Detectarea coliziunilor

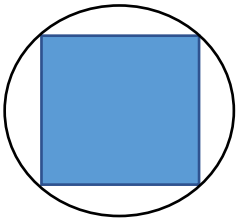
Detectarea coliziunilor este o parte esentiala din tot ceea ce inseamna jocurile moderne. In cadrul acestui proiect, se propune implementarea unui mecanism de detectare a coliziunilor dintre diverse forme geometrice.

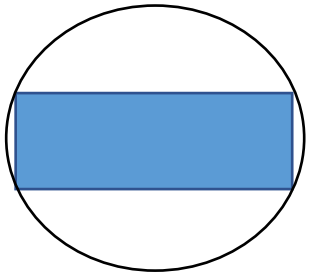
Descriere

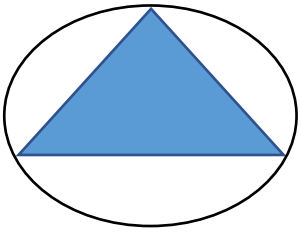
In cadrul acestui proiect vom lucra cu mai multe tipuri de forme geometrice.

- Cerc
- Patrat
- Dreptunghi
- Triunghi

Fiecare din figurile mentionate va fi incadrata de un cerc(cu exceptia cercului), care ne va ajuta in detectarea coliziunilor.

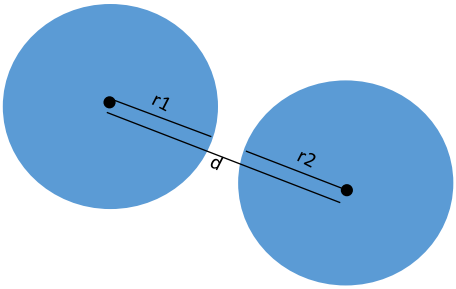






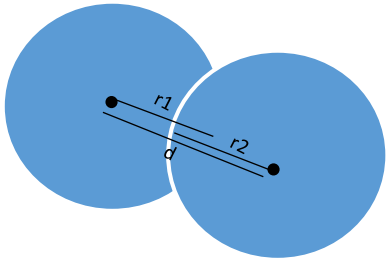
Doua cercuri se afla in coliziune daca distanta dintre centrelor lor este mai mica sau egala decat suma razelor celor doua cercuri.

Lipsa coliziune



Se observa cum, in situatia in care nu avem coliziune, d > r1 + r2.

Coliziune



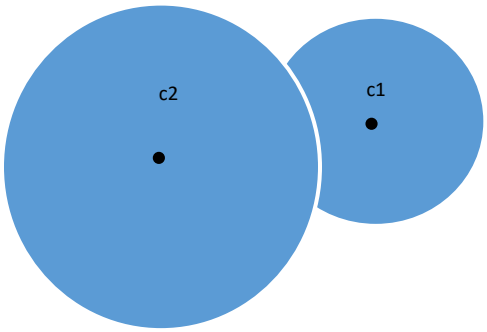
Se observa cum, in situatia in care avem coliziune, d <= r1 + r2.

Pentru usurinta, vom avea cateva reguli in ceea ce priveste determinarea razei cercului incadrator:

- Pentru patrat, va fi jumatate din lungimea laturii. raza = latura / 2
- Pentru dreptunghi, va fi jumatate din max(lungime, latime). raza = max(lungime, latime) / 2
- Pentru triunghi, va fi jumatate din max(latura1, latura2, latura3) raza = max(latura1, latura2, latura3) / 2

