liczbę punktów, które zyska gracz, jeśli ominie przeszkodę.

The documentation for this class was generated from the following files:

- [VerticallyMovingObject.h](#)
- [VerticallyMovingObject.cpp](#)



## Chapter 5

# File Documentation

### 5.1 Background.cpp File Reference

```
#include "Background.h"
```

### 5.2 Background.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <exception>
```

#### Classes

- class [Background](#)

### 5.3 Bonus.cpp File Reference

```
#include "Bonus.h"
```

### 5.4 Bonus.h File Reference

```
#include "HorizontallyMovingObject.h"
```

## Classes

- class [Bonus](#)

## 5.5 Button.cpp File Reference

```
#include "Button.h"
```

## 5.6 Button.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <string>
#include "Sound.h"
```

## Classes

- class [Button](#)

## 5.7 Car.cpp File Reference

```
#include "Car.h"
```

## 5.8 Car.h File Reference

```
#include "VerticallyMovingObject.h"
```

## Classes

- class [Car](#)

## 5.9 car\_driving.cpp File Reference

```
#include "Menu.h"
```

## Functions

- int [main](#) ()

### 5.9.1 Function Documentation

#### 5.9.1.1 main()

```
int main ( )
```

## 5.10 Cat.cpp File Reference

```
#include "Cat.h"
```

## 5.11 Cat.h File Reference

```
#include "RandomlyMovingObject.h"
```

## Classes

- class [Cat](#)

## 5.12 Collision.cpp File Reference

```
#include "Collision.h"
```

## 5.13 Collision.h File Reference

```
#include <cstdlib>
#include <crtdbg.h>
#include <vector>
#include "Player.h"
#include "Car.h"
#include "Cat.h"
#include "Dog.h"
#include "Bonus.h"
#include "Pedestrian.h"
#include "Puddle.h"
#include "Background.h"
```

## Classes

- class [Collision](#)

## 5.14 Credits.cpp File Reference

```
#include "Credits.h"
```

## 5.15 Credits.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <exception>
```

## Classes

- class [Credits](#)

## 5.16 Dog.cpp File Reference

```
#include "Dog.h"
```

## 5.17 Dog.h File Reference

```
#include "RandomlyMovingObject.h"
```

## Classes

- class [Dog](#)

## 5.18 Game.cpp File Reference

```
#include "Game.h"
```

## 5.19 Game.h File Reference

```
#include <cstdlib>
#include <crtdbg.h>
#include <iostream>
#include "Player.h"
#include "Collision.h"
#include "Background.h"
```

### Classes

- class [Game](#)

## 5.20 GameOver.cpp File Reference

```
#include "GameOver.h"
```

## 5.21 GameOver.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <string>
#include "Player.h"
```

### Classes

- class [GameOver](#)

## 5.22 HorizontallyMovingObject.cpp File Reference

```
#include "HorizontallyMovingObject.h"
```

## 5.23 HorizontallyMovingObject.h File Reference

```
#include "Obstacle.h"
```

## Classes

- class [HorizontallyMovingObject](#)

## 5.24 HorizontalMovingObstacle.h File Reference

### Classes

- class [HorizontalMovingObstacle](#)

## 5.25 Hp.cpp File Reference

```
#include "Hp.h"
```

## 5.26 Hp.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
```

### Classes

- class [Hp](#)

## 5.27 Instruction.cpp File Reference

```
#include "Instruction.h"
```

## 5.28 Instruction.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <exception>
```



## Classes

- class [Instruction](#)

## 5.29 Introduce.cpp File Reference

```
#include "Introduce.h"
```

## 5.30 Introduce.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <string>
#include <iostream>
#include <regex>
#include <exception>
#include "Player.h"
```

## Classes

- class [Introduce](#)

## 5.31 list.cpp File Reference

```
#include "list.h"
```

## 5.32 list.h File Reference

```
#include "Stats.h"
#include <fstream>
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
```

## Classes

- class [list](#)

### 5.33 Menu.cpp File Reference

