Działanie Serwera:

- Serwer przechowuje i udostępnia informacje konfiguracyjne: listę najlepszych wyników, definicje wyglądu plansz, poziomów gry, scenariusza gry oraz wszelkie pozostałe parametry aplikacji.

- Serwer obsługuje dowolną liczbę klientów.

- Serwer sieciowy jest niezależną, bezobsługową, parametryzowaną aplikacją.

- Aplikacja w wersji sieciowej działa także bez serwera – z wykorzystaniem lokalnych plików konfiguracyjnych.

- Adres serwera i numer portu są parametrami aplikacji klienta.

- Każdy klient może osobno połączyć się z serwerem. Obsługiwani są na oddzielnych wątkach.

Działanie klienta:

- Po uruchomieniu aplikacji użytkownik może wybrać działanie z lokalnych plików konfiguracyjnych lub zalogować się podając numer portu i adres hosta.

- Po wyrażeniu chęci przez użytkownika gry online klient wysyła do serwera żądanie aby otrzymać pliki konfiguracyjne.

- Zakończenie przez użytkownika rozgrywki skutkuję wysyłaniem przez klienta żądanie do serwera aby wynik końcowy został wpisany na listę.

Opis podstaw działania protokołu:

- Typ tekstowy.

- Dane klient-serwer, serwer-klient wysyłane są za pomocą pojedynczych linii tekstu. Poszczególne parametry rozdzielane są znakiem “ ”, jeśli składają się wyłącznie z jednej wartości. Jeśli natomiast w wysyłanej wiadomości znajdują się także parametry o wielu wartościach (np. współrzędne x-owe) to ciąg wartości rozdzielany jest za pomocą “ ”, a poszczególne parametry “ - ”. Wiadomość zakończona jest znakiem końca linii “ /n ”.
  Przykładowo: 800 600 1280 720 730 600 300 400 800 600 280 190 4 50
  oraz 4-0 260 410 410 615 615 576 572 465 465 617 617 658 658 771
  1280-720 576 576 430 430 513 513 479 479 645 645 604 604 645 645
  720-300-500-80-20-1-300

Sposób działania protokołu:

- Prośba klienta o wysłanie wymiarów:
  okien(int), liczby buforowań(int), wielkości bonusu paliwa(int).
  C: getConfig => S
  Przykładowo:
  From client: getConfig

- Serwer odpowiada na prośbę klienta wysyłając 14 parametrów:
  wymiary okien(int), liczby buforowań(int), wielkości bonusu paliwa(int). Są one rozdzielane za pomocą “ ”, a zakończone znakiem końca linii “ /n ”
  S: loadConfig=> C
  Przykład:
  Server respond: 800(wysokość okna głównego) 600(szerokość okna głównego) 1280(wysokość okna rozgrywki) 720(szerokość okna rozgrywki) 730(wysokość okna pomocy) 600(szerokość okna pomocy) 300(wysokość okna Rankingu) 400(szerokość okna Rankingu) 500(wysokość okna wyboru rodzaju rozgrywki) 600(szerokość okna wyboru rodzaju rozgrywki) 280(wysokość okna nazwy gracza) 190(szerokość okna nazwy gracza) 4(liczba buforowań) 50(bonus paliwa)

- Klient prosi serwer o wysłanie parametrów:
  grawitacja(int), wysokość statku(int), szerokość statku(int), maksymalna prędkość statku(int), przyśpieszenie w lewo(int), przyśpieszenie w prawo(int) oraz przyspieszenie w górę(int).
  C: getLander=> S
  Przykładowo:
  From client: getLander

- Serwer odpowiada na prośbę klienta wysyłając 7 parametrów: grawitacja(int), wysokość statku(int), szerokość statku(int), maksymalna prędkość statku(int), przyśpieszenie w lewo(int), przyśpieszenie w prawo(int) oraz przyspieszenie w górę(int). Są one rozdzielane za pomocą " " a zakończone znakiem końca linii " /n ".
  S: loadLander=> C
  Server respond: 0.02(siła grawitacji) 32(wysokość lądownika) 32(szerokość lądownika) 0.9(maksymalna prędkość) 0.01(przyśpieszenie w lewo) 0.01(przyśpieszenie w prawo) 0.05(przyśpieszenie w górę)