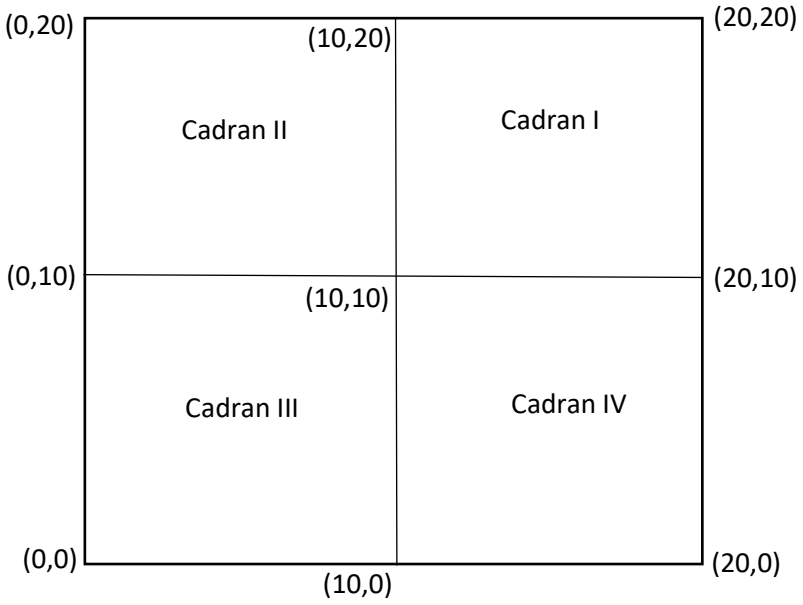
Centru figurii va fi dat la initializarea formei.

Vom introduce notiunea de **semi-incluziune**. Un cerc c1 este semi-inclus de cercul c2 daca c1 este in coliziune cu cercul c2 si raza lui c1 este strict mai mica decat raza lui c2. Daca razele celor doua figuri sunt egale, nu vom spune nici ca c1 semi-include c2, nici invers.



In exemplul de mai sus putem spune ca c1 este inclus in c2.

O alta entitate cu care vom lucra este **cadranul.** La initializarea programului, veti primi un spatiu de lucru in care pot fi asezate figurile geometrice. Orice figura geometrica cu centrul in afara acestui spatiu, va fi ignorata. Spatiul de lucru dat, va fi impartit in patru cadrane cu **dimensiuni egale**. In exemplu de mai jos veti vedea un spatiu de lucru impartit in cele patru cadrane.



La initializare, pentru spatiu de lucru, vor fi date doar coordonatele punctului din stanga-jos si din dreapta-sus. Pentru spatiul de lucru (0, 0) - (20, 20) se vor crea cele patru cadrane care vor fi marginite de urmatoarele puncte:

- Cadran I: (10, 10) - (20, 10) - (20, 20) - (10, 20)
- Cadran II: (0, 10) - (10, 10) - (10, 20) - (0, 20)
- Cadran III: (0, 0) - (10, 0) - (10, 10) - (0, 10)
- Cadran IV: (10, 0) - (20, 0) - (20, 10) - (10, 10)

La ANEXE vom gasi atat un exemplu de impartire in cadrane cat si un exemplu de conditii care pot determina daca un punct apartine sau nu unui cadran.

Cerinte:

Veti avea de implementat o clasa parinte pentru formele geometrice care va avea cel putin o metoda abstracta. Veti crea cate o clasa pentru fiecare forma specifica, care va trebui sa mosteneasca acesta clasa generica. In ceea ce priveste o clasa cu o forma specifica, trebuie sa aveti metode pentru:

- calcul raza cerc incadrator in functie de figura specifica(conform regulilor specificate)
- detectie coliziuni cu alte obiecte
- detectie semi-incluziune(este important sa stim daca semi-includem vreo figura deoarece asta va fi o informatie pe care o vom scrie in fisierul de output la output)

Pentru fiecare figura se vor specifica coordonatele centrului.

Va trebui sa creati o clasa spatiu de lucru care la initializare primeste coordonatele spatiului de lucru si isi creeaza patru atribute instanta care reprezinta cadranele(este la alegerea voastra daca tineti cadranele sub forma de obiecte sau retineti doar coordonatele pentru fiecare). Aceasta clasa va trebui sa contina o metoda care sa verifice pentru o forma specifica daca se afla sau nu intr-un cadran. Apartenenta la un anumit cadran se face pe baza coordonatelor centrului figurii.