```
#include "Menu.h"
```

### 5.34 Menu.h File Reference

```
#include "Game.h"  
#include "Button.h"  
#include "list.h"  
#include "Introduce.h"  
#include "GameOver.h"  
#include "Player.h"  
#include "Instruction.h"  
#include "Credits.h"  
#include "Music.h"  
#include <SFML/Audio.hpp>
```

#### Classes

- class [Menu](#)

### 5.35 Music.cpp File Reference

```
#include "Music.h"
```

### 5.36 Music.h File Reference

```
#include <SFML/Audio.hpp>  
#include <string>
```

#### Classes

- class [Music](#)

### 5.37 Obstacle.cpp File Reference

```
#include "Obstacle.h"
```

## 5.38 Obstacle.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <exception>
#include "Random.h"
```

### Classes

- class [Obstacle](#)

## 5.39 Pedestrian.cpp File Reference

```
#include "Pedestrian.h"
```

## 5.40 Pedestrian.h File Reference

```
#include "HorizontallyMovingObject.h"
```

### Classes

- class [Pedestrian](#)

## 5.41 Player.cpp File Reference

```
#include "Player.h"
```

## 5.42 Player.h File Reference

```
#include <cstdlib>
#include <crtdbg.h>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include "Hp.h"
#include "Points.h"
```

## Classes

- class [Player](#)

## 5.43 Points.cpp File Reference

```
#include "Points.h"
```

## 5.44 Points.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
```

## Classes

- class [Points](#)

## 5.45 Puddle.cpp File Reference

```
#include "Puddle.h"
```

## 5.46 Puddle.h File Reference

```
#include "VerticallyMovingObject.h"
```

## Classes

- class [Puddle](#)

## 5.47 Random.cpp File Reference

```
#include "Random.h"
```

## 5.48 Random.h File Reference

```
#include <iostream>
#include <random>
```

### Classes

- class [Random](#)

## 5.49 RandomlyMovingObject.cpp File Reference

```
#include "RandomlyMovingObject.h"
```

## 5.50 RandomlyMovingObject.h File Reference

```
#include "Obstacle.h"
```

### Classes

- class [RandomlyMovingObject](#)

## 5.51 resource.h File Reference

## 5.52 Sound.cpp File Reference

```
#include "Sound.h"
```

## 5.53 Sound.h File Reference

```
#include <SFML/Audio.hpp>
#include <string>
```

### Classes

- class [Sound](#)

## 5.54 statistics.txt File Reference

## 5.55 Stats.cpp File Reference

```
#include "Stats.h"
```

## 5.56 Stats.h File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <exception>
#include <string>
```

### Classes

- class [Stats](#)

## 5.57 VerticallyMovingObject.cpp File Reference

```
#include "VerticallyMovingObject.h"
```

## 5.58 VerticallyMovingObject.h File Reference

```
#include "Obstacle.h"
```

### Classes

- class [VerticallyMovingObject](#)

# Index

- ~Background
  - Background, 7
- ~Bonus
  - Bonus, 9
- ~Button
  - Button, 11
- ~Car
  - Car, 12
- ~Cat
  - Cat, 14
- ~Collision
  - Collision, 16
- ~Credits
  - Credits, 18
- ~Dog
  - Dog, 19
- ~Game
  - Game, 21
- ~GameOver
  - GameOver, 23
- ~HorizontallyMovingObject
  - HorizontallyMovingObject, 24
- ~Hp
  - Hp, 25
- ~Instruction
  - Instruction, 27
- ~Introduce
  - Introduce, 28
- ~Menu
  - Menu, 35
- ~Music
  - Music, 37
- ~Obstacle
  - Obstacle, 39
- ~Pedestrian
  - Pedestrian, 42
- ~Player
  - Player, 43
- ~Points
  - Points, 46
- ~Puddle
  - Puddle, 48
- ~Random
  - Random, 49
- ~RandomlyMovingObject
  - RandomlyMovingObject, 51
- ~Sound
  - Sound, 53
- ~Stats
  - Stats, 54
- ~VerticallyMovingObject
  - VerticallyMovingObject, 56
- ~list
  - list, 31
- accelerate
  - Obstacle, 39
- actual\_save
  - list, 31
- add
  - list, 32
- addObstacles
  - Obstacle, 39
- addPoints
  - Points, 46
- Background, 7
  - ~Background, 7
  - Background, 7
  - faster\_background, 8
  - render, 8
  - update, 8
- Background.cpp, 59
- Background.h, 59
- Bonus, 8
  - ~Bonus, 9
  - Bonus, 9
  - getSprite, 9
  - move, 9
  - render, 9
  - update, 10
- Bonus.cpp, 59
- Bonus.h, 59
- Button, 10
  - ~Button, 11
  - Button, 10
  - change\_activity, 11
  - getSprite, 11
  - is\_active, 11
  - playSound, 11
- Button.cpp, 60
- Button.h, 60
- Car, 12
  - ~Car, 12
  - Car, 12
  - getSprite, 13
  - move, 13
  - render, 13

