**ChillShips**

Język: C#/Unity

Autorzy: Miłosz Jakóbik i Marceli Błaszczyk

Link do repozytorium: https://github.com/Wh1tEW0lf13/ChillShips

Spis treści

[1. Struktura klas 2](#_Toc169477188)

[2. Fabuła 2](#_Toc169477189)

[3. Cel symulacji 2](#_Toc169477190)

[4. Zasady 3](#_Toc169477191)

[5. Opis klas 3](#_Toc169477192)

[5.1 <<Abstract>>ShipScript 4](#_Toc169477193)

[5.1.2 FastShip 4](#_Toc169477194)

[5.1.3 BigShip 4](#_Toc169477195)

[5.1.4 ShipKiller 4](#_Toc169477196)

[5.2 Baza 4](#_Toc169477197)

[5.3 QuarryAsteroid 5](#_Toc169477198)

[5.4 TrapAsteroid 5](#_Toc169477199)

[5.5 ~~BlackHole~~ 5](#_Toc169477200)

[6. Badania 7](#_Toc169477201)

[6.1 Symulacja numer 1 8](#_Toc169477202)

[6.2 Symulacja numer 2 10](#_Toc169477203)

[6.3 Symulacja numer 3 10](#_Toc169477204)

[6.4 Symulacja numer 4 12](#_Toc169477205)

[6.5 Symulacja numer 5 12](#_Toc169477206)

[6.6 Wnioski 13](#_Toc169477207)

[7. Sugerowane ilości 13](#_Toc169477208)

[8. Autorzy assetów: 13](#_Toc169477209)

# **Struktura klas**

- <<Abstract>> Ship

- ShipKiller  
 - PoorShip  
 - FastShip

- BigShip  
- Base  
- QuarryAsteroid  
- TrapAsteroid

- ~~BlackHole~~

# **Fabuła**

Mamy rok 2137. Ziemia została zniszczona przez głupich studentów, którzy nie potrafili sami o siebie zadbać, co doprowadziło do katastrofy globalnej. Panujący głód i chaos doprowadził do tego, że zebrała się grupka myślących studentów, którzy zbudowali stacje kosmiczne i wylecieli nimi w kosmos, by móc rozwijać się dalej. Po drodze gdy podążali ku gwiazdom adepci PWR wpadli na pomysł, że będą wysyłać komunikaty we wszystkie strony wszechświata. Sygnałem była muzyka z Minecrafta. Wierzyli oni, że w ten sposób uda im się skontaktować z obcymi. Gdy zabrakło im paliwa, musieli wymyślić sposób co zrobić, by nie utknąć w jednym miejscu na wieki. Pomyśleli, że zbudują statki, którymi mogliby się przemieszczać po kosmosie. Jednak nie mieli wystarczających zasobów, by zbudować pojazd kosmiczny. Jednym wyjściem z tej ciężkiej sytuacji, było stworzenie dronów, które wydobywałyby surowce z pobliskich asteroid. Studenci nie mieli jednak wcześniej doświadczenia z budowaniem takich dronów, dlatego działały one w bardzo prymitywny sposób. Z czasem, gdy zbierali więcej surowców, ulepszali technologicznie te drony. Po pewnym czasie zebrali na tyle materiałów, że mogli zbudować statki kosmiczne, którymi mogli się przemieszczać i podbijać inne bazy.

# **Cel symulacji**

Symulacja pokaże, która z dwóch ( ~~lub może więcej~~ ) drużyn ulepszy jako pierwsza bazę do najwyższego stopnia (4 levelu bazy). Statki będą mogły się niszczyć, jak i również asteroidy będą mogły niszczyć statki. Bazy będą niezniszczalne. Jeżeli wszystkie statki zostaną wyeliminowane, lub Baza danej drużyny osiągnie 4 poziom swojej bazy, symulacja zakańcza się zwycięstwem danej drużyny, a na ekranie wyświetla się napis, która drużyna wygrała. Symulacja zostanie przeprowadzona na dwóch światach o zadanych parametrach. Pierwsza symulacja będzie nastawiona na walkę, świat będzie mniejszy i ilość asteroid będzie ograniczona. Druga będzie nastawiona na wydobycie, świat będzie większy, a ilość asteroid będzie ogromna.