Impartirea in cadrane se va face in ordinea precizata in figura. Cadranul I va fi cel din dreapta-sus, cadranul II va fi in stanga-sus, cadranul III va fi in stanga-jos iar cadranul IV in dreapta-jos.

Format fisier intrare:

Pe prima linie se vor specifica coordonatele spatiului de lucru(stanja-jos si dreapta-sus), separate prin spatiu(x1 y1 x2 y2). Pe urmatoarele linii vor fi introduse figuri sub forma:

- Pentru cerc, se va specifica id-ul, coordonatele centrului(x, y) si raza, separate prin spatiu.
- Pentru patrat, se va specifica id-ul, coordonatele centrului(x, y) si latura, separate prin spatiu
- Pentru dreptunghi, se va specifica id-ul, coordonatele centrului(x, y), latimea si lungimea, separate prin spatiu.
- Pentru triunghi, se va specifica id-ul, coordonatele centrului(x, y), si cele trei laturi, separate prin spatiu.

Exemplu fisier input:

0 0 20 20

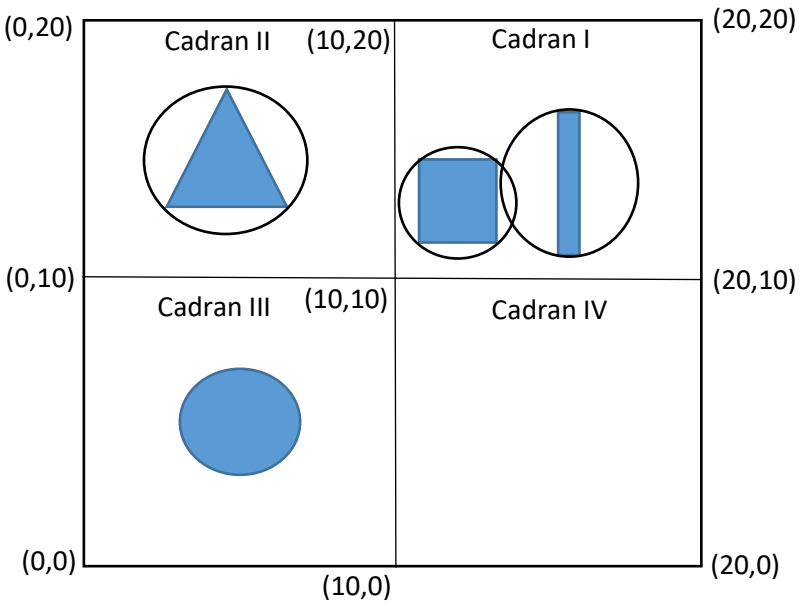
cerc c0 5 5 2

triunghi t0 5 15 2 2 2

patrat p0 12 13 4

dreptunghi d0 16 14 1 6

Vizualizare



Va trebui sa scrieti un fisier de output pentru fiecare fisier de input. Detaliile fiecarei figuri, in fisierul de output, vor fi scrise pe cate o linie separata. Pentru fiecare figura se va specifica:

- id-ul formei, id-ul/id-urile formelor cu care intra in coliziune, id-ul/id-urile formelor pe care le semi-include, cadranul in care se afla.

Exemplu fisier output

c0 | | | 3

t0 | | | 2

p0 | d0 | | 1

d0 | p0 | p0 | 1

Observatii:

- in fisierul de output, am despartit informatiile folosind 'pipe'('|')
- inainte si dupa pipe avem spatiu
- putem avea mai multe forme cu care suntem in coliziune sau pe care le semi-includem si afisarea se va face in felul urmator:
  ex. c12 | p7 t3 | t3 p7 | 2
- daca o figura nu apartine niciunui cadran, nu va aparea in fisierul de output
- **tratarea figurilor in fisierul de output se va face in ordinea data de fisierul de input. Acelasi lucru se va intampla si daca avem coliziune cu mai multe obiecte, sau semi-includem mai multe obiecte. Ordinea id-urilor va fi cea din fisierul de input.**

Observatii generale:

- Toate coordonatele sunt date in format (x, y).
- Pentru calculul distantei dintre doua punte puteti folosi functia de la ANEXE.
- NU COPIATI. COPIATUL VA DUCE LA ANULAREA PUNCTAJULUI.