- update, [13](#)
- Car.cpp, [60](#)
- Car.h, [60](#)
- car\_driving.cpp, [60](#)
  - main, [61](#)
- Cat, [13](#)
  - ~Cat, [14](#)
  - Cat, [14](#)
  - getSprite, [14](#)
  - limit, [14](#)
  - move, [15](#)
  - render, [15](#)
  - update, [15](#)
- Cat.cpp, [61](#)
- Cat.h, [61](#)
- change\_activity
  - Button, [11](#)
- check
  - Introduce, [28](#)
- Collision, [15](#)
  - ~Collision, [16](#)
  - Collision, [16](#)
  - collision, [16](#)
  - delete\_from\_scene, [16](#)
  - make\_harder, [16](#)
  - render, [17](#)
  - spawnObstacles, [17](#)
  - update, [17](#)
  - updateObstacles, [17](#)
- collision
  - Collision, [16](#)
- Collision.cpp, [61](#)
- Collision.h, [61](#)
- Credits, [17](#)
  - ~Credits, [18](#)
  - Credits, [18](#)
  - render, [18](#)
- Credits.cpp, [62](#)
- Credits.h, [62](#)
- delete\_from\_scene
  - Collision, [16](#)
- delete\_record
  - list, [32](#)
- delete\_stats
  - list, [32](#)
- Dog, [19](#)
  - ~Dog, [19](#)
  - Dog, [19](#)
  - getSprite, [20](#)
  - limit, [20](#)
  - move, [20](#)
  - render, [20](#)
  - update, [20](#)
- Dog.cpp, [62](#)
- Dog.h, [62](#)
- end
  - Game, [21](#)
- faster\_background
  - Background, [8](#)
- find
  - list, [32](#)
- Game, [20](#)
  - ~Game, [21](#)
  - end, [21](#)
  - Game, [21](#)
  - render, [21](#)
  - set\_was\_played, [22](#)
  - update, [22](#)
  - was\_played, [22](#)
- Game.cpp, [62](#)
- Game.h, [63](#)
- game\_over
  - Menu, [35](#)
- GameOver, [22](#)
  - ~GameOver, [23](#)
  - GameOver, [22](#)
  - render, [23](#)
- GameOver.cpp, [63](#)
- GameOver.h, [63](#)
- get\_damage
  - Obstacle, [39](#)
- get\_points
  - HorizontallyMovingObject, [24](#)
  - RandomlyMovingObject, [51](#)
  - VerticallyMovingObject, [57](#)
- get\_random\_number
  - Random, [49](#)
- get\_speed
  - Obstacle, [39](#)
- getHp
  - Hp, [25](#)
  - Player, [44](#)
- getMaxObstacles
  - Obstacle, [40](#)
- getName
  - Player, [44](#)
  - Stats, [54](#)
- getPoints
  - Player, [44](#)
  - Points, [46](#)
- getScore
  - Stats, [55](#)
- getSprite
  - Bonus, [9](#)
  - Button, [11](#)
  - Car, [13](#)
  - Cat, [14](#)
  - Dog, [20](#)
  - Pedestrian, [42](#)
  - Player, [44](#)
  - Puddle, [48](#)
- HorizontallyMovingObject, [23](#)
  - ~HorizontallyMovingObject, [24](#)
  - get\_points, [24](#)



