Programowanie Komputerów [PK2]
Wydział Automatyki, Elektroniki i Informatyki
semestr 2 studia stacjonarne
2020/2021
album 297925
02.07.2021r.

Wojciech Dranka

Projekt: Gra w statki

Polecenie:

Stworzyć grę w statki opartą o programowanie obiektowe w języku c++

Analiza Problemu:

W grze możemy wyróżnić takie elementy jak: statek, plansza, gracz i strzał.
Obiektowość można wykorzystać w „budowie" gry. Np. Obiekt statek będzie się składał z mniejszych obiektów „część statku". Lub obiekt „plansza" może się składać z obiektów „pole. Oczywiście każdy z graczy będzie miał swoją „planszę" i swoje „statki".
Polimorfizm można zastosować w określaniu czy dane pole jest trafieniem czy wodą itp.
Ja jednak zdecydowałem się na zastosowanie polimorfizmu w sprawdzaniu warunku końca gry przez gracza i komputer.

Opis Interfejsu:

- Całość będzie aplikacją konsolową (rozważam jednak dodanie biblioteki graficznej)

- gra będzie możliwa tylko z „myślącym" botem

- na samym początku pojawi się menu z opcjami 1. Graj, 2. Zasady, 3. Leaderboard.
Przy wyborze opcji pierwszej trzeba będzie podać swój nick

- rozgrywka rozpocznie się od ustawienia statków na planszy. Aplikacja wskaże, które statek należy teraz rozmieścić. Użytkownik zostanie poproszony o podanie odpowiedniej ilości współrzędnych. Plansza będzie się odświeżała z każdym statkiem

- gracz będzie widział planszę ze swoimi statkami i strzałami przeciwnika oraz planszę z polami, w które sam strzelał. Obie będą aktualizowane z każdym strzałem

- każdy będzie dysponował: 1 statkiem 4 masztowym, 2 – 3, 3 – 2 i 4 – 1

- dodatkowo statki muszą mieć szerokość 1 kratki i nie mogą się „dotykać" (nawet na ukos)

- każde z pól będzie oznaczone innym znakiem (woda, trafienie, nasz statek, wrak, pudło/obrys zatopionego statku)

- na końcu zostanie pokazany wynik oparty na ilości potrzebnych strzałów.

- do pliku tekstowego zostanie zapisany początkowe ustawienie gracza, komputera, końcowe plansze obu w raz z zaznaczonymi strzałami przeciwnika, oraz informacja o każdym strzale (współrzędne, trafienie czy pudło, czy zatopienie).

- po każdej grze do osobnego pliku tekstowego będzie zapisywana tabela liderów.
(będzie ona również pobierana przy początku każdej gry.)

Testowanie:

Funkcje odpowiadające za zatapianie, obliczanie strzału przez bota czy umieszczanie statków testowałem ręcznie. Dlatego aby przyśpieszyć ten proces napisałem funkcje pomocnicze. Np. gdy chciałem sprawdzić, czy Bot jest „inteligentny" i potrafi zatopić moją flotę w jak najkrótszym czasie, nie potrzebowałem za każdym razem ustawiać statków tak jak by to robił użytkownik przy starcie nowej gry. Dlatego napisałem funkcję. Która tworzy obiekty klasy Ship (statki) i już w konstruktorze podaje potrzebne informacje o umieszczeniu statku. Dzięki temu nie musiałem za każdym razem przechodzić przez proces ustawiania statków, który był trochę czasochłonny.

Wycieki pamięci sprawdziłem przy pomocy specjalnej funkcji, zwracające odpowiednie informacje przy debugowaniu. Po ostatecznych testach usunąłem jednak tę funkcję, aby nie zabierała niepotrzebnie pamięci i czasu. Testy nie wykryły żadnych wycieków.

Wnioski:

Programowanie obiektowe potrafi naprawdę ułatwić i przyśpieszyć proces tworzenia gier czy baz danych. Niestety w moim programie nie ma za wiele polimorfizmu. Wydaje mi się on zbędny i tylko negatywnie by wpłynął na optymalizację rozwiązania.
Postanowiłem jednak zastosować go w warunku sprawdzania końca gry.
Stworzyłem wektor wskaźników na abstrakcyjną klasę rodzica.
Następnie dodałem do niego wskaźniki na dzieci.
Przy wywoływaniu wirtualnej funkcji sprawdzałem po prostu w pętli: *v[i] ->f();*
Przy większych projektach lub przy bazach danych takie rozwiązanie bardzo upraszcza i przyśpiesza proces twórczy.

Jako że był to mój pierwszy projekt związany z programowaniem obiektowym, natrafiłem na kilka problemów. Jednak po czasie uważam, że jest to naprawdę świetne rozwiązanie i pozwala na proste i szybkie rozwiązania wielu problemów.

