|  |
| --- |
| Prirodno-matematički fakultet u sarajevu |
| Kontrola leta iznad Kyrata |
| Kompjuterska geometrija |
|  |
| **Irhad Šarić** |
| **2/22/2018** |

**Kratki opis problema**

Zračni proctor je zadat kao ekstrudirani poligon. Dozvoljenu visinu leta određuje korisnik. Svake sekunde se generišu letovi. Postoje tri vrste letova:

* Eksterni letovi
* Interni letovi
* Poluinterni letovi

Eksterni letovi su letovi koji nadlijeću zračni prostor u granici od 10 kilometara. Interni letovi su oni letovi koji polijeću iz Kyrata do određene visine unutar zračnog prostora. Nakon dostignute određene visine, krstare tom visinom određen period nakon čega se spuštaju u Kyrat. Poluinterni letovi ulaze u zračni prostor na određenoj visini nakon čega slijeću u Kyrat ili izlaze iz zračnog prostora na njegovoj gornjoj granici. Do bliskog mimoilaženja dva aviona će doći ako se oni nalaze na udaljenosti manjoj od 5 kilometara. Potrebno je naći algoritam koji će naći letove kod kojih će doći do bliskog mimoilaženja i uraditi vizualizaciju.

**Implementacija**

**Klase:**

* Point
  + Event point
* Segment
* Polygon
* Flight

Klasa Point:

Atributi:

* X ( float )
* Y ( float )
* Z ( float )

Metode:

euclidean\_distance\_2D(self, b: 'Point') -> float

euclidean\_distance\_3D(self, b: 'Point') -> float

between2D(self, p1: 'Point', p2: 'Point') -> bool

@staticmethod  
orientation2D(a: 'Point', b: 'Point', c: 'Point') -> int

\_\_lt\_\_(self, other: 'Point') -> bool

\_\_sub\_\_(self, other: 'Point') -> 'Point'

Redom rade sljedeće:

* Vraća Euklidsku udaljenost dvije tačke u 2D prostoru
* Vraća Euklidsku udaljenost dvije tačke u 3D prostoru
* Ispituje da li se tačka nalazi između dvije tačke proslijeđene kao parametric
* Određuje orijentaciju tačke A u odnosu na segment BC
* Operator koji ispituje da li je tačka manja od tačke iz parametra
* Operator za oduzimanje tačaka

Klasa EventPoint(Point):

Atributi:

Pored atributa koje nasljeđuje:

* eventType (string) : vrsta tačke (start, end)
* flightIdentifier (Flight): kojem avionu pripada ova tačka

Klasa Segment:

Atributi:

* start ( Point ) : početak segmenta
* end ( Point ) : kraj segmenta

Metode:

getRandomPoint(self, lower: int, upper: int, flightType: str) -> 'Point'

@staticmethod  
def doIntersect(s\_1: 'Segment', s\_2: 'Segment') -> bool

Redom rade sljedeće:

* Vraća slučajnu tačku na segmentu
* Vraća da li se sjeku dva segmenta

Klasa Polygon:

Atributi:

* listOfEdges ( list ) : lista stranica poligona
* lowerLimit ( int ) : donja granica zračnog prostora
* upperLimit ( int ) : gornja granica zračnog prostora

Metode:

isInPoly(self, point:'Point') -> bool

draw(self, screen)

@staticmethods

getRandomPoint(poly: 'Polygon', lower: int) -> 'Point'

getRandomPointHigher(poly: 'Polygon', lower: int, upper: int) -> 'Point'

getRandomPointSameHeight(poly: 'Polygon', point: 'Point') -> 'Point'

Redom rade sljedeće:

* Provjerava da li je tačka proslijeđena kao parameter unutar poligona
* Iscrtava poligon na ekranu
* Vraća slučajnu tačku čija je z koordinata jednaka proslijeđenom parametru
* Vraća slučajnu tačku čija je z koordinata između vrijednosti parametara
* Vraća slučajnu tačku čija je z koordinata jednaka z koordinati tačke proslijeđene kao parameter

Klasa Flight:

Atributi:

* start ( Point ): početak putanje
* end ( Point ): kraj putanje
* velocity ( int ): brzina aviona
* currentPosition ( Point ): trenutna pozicija
* closeRange ( int ): radius u kojem dolazi do bliskog mimoilaženja
* flightType ( string ): tip leta (eksterni, interni, poluinterni)
* path ( list ): lista segmenata (eksterni ima samo jedan segment, poluinterni dva, interni tri)
* outsidePoly ( bool ): da li se nalazi unutar poligona

Metode:

calculateCurrentPosition(self)

@staticmethods

intersects(f1: 'Flight', f2: 'Flight') -> bool

generateFlight(poly: 'Polygon', lower: int, upper: int, flightType: str, closeRange: int) -> 'Flight'

draw(flightList, pygame, screen, intersections, myfont)

Redom rade sljedeće:

* Računa trenutnu poziciju aviona
* Provjerava da li se avioni blisko mimoilaze
* Generiše let proslijeđenog tipa, granice bliskog mimoilaženja i brzine u proslijeđenim granicama
* Iscrtava avion na ekranu

**Opis algoritma**

getIntersections(flightsList: list) -> set

Funkcija koja vraća set letova koji se blisko mimoilaze radi na sljedeći način. Glavna ideja je da se oko aviona napravi kružnica poluprečnika closeRange sa centrom u currentPosition i provjerava da li dolazi do presjeka kružnica. U slučaju da dođe do presjeka kružnica, dobijamo potencijalne kandidate za avione koji će se blisko mimoići. U tom slučaju ispitujemo Euklidsku udaljenost dva aviona. U slučaju da je ta udaljenost manja od granice za blisko mimoilaženje, te avione dodajemo u set. Detaljnije:

Na početku instanciramo priority queue (PQ), deque i set. Prolazimo listom flightList i u PQ dodajemo najljevlju i najdesniju tačku kružnice, ali na način da ona bude tipa Event Point, iz razloga da kasnije možemo skontati da li je ona početna ili krajnja tačka kružnice. Također da možemo identificirati kojem avionu pripada ta tačka. Iz PQ će se ugrađenom funkcijom pop() uzimati prvo one tačke kojima je x koordinata najmanja, zbog načina na koji je definisan operator u klasi Point. Sve dok PQ nije prazan uzimamo tačku i ispitujemo da li je ona početna ili krajnja tačka kružnice. U slučaju da je početna, odredimo kojem letu ta tačka pripada i taj let provjeravamo sa svakim letom u deku. Ako se mimoilaze dodajemo ih u set. Dodajemo taj avion u deque na početak. U slučaju da je tačka na koju smo naišli krajnja, tada iz deka brišemo zadnji avion, zbog toga što je prva krajnja tačka na koju naiđemo sigurno tačka aviona kojem je najljevlja tačka ona tačka na koju smo prvu naišli. Ovo vrijedi samo u slučaju kada su prečnici kružnica jednaki što je kod nas slučaj.

**Detaljnije o metodama**

Opisat ću metode koje nismo koristili na vježbama

getRandomPoint(self, lower: int, upper: int, flightType: str) -> 'Point'

Ova metoda klase Segment radi na način da se izračuna jednačina prave kroz dvije tačke. Nakon toga se bira nasumična x koordinata u reonu između početne x koordinate segmenta i krajnje x koordinate segmenta. Nakon toga se provjerava da li je prava okomita na x osu. U slučaju da jeste tada se y koordinata bira na način da se izabere nasumičan broj između y koordinata segmenta. Ako nije okomita na x osu tada se y koordinata računa prema prethodno definisanoj formuli. Z koordinata tačke se bira u odnosu na vrstu leta. Ako je let eksternog tipa tada se z koordinata bira u reonu između upper i upper+100, inače se bira iz reona od lower do upper.

def isInPoly(self, point:'Point') -> bool

Ova metoda klase Polygon radi na način da se nađe tačka koja ne pripada datom poligonu tako što se nađe najveća y koordinata i najmanja x koordinata. Da bismo dobili tačku koja je sigurno van poligona dovoljno je da x koordinatu umanjimo za 1. Nakon toga se pravi segment čije su krajnje tačke tačka koja se nalazi van poligona i tačka proslijeđena kao parametar. Provjerava se broj presjeka sa poligonom, ako je on paran tada se tačka nalazi van poligona, inače unutra.

def getRandomPoint(poly: 'Polygon', lower: int) -> 'Point'