Punctaj

70 % teste automate(10 teste, creste dificultatea cu fiecare test, 7p/test).

30 % coding style & originalitate.

Teste

In directorul input se gasesc 10 teste. Rezolvarea fiecarui test se gaseste in directoul rezultate. Pentru fiecare fisier de input, va trebui sa obtineti cate un fisier output, in care sa se gaseasca informatiile cerute. Sugestia este sa creati un nou director, numit output si pentru fiecare fisier test din directorul input sa aveti un fisier numit identic in directorul output.

Rulare teste automate

Va punem la dispozitie doua solutii de verificare:

- Testare fisier individual. Pentru fiecare fisier de input veti avea un fisier de output. Ideea de baza este verificare diferentelor dintre fisierul obtinut de voi si fisierul pe care ar trebui sa il obtineti. Pentru a valida ca ceea ce ati obtinut este corect veti rula urmatoarea comanda in cmd sau in terminalul de PyCharm. Inainte de rularea comenzii trebuie sa va asigurati ca sunteti in acelasi director cu fisierul checker.py.
  python checker.py --obtinut cale_fisier_obtinut --real cale_fisier_real

  cale_fisier_obtinut este calea catre fisierul obtinut de voi

  cale_fisier_real este calea catre fisierul corespunzator testului din directorul rezultate

- Al doilea mod, ruleaza toate testele in acelasi timp. Pentru a putea fi posibil acest lucru, numele fisierului de output, trebuie sa fie acelasi cu numele fisierului de input, de aici si sugestia de a crea un director separat in care sa le puneti. Daca numele fisierelor nu este acelasi, rularea nu va fi posibila. Inainte de rularea comenzii trebuie sa va asigurati ca sunteti in acelasi director cu fisierul checker.py.
  python checker.py --multiplu --obtinut cale_director_obtinut --real cale_director_real

  cale_director_obtinut este calea catre directorul care contine rezolvarile testelor obtinute de voi.
  cale_director_real este calea catre directorul care contine rezultatele testelor reale, adica directorul rezultate.

ANEXE

Anexa 1

Pentru determinarea distantei dintre doua puncte P1 (x1, y1) si P2(x2, y2) puteti folosi urmatoarea metoda:

```
import math

def calculeaza_distanta(x1, y1, x2, y2):
        return math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)
```

Anexa 2

Avand coordonatele unui spatiu de lucru punctul stanga-jos (x1, y1) si punctul dreapta-sus(x2, y2). Coordonatele cadranelor(punctele sanga-sus si dreapta-jos) vor fi urmatoarele:

cadran1 = ((x1 + x2) / 2, (y1 + y2) / 2) (x2 , y2)
cadran2 = (x1, (y1 + y2) / 2) ((x1 + x2) / 2, y2)
cadran3 = ((x1, y1) ((x1 + x2) / 2, (y1 + y2) / 2)
cadran4 = ((x1 + x2) / 2, y1) (x2, (y1 + y2) / 2)

Anexa 3

Fiind date coordonatele unui spatiu de lucru/cadran (x1, y1, x2, y2) un punct(x, y) apartine acelui spatiu/cadran daca toate conditiile de mai jos sunt simultan indeplinite:
x > x1 and x < x2 and y > y1 and y < y2
Se observa ca se folosesc comparatori stricti(<, > nu <= sau >=). Daca exact conditia de sus nu este indeplinita, punctul nu va apartine acelui spatiu/cadra. In teste NU veti avea cazuri in care punctele sunt chiar pe axa ce desparte doua cadrane sau pe limita exterioara a spatiului de lucru.