My Project

Wygenerowano przez Doxygen 1.9.1

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

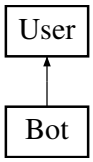
Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja klasy Bot

Diagram dziedziczenia dla Bot



Metody publiczne

- void where (int size, Bot &b)

  "Zycie" Komputera.
- void coords (Player &p)

  Funkcja ustawiajaca statek komputera.
- bool shot (Player &p)

  Funkcja obliczajaca wspolrzedne kolejnego strzalu.
- bool stop ()

  Funkcja strzelania, zwracajaca informacje o tym czy bylo trafienie czy pudlo.
- void dmg ()

  Funkcja sprawdzajaca warunek konca gry (zwyciestwo Gracza) - funkcja wirtualna, rozniaca sie od funkcji stop()Player wypisywana informacja.
- int getHp ()

  Funkcja uszkadzajaca statek Bota. (zmniejszajaca jego "zycie").
- void display (Player &p, Bot &b)

  Funkcja zwracajaca zycie Bota.
- void load1 ()

  Funkcja wypisujaca w konsoli plansze Bota z zamaskowanymi nietrafionymi statkami.
- void load2 (Player &p)

  Funkcja zapisujaca do pliku poczatkowe ustawienie statkow Komputera.

Dodatkowe Dziedziczone Składowe

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Bot.h
- C:/Users/Piotr/Desktop/PK/Ships/Bot.cpp

3.2 Dokumentacja klasy Bullet

Metody publiczne

- int getX ()

  Zmienna statyczna zliczajaca ilosc strzalow.
- int getY ()

  Funkcja zwracajaca wspolrzedna X danego strzalu.
- void allHit ()

  Funkcja zwracajaca wspolrzedna Y danego strzalu.
- int getHit ()

  Funkcja ustawiajaca wartosc zmiennej m_hit na "trafienie zatapiajace".
- Bullet (int x, int y, int hit)

  Funkcja zwracajaca zmienna m_hit.

Statyczne atrybuty publiczne

- static int ilosc = 0

  Zmienna informujaca o tym czy dany strzal byl trafieniem, pudlem czy strzalem zatapiajacym.

Przyjaciele

- ostream & operator<< (ostream &out, const Bullet &bullet)

  Konsturktor wieloargumentowy obiektu Bullet.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Bullet.h
- C:/Users/Piotr/Desktop/PK/Ships/Bullet.cpp

3.3 Dokumentacja klasy Leader

Metody publiczne

- void setNick (string s)

  Zmienna przechowujaca punktacje gracza w tabeli.
- void setScore (int a)

  Funkcja odpowiadajaca za ustawienie nicku.
- string getNick ()

  Funkcja odpowiadajaca za ustawienie wyniku.
- int getScore ()

  Funkcja zwracajaca nick gracza.

Przyjaciele

- ostream & operator<< (ostream &out, Leader &leader)

  Funkcja zwracajaca punktacje gracza.
- bool operator> (const Leader &l1, const Leader &l2)

  Przeciazanie operatora wypisu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Leader.h
- C:/Users/Piotr/Desktop/PK/Ships/Leader.cpp

3.4 Dokumentacja klasy Leaderboard

Metody publiczne

- void getLeaders ()

  Wektor wskaznikow na obiekty klasy lider, bedace najlepszymi graczami.
- void check (Player &p)

  Funkcja zwracajaca Tabele liderow z pliku.
- void show ()

  Funkcja sprawdzajaca czy gracz wpisuje sie do tabeli.
- void load ()

  Funkcja wypisujaca liderow.
- void sort ()

  Funkcja ladujaca tabele liderow do pliku tekstowego.
- void swap (Leader &l1, Leader &l2)

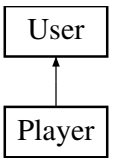
  Funkcja sortujaca Tabele Liderow (uzywa przeciazanego operatora porownania)

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Leaderboard.h
- C:/Users/Piotr/Desktop/PK/Ships/Leaderboard.cpp

3.5 Dokumentacja klasy Player

Diagram dziedziczenia dla Player



Metody publiczne

- void shot_coords ()

  Zmienna przechowujaca nick gracza.
- bool shot (Bot &b)

  Funkcja pytajaca i pobierajaca wspolrzedne kolejnego strzalu.
- void display (Bot &bot)

  Funkcja strzelajaca, zwraca wartosc zalezna od tego czy bylo trafienie czy nie.
- void where (int size, Bot &b)

  Funkcja wypisujaca aktualna plansze gracza.
- bool stop ()