1. Zasady

Statki muszą wydobywać z asteroid tytan i żelazo. Jeżeli wydobędą i przyniosą do bazy 5 żelaza, to powstanie kolejny statek. Jeżeli wydobędą odpowiednią ilość tytanu, to ulepszają bazę. Wygrywa ten który najszybciej ulepszy bazę do najwyższego poziomu, lub ten który wyeliminuje statki przeciwnika. Statki mogą się niszczyć nawzajem tylko wtedy gdy się poruszają. Jeżeli wydobywają to nie mogą zostać zniszczone. Z każdym ulepszeniem bazy, będą odblokowywane nowe statki. To jaki statek się wybuduje, będzie losowane. Na mapie oprócz asteroid wydobywczych, będą asteroidy pułapki, które nie różnią się w żaden sposób od zwykłych Asteroid, natomiast gdy statek uderzy w nie (za wyjątkiem KillerShipa), to zostaje unicestwiony, a Asteroida zmienia swoją pozycję. ~~Asteroidy zabójcze będą odbijać się od ścian mapy jak i od asteroid wydobywczych~~. Asteroidy wydobywcze mają określoną ilość surowców. Gdy taka asteroida się skończy, na mapie pojawia się nowa asteroida w innej pozycji. Jedna tura (iteracja) w założeniu to 1 sekunda, ze względu na to że wykorzystujemy silnik Unity.

RedTeam ma 2 razy większą prędkość niż BlueTeam, natomiast Ci drudzy mają 2 razy większą pojemność statków od tych pierwszych.

# **Opis klas**

**Obraz zawierający tekst, zrzut ekranu, diagram, krąg

Opis wygenerowany automatycznie**

Rysunek 1. Diagram przypadków użycia.

5.1 <<Abstract>>ShipScript– poruszają się do losowego punktu X i Y na mapie. Gdy dolecą po drodze do asteroidy to wydobywają złoża do momentu, aż się zapełnią lub wykorzystają zasoby asteroidy. Po wydobyciu wracają do bazy.

# Move() – służy do poruszania się przez statek do danego punku i pozwala wrócić mu do bazy.

# Prepare() – wywołuje SetBasePosition() oraz SetFollowPosition().

# SetBasePositon() – znajduję pozycje bazy na świecie.

# SetFollowPosition() – losuje w jakim kierunku ma się udać statek.

# ShipDestroyer() – służy do niszczenia statków, gdy dojdzie do zderzenia z innym statkiem.  
  
  
5.1.1 PoorShip – dziedziczy wszystkie metody i zmienne z klasy Ship. Jest w stanie wydobywać zasoby.

+IsMining() – służy do pozyskiwania z asteroid złóż.

5.1.2 FastShip – oprócz szybszego poruszania się, ma możliwość lokalizowania najbliższej asteroidy.  
- FindNearestAsteroid() – lokalizuje najbliższą asteroidę.

5.1.3 BigShip – gdy jest w stanie załadować z asteroidy całe złoża, to w ciągu jednej tury zbiera je, jeżeli nie może, to prędkość wydobycia jest taka sama jak w zwykłym statku.  
+IsMining() – polimorfizm względem funkcji z ShipScript, która dodatkowo pozwala zebrać całą asteroidę naraz, jeżeli ta ma mniej zasób niż pojemność statku.

5.1.4 ShipKiller – jedyne złoża jakie zbiera, to te które zbierze z pokonanych wrogów. Nie można go zniszczyć, chyba że zaatakuje go inny ShipKiller. Niszczy statki, które nie wydobywają nic w danym momencie.

- Circle Collider2D() - jeżeli napotka wroga w promieniu 20 od siebie, zaczyna zmierzać w jego pozycję. (circle collider znajduje się w obiekcie childShipKiller z powodu niemożności dodania dwóch colliderów do jednego obiektu)

- Box Collider2D() - gdy obiekt wejdzie w kolizję z statkiem przeciwnika - niszczy go i przejmuje zgromadzone przez niego zasoby.

- Follow() - funkcja zastępująca podstawowe Move() i jest wykorzystywana podczas podążania za przeciwniem.

- Prepare() - polimorfizm względem funkcji z ShipScript, dodatkowo zapamiętuje tag przeciwnika w zmiennej enemyTag.