- HorizontallyMovingObject, 24
  - set\_points, 24
- HorizontallyMovingObject.cpp, 63
- HorizontallyMovingObject.h, 63
- HorizontalMovingObstacle, 25
- HorizontalMovingObstacle.h, 64
- Hp, 25
  - ~Hp, 25
  - getHp, 25
  - Hp, 25
  - render, 26
  - subtractHp, 26
  - update, 26
- Hp.cpp, 64
- Hp.h, 64
- input
  - Player, 44
- Instruction, 26
  - ~Instruction, 27
  - Instruction, 27
  - render, 27
- Instruction.cpp, 64
- Instruction.h, 64
- Introduce, 27
  - ~Introduce, 28
  - check, 28
  - Introduce, 28
  - is\_correct, 28
  - pollEvents, 28
  - render, 29
  - set\_backspace, 29
  - set\_clear, 29
  - set\_confirm, 29
  - set\_player\_name, 29
  - update, 29
- Introduce.cpp, 65
- Introduce.h, 65
- introduce\_player
  - Menu, 35
- is\_active
  - Button, 11
- is\_correct
  - Introduce, 28
- is\_running
  - Menu, 35
- limit
  - Cat, 14
  - Dog, 20
  - Player, 44
- list, 30
  - ~list, 31
  - actual\_save, 31
  - add, 32
  - delete\_record, 32
  - delete\_stats, 32
  - find, 32
  - list, 31
  - load, 32
  - open\_close\_file, 32
  - operator=, 33
  - pHead, 34
  - prepare\_to\_print, 33
  - print, 33
  - save, 33
  - try\_to\_add, 33
- list.cpp, 65
- list.h, 65
- load
  - list, 32
- main
  - car\_driving.cpp, 61
- make\_harder
  - Collision, 16
- make\_the\_game
  - Menu, 35
- Menu, 34
  - ~Menu, 35
  - game\_over, 35
  - introduce\_player, 35
  - is\_running, 35
  - make\_the\_game, 35
  - Menu, 35
  - pollEvents, 35
  - render, 36
  - render\_credits, 36
  - render\_game, 36
  - render\_instruction, 36
  - render\_menu, 36
  - render\_stats, 36
  - update, 36
- Menu.cpp, 66
- Menu.h, 66
- move
  - Bonus, 9
  - Car, 13
  - Cat, 15
  - Dog, 20
  - Obstacle, 40
  - Pedestrian, 42
  - Player, 44
  - Puddle, 48
- Music, 37
  - ~Music, 37
  - Music, 37
  - playMusic, 38
- Music.cpp, 66
- Music.h, 66
- Obstacle, 38
  - ~Obstacle, 39
  - accelerate, 39
  - addObstacles, 39
  - get\_damage, 39
  - get\_speed, 39
  - getMaxObstacles, 40