Ova metoda klase Polygon radi na način da se napravi pravougaonik koji sadrži dati poligon, nakon čega se generiše slučajna tačka iz tog pravougaonika i provjerava se da li ona leži unutar poligona. Ako ne leži na isti se način generiše nova tačka sve dok se ne dobije tačka koja je unutar poligona. Data tačka će imati z koordinatu jednaku proslijeđenom parametru.

def getRandomPointHigher(poly: 'Polygon', lower: int, upper: int) -> 'Point'

Ova metoda klase Polygon radi na način da se generiše slučajna tačka čija će z koordinata biti u granicama između lower i upper.

getRandomPointSameHeight(poly: 'Polygon', point: 'Point') -> 'Point'

Ova metoda klase Polygon radi na način da se generiše slučajna tačka čija će z koordinata imati istu vrijednost kao z koordinata tačke proslijeđene kao parametar.

calculateCurrentPosition(self)

Ova metoda klase Flight radi na način da se izračuna dužina puta koju let mora preći. Nakon toga se računa vrijeme koje je potrebno da se taj put pređe. To se zna uraditi jer je zadana brzina aviona. Nakon toga put se projecira na x osu i računa se brzina promjene x koordinate. Isto se radi i za z koordinatu, dok se y koordinata računa po formuli prave kroz dvije tačke.

intersects(f1: 'Flight', f2: 'Flight') -> bool

Ova metoda klase Flight radi na način da se oko svakog aviona napravi kružnica sa centrum u trenutnoj poziciji aviona čiji je poluprečnik granica za mimoilaženje aviona. Kružnice će se sjeći u slučaju da je zbir njihovih poluprečnika veći ili jednak od Euklidske udaljenosti u 2D. Ovo je potreban, ali ne i dovoljan uslov da se letovi blisko mimoiđu. Ukoliko je ovaj uslov zadovoljen ispituje se Euklidska udaljenost aviona u 3D prostoru i ispituje se da li se zamišljene lopte oko aviona sjeku. One će se sjeći ako im je zbir poluprečnika veći od Euklidske udaljenosti u 3D prostoru.

def generateFlight(poly: 'Polygon', lower: int, upper: int, flightType: str, closeRange: int) -> 'Flight'

Ova metoda klase Flight radi na način da se prvo provjeri tip leta i u odnosu na taj podatak pravi putanja leta. U slučaju da je let eksternog tipa, pomoću prethodno opisanih funkcija će se generisati dvije tačke koje leže na ivicama poligona, zadat će mu se brzina u proslijeđenim granicama i atribut putanje će postati lista sa samo jednim segmentom.

U slučaju da je let internog tipa tada će se generisati tačka ulaska u zračni prostor i tačka unutar poligona na određenoj visini i taj segment će predstavljati jedan od tri segmenta koji će biti u putanji. Drugi segment će biti s početkom u krajnjoj tački prvog segmenta i random tački iste visine koja se nalazi unutar poligona, dok će treći segment biti s početkom u kranjoj tački prethodnog segmenta i random tački na donjoj strain zračnog prostora.

U slučaju da je let poluinternog tipa, tada će se generisati tačka na nekoj od ivica poligona čija će z koordinata biti između donje i gornje granice zračnog prostora i random tačka unutar zračnog prostora. To će biti prvi segment putanje, dok će drugi segment biti segment s početkom u prethodnom kraju segmenta i slučajnoj tački na gornjoj ili donjoj granici zračnog prostora.

draw(flightList, pygame, screen, intersections, myfont)

Ova metoda klase Flight iscrtava letove na ekranu određenom bojom. Eksterni crnom, interni ljubičastom, poluinterni smeđom. Ova funkcija također ispituje da li je let došao do svoje krajnje destinacije, ili pak do krajnje tačke trenutnog segmenta na kojem leti. U slučaju da je završio kompletan let, briše se iz liste letova. Ako je završio s jednom putanjom, ispituje se da li postoji iduća putanja. Ako postoji mijenja se trenutni početak i kraj letenja aviona, a ako ne postoji briše se iz liste trenutnih letova. Ukoliko se avionic blisko mimoilaze, bit će iscrtani crvenom bojom. Također ova funkcija ispisuje trenutne pozicije aviona zaokružene na cijeli broj.