5.2 Baza – do niej są składowane złoża i z niej statki startują. Jeżeli statek przeciwnika w nią wleci, to baza zbiera mu połowę surowców oraz przejmuje kontrole nad statkiem. Nie może zostać zniszczona.

+ ShipSpawner() - zarządza tworzeniem statków.

- PoorShipSpawner() - tworzy statek klasy PoorShip.

- BigShipSpawner() - tworzy statek klasy BigShip.

- FastShipSpawner() - tworzy statek klasy FastShip.

- ShipKillerSpawner() - tworzy statek klasy ShipKiller.

- LvlUp() - sprawdza ile tytanu zgromadziła baza i na tej podstawie zwiększa jej poziom. (Poniżej konkretne poziomy baz).

**Baza Lvl.2** – Dodatkowo pozwala tworzyć BigShip i FastShip (szansa na stworzenie PoorShip/BigShip/FastShip to 50/25/25).

**Baza Lvl.3** – pozwala dodatkowo stworzyć BigShip, FastShip oraz KillerShip (sznasa na stowrzenie PoorShip/BigShip/FastShip/KillerShip to 40/25/25/10).

**Baza Lvl.4** – kończy symulacje.

5.3 QuarryAsteroid – Z nich statki wydobywają zasoby. Po wydobyciu znikają i pojawiają się w innym losowym miejscu.

- ResetPosition() – służy do zmienienia swojej pozycji i wywołania funkcji AsteroidCapacity().

- AsteroidCapacity() – losowo ustawia ilość żelaza lub ilość tytanu dostępnego na asteroidzie.

5.4 TrapAsteroid – są to asteroidy pułapki. Gdy statek na nie wleci, niszczy się, a asteroida pojawia się w innym miejscu na mapie.

-ResetPosition() – służy do zmienienia swojej pozycji, jak i również zniszczenia statku, który w nią wleci.

-DestroyShip(GameObject shipInfo) – niszczy statek, który wleci w asteroidę. shipInfo to obiekt, który wleciał w asteroidę. Jest on wykorzystywane do tworzenia statystyk.

5.5 ~~BlackHole~~ **~~–~~** ~~Jest losowo umieszczona na świecie. Jeżeli jakiś statek wleci w nią, to teleportuje się w miejsce gdzie umieszczony jest childBlackHole. Stamtąd kontynuuje swoją przygodę.~~

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 2. Diagram obiektów.

PoorShip wykorzystuje funkcję Move() i zmiennej isComingBack, by wrócić do bazy. Gdy wróci, baza wykorzystuje funkcję GetResources(Collider2D col), która niszczy ten obiekt i pozwala pobrać z PoorShipa zasoby oraz zwiększa stackShip o jeden. Z col są pobierane informacje, ile PoorShip ma złóż i jest w stanie przez to sobie baza je przypisać. Do stworzenia FastShipa jest wykorzystana funkcja ShipSpawner(), w której jest losowane, jaki statek ma zostać stworzony. Gdy wylosuje FastShipa, to wywołuje funkcję FastShipSpawner(string tag), która tworzy takowy obiekt na scenie. tag jest wykorzystany do tego, by przypisać konkretny kolor i tag do statku. Gdy FastShip wykorzystuje polimorficzną funkcję Prepare(), w której ustala pozycję bazy i wykorzystuje funkcję FindNearestAsteroid() do namierzenia nabliższej asteroidy, po czym udaje się w jej stronę. Chwilę przed tym jak zostaje stworzony FastShip, tworzy się Asteroida niedaleko bazy czerwonych. Używa ona AsteroidCapacity(), która losuje pozycję, w jakiej Asteroida ma się pojawić oraz ile ma mieć żelaza bądź tytanu.

Obraz zawierający tekst, zrzut ekranu, diagram

Opis wygenerowany automatycznie

Rysunek 3. Diagram obiektów.

BigShip wykorzystuje funkcję Move(), by udać się w stronę losowo wybranych koordynatów. Po drodze spotyka TrapAsteroidę, która wykorzystuje funkcję DestroyShip(GameObject shipInfo), by zniszczyć statek, który w nią wleci. W tym momencie z shipInfo pobierany jest tag, po to by dodać do statystyk to, że asteroida zniszczyła niebieski statek.

Obraz zawierający tekst, zrzut ekranu, kreskówka, ilustracja

Opis wygenerowany automatycznie

Rysunek 4. Diagram obiektów.