  Funkcja ustawiajaca dany statek gracza.
- void dmg ()

  Funkcja wirutalna sprawdzajaca warunek konca gry. Informuje o porazce gracza.
- int getHp ()

  Funkcja uszkadzajaca statek gracza. Wyowylowana przy kazdym trafieniu przez Bota.
- void load2 (Bot &b)

  Funkcja zwracajaca zycie gracza.
- void load1 ()

  Funkcja zapisujaca do pliku koncowa plansze Gracza z naniesionymi strzalami komputera.
- int score (Bot &b)

  Funkcja zapisujaca do pliku poczatkowe ustawienie statkow gracza.
- void setNick ()

  Funkcja obliczajaca wynik gracza.
- int **getScore** ()
- string getNick ()

  Funkcja zwracajaca wynik gracza.

Dodatkowe Dziedziczone Składowe

3.5.1 Dokumentacja funkcji składowych

3.5.1.1 setNick()

```
void Player::setNick ( )  [inline]
```

Funkcja obliczajaca wynik gracza.

Funkcja ustawiajaca nick gracza.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Player.h
- C:/Users/Piotr/Desktop/PK/Ships/Player.cpp

3.6 Dokumentacja klasy Ship

Metody publiczne

- vector< shared_ptr< ShipPart > > getParts ()

  Wektor wskaznikow na obiekty klasy ShipPart bedacymi elementami statku np. Statek 4 masztowy "sklada sie" z 4 obiektow klasy ShipPart.
- Ship (int x, int y, int size, bool vertical)

  Funkcja zwracajaca wektor czesci statku.
- bool sink ()

  Konstruktor obiektu Klasy Ship.
- int getSize ()

  Funkcja sprwadzajaca czy statek zatonal.
- bool getOrientation ()

  Funkcja zwracajaca dlugosc statku.
- int getX ()

  Funkcja zwracajaca orientacje statku.
- int getY ()

  Funkcja zwracajaca poczatkowa wspolrzedna X statku.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/Ship.h
- C:/Users/Piotr/Desktop/PK/Ships/Ship.cpp

3.7 Dokumentacja klasy ShipPart

Metody publiczne

- ShipPart (int x, int y, bool alive)

  Informacja czy w dany element oddano juz strzal.
- int getX ()

  Konstruktor wieloargumentowy.
- int getY ()

  Funkcja zwracajaca wspolrzedna X danego elementu.
- void damage ()

  Funkcja zwracajaca wspolrzedna Y danego elementu.
- bool isAlive ()

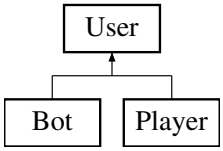
  Funkcja zmieniajaca wartosc m_alive danego elementu na false.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/ShipPart.h
- C:/Users/Piotr/Desktop/PK/Ships/ShipPart.cpp

3.8 Dokumentacja klasy User

Diagram dziedziczenia dla User



Metody publiczne

- void addBullet (int x, int y, int hit)

  Wektor wskaznikow na obiekty klasy Bullet (tylko oddane strzaly).
- void addShot (int x, int y, int hit)

  Funkcja dodajaca element do wektora m_bullets i wywolujaca konstruktor Bullet.
- vector< Bullet > getShots ()

  Funkcja dodajaca element do wektora m_shots i wywolujaca konstruktor Bullet.
- vector< shared_ptr< Bullet > > getBullets ()

  Funkcja zwracajaca wektor m_shots.
- void addShip (int x, int y, int size, bool horizontal)

  Funkcja zwracajaca wektor m_bullets.
- vector< shared_ptr< Ship > > getShips ()

  Funkcja umieszczajaca statek na planszy.
- void alive (int i)

  Funkcja zwracajaca wektor wskaznikow na obiekty klasy Ship (statki).
- bool collision (int x, int y, int size, bool orientation)

  Funkcja usuwajaca statek z wektora.
- virtual bool stop ()=0

  Funkcja walidacyjna, pilnujaca aby statki przy ich ustawieniu sie nie stykaly.
- void shots ()

  Metoda czysto wirtualna odpowiadajaca za sprawdzanie warunku konca gry.

Atrybuty chronione

- int **m_x**
- int m_y

  Wspolrzedna x strzalu.
- vector< shared_ptr< Ship > > m_ships

  Wsporzledna y strzalu.
- vector< shared_ptr< Bullet > > m_bullets

  Wektor wskaznikow na obiekty klasy statek (statki).
- vector< Bullet > m_shots

  Wektor wskaznikow na obiekty klasy Bullet (oddane strzaly + obrys).

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Piotr/Desktop/PK/Ships/User.h
- C:/Users/Piotr/Desktop/PK/Ships/User.cpp

Indeks