- Klient prosi serwer o wysłanie parametrów mapy o podanym numerze indeksu:
  liczba poziomów(int), położenie lądowisk(int), rozmiary lądowisk(int), bonus lądowiska(int), budowę mapy(int), wymiary paliwa(int), rozmiar paliwa(int). Są one rozdzielane za pomocą " - " a zakończone znakiem końca linii " /n ". Po każdym ukończonym poziomie klient ponawia prośbę o przesłanie parametrów mapy z indeksem+1.
  C: getMap=> S
  Przykładowo:
  "From client: getMaps-1"

- Serwer odpowiada na prośbę klienta wysyłając parametry: liczba leveli(int), budowę mapy(int), położenie lądowisk(int), rozmiary lądowisk(int), bonus lądowiska(int), wymiary paliwa(int), rozmiar paliwa(int). Wszystkie te cechy przesyłane są w tym formacie:

  /*(...Shape to tzw. hitboxy - niewidoczne obiekty o tym samym rozmiarze i położeniu co obiekt widziany na ekranie, służące wyłącznie do wykrywania kolizji (instancje klasy Shape))*/

Server respond: 4 (liczba poziomów)- 0 260 410 410 615 615 576 572 465 465 617 617 658 658 771 1280 (współrzędne X-owe terenu)- 720 576 576 430 430 513 513 479 479 645 645 604 604 645 645 720 (współrzędne Y-owe)-300(współrzędne X-owe pierwszego lądowiska)-500(współrzędne Y-owe pierwszego lądowiska)-80(szerokość pierwszego lądowiska)-20(wysokość pierwszego lądowiska)- 1(mnożnik punktowy przy wylądowaniu na danym lądowisku)- 300(współrzędne X-owe Shape'a dla pierwszego lądowiska)-500(współrzędne Y-owe Shape'a dla pierwszego lądowiska)-20(wysokość Shape'a dla pierwszego lądowiska)- 80(szerokość Shape'a dla pierwszego lądowiska)-520(współrzędne X-owe drugiego lądowiska)-635(współrzędne Y-owe drugiego lądowiska)-80(szerokość drugiego lądowiska)-10(wysokość drugiego lądowiska)-3(mnożnik punktowy dla drugiego lądowiska)-520(współrzędne X-owe Shape'a dla drugiego lądowiska)-635(współrzędne Y-owe Shape'a dla drugiego lądowiska)-10(wysokość Shape'a dla drugiego lądowiska)-80(szerokość Shape'a dla drugiego lądowiska)-1000(współrzędne X-owe lądownika)-80(współrzędne Y-owe lądownika)-150(dostępna ilość paliwa)-2(szybkość paliwa)-32(szerokość paliwa)-32(wysokość paliwa)-500(współrzędne X-owe paliwa)-200(współrzędne X-owe paliwa)

Poszczególne elementy oddzielone są "-", a znakiem " " rozdziela się współrzędne mapy.
S: loadMap=> C

- Klient prosi serwer o wysłanie rankingu(String). Wyniki rozdzielone są znakiem " - ".
  C: getRanking=> S
  Przykładowo:
  From client: getRanking

- Serwer odpowiada na prośbę klienta wysyłając ranking(String).
  S: loadRanking=> C

  Przykładowy komunikat serwera:
  Server respond: babass 6512-Mikołaj 4440-fgu 4326-Paweł 4185-kasztan12 1998-kaka 1940-jk 1761-kkak 1290-emil 1222-kasia 45

- Klient po zakończonej rozgrywce przesyła do serwera nazwę (String) oraz uzyskany wynik (int). Poszczególne człony rozdzielane są “-”. Schemat przesyłanych danych: “saveScore” + “-” +name +“-” + score.
  C: sendScoreToServer=> S
  Przykładowo:
  saveScore-Mikołaj-4440

- Serwer odbiera dane od klienta(komendę, imię gracza(String), wynik(int)). O tym, czy przesłany przez gracza wynik znajdzie się w pliku pośród 10 rekordów, decyduje algorytm po stronie serwera. Sortuje on wynik i zapisuje do pliku (jeśli będzie lepszy od któregokolwiek z wpisanych tam rekordów). Serwer zwróci nową, zaktualizowaną listę dopiero po ponownym wysłaniu przez klienta komendy getRanking.
  C: saveScore => S
  Przykładowo:
  “From client: saveScore-Mikołaj-4440”.