Gdy BlueShipKiller3 zbliży się na tyle do RedPoorShip3, że Circle Collider2D() wykryje kolizję z przeciwnikiem, zaczyna on za nim podążać. Gdy czerwony statek wejdzie z Box Collider2D() niebieskiego statku w kolizję, wtedy ShipKiller zabiera 10 irona od krwistego statku i wywoływana jest funkcja ShipDestroyer(), która niszczy statek przeciwnika.

# Badania

Różnice pomiędzy drużynami opisano w punkcie 4. Zasady. Zgodnie z zapisami tam zamieszczonymi, przed rozpoczęciem badań zwiększono shipSpeed, do wartości 60 z podstawowych 30, wszystkich statków drużyny czerwonej z wyłączeniem statków klasy FastShip, gdzie dokonano korekty z 40 na 80. Z kolei w drużynie niebieskiej zwiększono wartość zmiennej capacity: dla obiektów klasy BigShip z 20 do 40, a dla pozostałych statków drużyny niebieskiej z podstawowych 10 do 20.

Parametrami zmienianymi pomiędzy symulacjami była wielkość mapy oraz ilość asteroid. W każdym z przypadków kształt mapy pozostawał kwadratowy.

Badanymi parametrami w symulacjach były:

* winner – przyjmuje wartość Blue lub Red i określa zwycięzcę symulacji.
* winCase – określa okoliczności zwycięstwa i przyjmuje wartości Resources, gdy zwycięzca zgromadził 500 tytanu lub KilledEnemy, gdy przeciwnik nie ma już żadnych statków.
* blueKills – przyjmuje wartości całkowitoliczbowe określa, ile statków przeciwnika zniszczyły statki drużyny niebieskiej.
* redKills – przyjmuje wartości całkowitoliczbowe określa ile statków przeciwnika zniszczyły statki drużyny czerwonej.
* asteroidKillsBlue – przyjmuje wartości całkowitoliczbowe określa, ile statków drużyny niebieskiej zostało zniszczonych przez obiekty klasy TrapAsteroid.
* asteroidKillsred – przyjmuje wartości całkowitoliczbowe określa, ile statków drużyny czerwonej zostało zniszczonych przez obiekty klasy TrapAsteroid.
* blueTytan – przyjmuje wartości całkowitoliczbowe określa, ile tytanu zgromadziła w swojej bazie drużyna niebieska.
* redTytan – przyjmuje wartości całkowitoliczbowe określa, ile tytanu zgromadziła w swojej bazie drużyna czerwona.
* simulationTime – przyjmuje wartości całkowitoliczbowe określa, ile sekund trwała symulacja.

## **6.1 Symulacja numer 1**

Rozmiar mapy: 200 x 200

Liczba asteroid: 20

Wykres 1. Wykres zwycięstw.

Na mapie o rozmiarach podanych wyżej wydać wyraźną przewagę drużyny niebieskiej, która w znaczącej większości przypadków wygrywała z powodu zgromadzenia 500 tytanu. Drużyna czerwona w każdą ze swoich wygranych zawdzięczała zniszczeniu wszystkich statków przeciwnika.

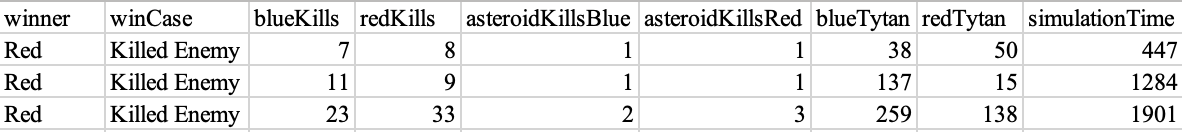


Tabela 1. Zwycięstwa drużyny czerwonej

Zgodnie ze wskazaniami Tabeli 1 tylko w jednym przypadku liczba zabójstw drużyny czerwonej przewyższała w wyraźny sposób liczbę zabójstw drużyny czerwonej. Ponadto każdy z przypadków w którym to redTeam odnosił zwycięstwo dotyczy symulacji, która trwała stosunkowo krótko (średnia czasu trwania przy zadanych parametrach to 2780,875). W każdym z tych przypadków asteroidy pułapki odpowiedzialne są za zniszczenia bliskie zeru. Można na tej podstawie wnioskować, że drużyna czerwona miała szansę wygrać w przypadku, gdy obie bazy pojawiały się blisko siebie i statki niebieskie niszczone były podczas kontaktu z bazą przeciwnika. Ilość takich zdarzeń nie jest przedmiotem badań, jednak biorąc pod uwagę krótki czas trwania symulacji i zbliżone do siebie ilości zabójstw obu drużyn można założyć, że większość zniszczeń pochodzi właśnie z tych kontaktów.