- move, 40
  - Obstacle, 39
  - random, 40
  - set\_damage, 40
  - set\_factor, 40
  - set\_speed, 40
- Obstacle.cpp, 66
- Obstacle.h, 67
- open\_close\_file
  - list, 32
- operator=
  - list, 33
  - Stats, 55
- Pedestrian, 41
  - ~Pedestrian, 42
  - getSprite, 42
  - move, 42
  - Pedestrian, 41
  - pedestrianMovement, 42
  - render, 42
  - update, 42
- Pedestrian.cpp, 67
- Pedestrian.h, 67
- pedestrianMovement
  - Pedestrian, 42
- pHead
  - list, 34
- Player, 43
  - ~Player, 43
  - getHp, 44
  - getName, 44
  - getPoints, 44
  - getSprite, 44
  - input, 44
  - limit, 44
  - move, 44
  - Player, 43
  - render, 45
  - setName, 45
  - slowTheCar, 45
  - update, 45
- Player.cpp, 67
- Player.h, 67
- playMusic
  - Music, 38
- playSound
  - Button, 11
  - Sound, 53
- pNext
  - Stats, 55
- Points, 45
  - ~Points, 46
  - addPoints, 46
  - getPoints, 46
  - Points, 46
  - render, 46
  - update, 47
- Points.cpp, 68
- Points.h, 68
- pollEvents
  - Introduce, 28
  - Menu, 35
- prepare\_to\_print
  - list, 33
- print
  - list, 33
- Puddle, 47
  - ~Puddle, 48
  - getSprite, 48
  - move, 48
  - Puddle, 48
  - render, 48
  - update, 48
- Puddle.cpp, 68
- Puddle.h, 68
- Random, 49
  - ~Random, 49
  - get\_random\_number, 49
  - Random, 49
- random
  - Obstacle, 40
- Random.cpp, 68
- Random.h, 69
- RandomlyMovingObject, 50
  - ~RandomlyMovingObject, 51
  - get\_points, 51
  - RandomlyMovingObject, 50
  - set\_points, 51
- RandomlyMovingObject.cpp, 69
- RandomlyMovingObject.h, 69
- render
  - Background, 8
  - Bonus, 9
  - Car, 13
  - Cat, 15
  - Collision, 17
  - Credits, 18
  - Dog, 20
  - Game, 21
  - GameOver, 23
  - Hp, 26
  - Instruction, 27
  - Introduce, 29
  - Menu, 36
  - Pedestrian, 42
  - Player, 45
  - Points, 46
  - Puddle, 48
  - Stats, 55
- render\_credits
  - Menu, 36
- render\_game
  - Menu, 36
- render\_instruction
  - Menu, 36
- render\_menu

- Menu, [36](#)
- render\_stats
  - Menu, [36](#)
- resource.h, [69](#)
- save
  - list, [33](#)
- set\_backspace
  - Introduce, [29](#)
- set\_clear
  - Introduce, [29](#)
- set\_confirm
  - Introduce, [29](#)
- set\_damage
  - Obstacle, [40](#)
- set\_factor
  - Obstacle, [40](#)
- set\_player\_name
  - Introduce, [29](#)
- set\_points
  - HorizontallyMovingObject, [24](#)
  - RandomlyMovingObject, [51](#)
  - VerticallyMovingObject, [57](#)
- set\_speed
  - Obstacle, [40](#)
- set\_was\_played
  - Game, [22](#)
- setName
  - Player, [45](#)
- setVariables
  - Stats, [55](#)
- slowTheCar
  - Player, [45](#)
- Sound, [51](#)
  - ~Sound, [53](#)
  - playSound, [53](#)
  - Sound, [52](#)
- Sound.cpp, [69](#)
- Sound.h, [69](#)
- spawnObstacles
  - Collision, [17](#)
- statistics.txt, [70](#)
- Stats, [53](#)
  - ~Stats, [54](#)
  - getName, [54](#)
  - getScore, [55](#)
  - operator=, [55](#)
  - pNext, [55](#)
  - render, [55](#)
  - setVariables, [55](#)
  - Stats, [54](#)
- Stats.cpp, [70](#)
- Stats.h, [70](#)
- subtractHp
  - Hp, [26](#)
- try\_to\_add
  - list, [33](#)
- update
  - Background, [8](#)
  - Bonus, [10](#)
  - Car, [13](#)
  - Cat, [15](#)
  - Collision, [17](#)
  - Dog, [20](#)
  - Game, [22](#)
  - Hp, [26](#)
  - Introduce, [29](#)
  - Menu, [36](#)
  - Pedestrian, [42](#)
  - Player, [45](#)
  - Points, [47](#)
  - Puddle, [48](#)
- updateObstacles
  - Collision, [17](#)
- VerticallyMovingObject, [56](#)
  - ~VerticallyMovingObject, [56](#)
  - get\_points, [57](#)
  - set\_points, [57](#)
  - VerticallyMovingObject, [56](#)
- VerticallyMovingObject.cpp, [70](#)
- VerticallyMovingObject.h, [70](#)
- was\_played
  - Game, [22](#)