Wykres 2. Warunki zwycięstw.

Drużyna niebieska zaledwie kilka razy wygrała z powodu wyniszczenia przeciwnika. Średnia czasów takich symulacji wynosiła 2322,875, a mediana 2279, nie jest możliwe na tej podstawie wyznaczenie jakiejkolwiek korelacji. Nie można wyznaczyć jakiejkolwiek zależności pomiędzy liczbą zabójstw.

A table with numbers and letters

Description automatically generated

Tabela 2. Zwycięstwa drużyny niebieskiej, spowodowane zwycięstwem przeciwnika.

## **6.2 Symulacja numer 2**

Rozmiar mapy: 200x200

Liczba asteroid: 50

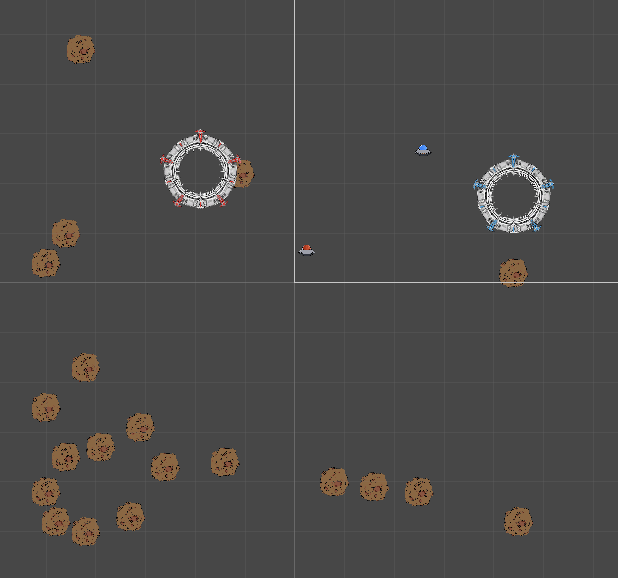
W tym przypadku przepadano jak na wyniki symulacji wpływa ilość asteroid na niezmienionym obszarze. Zebrane dane potwierdzają oczekiwania jakie można by mieć to takiej modyfikacji, mianowicie tym razem drużyna czerwona nie wygrała ani razu, a przewaga w wydobyciu statków niebieskich uwidoczniła się jeszcze bardziej, ponieważ wygrana poprzez zniszczenie przeciwnika wystąpiła tylko raz. Średni czas trwania symulacji w tym przypadku to 2350,8.

## **6.3 Symulacja numer 3**

Rozmiar mapy: 50x50

Liczba asteroid: 20

Badania dla tak małych danych mogą być bardzo nieprecyzyjne, ponieważ dla w tak małym obszarze bazy zajmują znaczący obszar, co może prowadzić do przyblokowania dostępu do części asteroid dla jednej z drużyn co wyraźnie widać na rysunku 5. Duża część asteroid zgromadzona jest w tym przypadku w pobliżu lewej i dolnej krawędzi mapy. Drużyna niebieska, która ma większą pojemność surowcową i odpowiada za większość wydobycia nie jest w stanie dotrzeć do nich, ponieważ napotyka na swojej drodze bazę czerwoną oraz stale napływające z niej statki, z którymi wchodzi w kolizję. Doświadczamy zatem swoistego monopolu asteroid dla jednej z drużyn w dodatku tej, która jest w stanie zmieścić mniej surowców w swoich statkach, wskutek czego asteroidy są wyeksploatowane wolniej, i nie zmieniają swojej pozycji przez długi czas.



Rysunek 5. Nagromadzenie asteroid poza zasięgiem jednej z drużyn.

Rozpatrując ogólną liczbę zwycięstw można zauważyć przewagę drużyny niebieskiej, jednak w przeciwieństwie do poprzedniej symulacji, tym razem zarówno drużyna czerwona odnosiła znaczące sukcesy. Większość zwycięstw w tym przypadków zostało odniesione wskutek zniszczenia jednej z drużyn, a za każde zwycięstwo odniesione wskutek przewagi surowcowej odpowiadają niebiescy.

Średnia czasu trwania symulacji w tym przypadku wynosi 3317,979. Znacząco zawyża ten wynik jednak czas trwania symulacji, w których wygrana odbyła się na surowce, ponieważ ich średnia to aż 5629,222. Wynika to z ilości statków, które miała każda z drużyn. Duża ich ilość była niszczona, przez co mniej było w stanie wydobywać surowce. Mogą potwierdzić to dwie statystyki: ilość wygranych przez zniszczenie przeciwnika oraz ilość statków niszczonych przez asteroidy pułapki. Obie te wartości są większe w tej symulacji niż w poprzednich.

|  |  |
| --- | --- |
| Symulacja 6.2 | Symulacja 6.3 |
| 10,457 | 22,276 |

Tabela 3. Średnia ilość zniszczonych statków przez asteroidy w poszczególnych symulacjach

Wykres 3. Wykres zwycięstw.

Wykres 4. Warunki zwycięstw.

6.4 Symulacja numer 4

Rozmiar mapy 500x500

Liczba asteroid 100

W tym przypadku, identycznie jak w symulacji 6.2, wszystkie symulacje wygrała drużyna niebieska, a tylko raz (na 83 powtórzenia) udało jej się to poprzez pokonanie przeciwnika. W tym szczególnym przypadku zarówno w kolizjach ze statkami jak i z asteroidami więcej zniszczonych statków zostało niebieskich. Można na tej podstawie wnioskować, że większe znaczenie niż szybkość poruszania ma przewaga liczebna.



Tabela 4. Szczególny przypadek zwycięstwa na wyniszczenie

Średni czas trwania symulacji to 6693.

## **6.5 Symulacja numer 5**

Rozmiar mapy: 1000x1000

Liczba asteroid: 400

Podobnie jak w pozostałych symulacjach na dużych mapach przewagę miała drużyna niebieska, która wygrała w 62 z 63 symulacji. Każda z symulacji zakończyła się wygraną na surowce, ponieważ na tak rozległej mapie kolizje między statkami były rzadkością. Wyjątkiem jest jedna iteracja symulacji, w której to drużyna czerwona odniosła zwycięstwo na surowce. Jest to wynik szczęśliwego rozmieszczenia asteroid dla tej drużyny i trafnego losowania docelowych koordynatów dla statków tej drużyny.



Tabela 5. Szczególny przypadek zwycięstwa drużyny czerwonej.

## **6.6 Wnioski**

• Drużyna niebieska ma przewagę w każdej mapie o wielkości przynajmniej 200 x 200.

• Wyniki symulacji na mapie o wielkości 50 x 50 są z grubsza losowe.

• Zwiększona szybkość okazuje się nie być żadnym atutem dla drużyny czerwonej, ponieważ całkowicie losowy system doboru destynacji dla większości statków sprawia, że większość z nich nigdy nie dotrze do żadnej asteroidy, więc kluczowe jest zebranie jak największej liczby surowców, gdy uda się już statkowi na taką natrafić. Ponadto na małych mapach zwiększona szybkość ruchu może doprowadzi do szybszej kolizji statków z wrogą bazą lub asteroidami pułapkami, które niszczą statki.

• W celu zrównoważenia symulacji należało by dodać system życia, lub bardziej zaawansowanej walki, jednak rozwiązania te wykraczają poza początkowe założenia projektu.

# Sugerowane ilości

- Na start po 15 statków. Brak limitu statków.

- Dwie bazy, niebieska i czerwona.

- Ilość asteroid wydobywczych można ustawić w zależności od wielkości świata.

- Asteroidy pułapki – tyle ile Asteroid wydobywczych/10.

# 8. Autorzy assetów:

Oliwia Harnik – statki oraz asteroidy

<https://www.pngwing.com/en/free-png-hftji> - baza

<https://www.youtube.com/watch?v=XuZDeT8zI5c> – muzyka

<https://www.youtube.com/watch?v=tUlthCngK9U> – wybuch statków