# Házi feladat Terv

Programozás alapjai 2.

## Feladat leírás

A program egy konzolos applikáció, amely képes bármilyen kétdimenziós konvex síkidom ábrázolására. Ezeket grafikus úton is képes megjeleníteni. Fő funkciója ezen síkidomok érintkezésének vizsgálata. A felhasználó parancssoros utasításokkal hozhat létre síkidomokat megadva azok típusát és paramétereit, és egyedi névvel különbözteti meg ezeket. A létrehozott síkidomokat mozgathatja, elforgathatja, és nagyíthatja. A program lehetőséget ad a létrehozott síkidomok érintkezésének vizsgálatára a GJK algoritmus segítségével. A program használható önmagában is, de könyvtár verzióban is elérhető, így a felhasználó a parancsokat és a síkidomtípusokat bővítheti.

## Síkidomok kezelése

A program támogatni fog alapvető síkidom típusokat. A könyvtár verzióban ezeket a felhasználó saját típusaival bővítheti úgy, hogy azokkal a parancsok és az érintkezés vizsgálata automatikusan működjenek.

A felhasználó a megadott típusokból választva parancssoros utasítással példányosíthat síkidomokat, amelyeket egyedi névvel lát el, és paraméterez az adott síkidom típusának megfelelően. Nem megfelelő paraméterezés esetén a program hibaüzenetet ír a standard kimenetre és a példányosítás sikertelen lesz. Ha a felhasználó egy síkidomnak egy, már létező síkidommal azonos nevet ad, a példányosítás sikertelen lesz. Sikeres példányosítás esetén a síkidomra a felhasználó a továbbiakban a neve alapján hivatkozhat.

Az alapból támogatott síkidomok és paramétereik:

* Pont - pont x, y koordinátája
* Kör – sugár hossza
* Ellipszis – nagytengely felének hossza, kistengely felének hossza
* Sokszög – csúcsok száma, csúcsonként x, y koordináta
* Szabályos sokszög – csúcsok száma, egy oldalának hossza
* Margós sokszög – csúcsok száma, margó mérete, csúcsonként x, y koordináta
* Bézier-görbe – szegmensek száma, szegmensenként bal, közép és jobb pont x, y komponense

A program az alaptípusokra ellenőrzi a konvexitást, és hibaüzenetet ír a standard kimenetre, ha a megadott síkidom konkáv (például sokszög esetében). A felhasználó által készített típusok esetében a konvexitás ellenőrzése a felhasználó feladata.

A program a síkidom példányokat képes *.shps* formátumú fájlba menteni és beolvasni azokat. Az ilyen fájlok felépítése a következő:

Egy síkidom adatainak kezdetét a *new* token jelzi, majd a síkidom típusa következik és a síkidom paraméterei. Ez a paraméterlista nem egyezik a parancssoros *create* parancs esetén megadandó paraméterlistával.

A formátum nem követeli meg hogy a síkidomok különböző sorokban szerepeljenek, de a program esztétikai okokból ezeket automatikusan sorokba rendezi.

## Parancsok

A program képes a standard bemenetről szöveges utasításokat fogadni. A könyvtár verzióban ezeket a felhasználó saját parancsaival bővítheti.

A program ezekkel a parancsokkal rendelkezik:

* help – Kilistázza a végrehajtható parancsokat, és leírásaikat.
* shapetype – Kilistázza a példányosítható síkidom típusokat.
* shapes – Kilistázza a létrehozott síkidomokat és azok típusát.
* create – Létrehoz egy síkidomot a megadott névvel és paraméterekkel.
* destroy – Törli a megnevezett síkidomot.
* move – Elmozgatja a megnevezett síkidomot egy megadott vektorral.
* rotate – Elforgatja a megnevezett síkidomot egy megadott szöggel.
* scale – Tengelyenként nagyítja a megnevezett síkidomot megadott x, y komponenssel
* contacts – Kilistázza az érintkező síkidomok nevét.
* contact – Megvizsgálja, hogy két megnevezett síkidom érintkezik e.
* save - A létrehozott síkidomokat elmenti a betöltött fájlba.
* saveas – A létrehozott síkidomokat egy új fájlba menti. A nem fogad el már létező fájlt.
* load – Beolvassa a megadott fájlban tárolt síkidomokat.
* merge – Beolvassa a megadott fájlban tárolt síkidomokat, és a már meglévőkhöz adja.
* openwin – Új ablakot nyit, ahol a síkidomok grafikusan ábrázolva tekinthetők meg.
* exit – Kilép a programból.

Egy parancsot meghívni a helyes kulcsszóval és helyes paraméterezéssel lehet. Ha a megadott kulcsszó nem felismerhető, vagy a felhasználó nem adott meg elég paramétert, a program hibaüzenetet ír a standard kimenetre, és a végrehajtani kívánt parancsnak nem lesz hatása. Ha túl sok paramétert ad a felhasználó az nem befolyásolja a parancs végrehajtását. A nem felhasznált paraméterekről a felhasználó szöveges értesítést kap a standard kimeneten.

## Terv

## GJK Algoritmus

A GJK algoritmussal eldönthető, hogy két konvex síkidom érintkezik e. Az algoritmus szükséglete, hogy legyen olyan függvény minden síkidomra, ami megadja egy bizonyos irányban a síkidom legmesszebbi pontját. Ezt a függvényt support függvénynek hívák.

Az alább leírt függvények a GJKSolver tagfüggvényei, melynek tagváltozói:

* simplex: két elemű vec2d tömb
* s1, s2: a két tesztelt síkidom
* is\_contact: bool

Az algoritmus a checkOverlap() eljárás meghívásával futtatható, és a végeredmény az is\_contact változóból olvasható ki:

* true: s1 és s2 síkidomok érintkeznek
* false: s1 és s2 síkidomok nem érintkeznek

függvény getSupportpoint(direction: vec2d, point: vec2d):  
 point = s1.support(direction) – s2.support(direction)  
 visszatér point \* direction > 0

függvény findFirstSimplex:  
 direction := s1.getPosition() – s2.getPosition()  
 Ha getSupportPoint(direction, simplex[0]):  
 visszatér hamissal  
 direction = -simplex[0]  
 Ha getSupportPoint(direction, simplex[1]:  
 visszatér hamissal  
 visszatér igazzal

függvény findNextPoint(point: vec2d):  
 edge := simplex[0] – simplex[1]  
 to\_origo := -simplex[1]  
 normal := getNormal(edge, to\_origo)  
 visszatér getSupportPoint(normal, point)

eljárás checkOverlap():  
 Ha nem findFirstSimplex():  
 visszatér  
 next\_point: vec2d  
 foundRtiangle = hamis  
 foundValidPoint = igaz  
 ciklus:  
 foundValidPoint = findNextPoint(next\_point)  
 Ha foundValidPoint:  
 normal1 := getNormal(simplex[0]–next\_point, next\_point–simplex[1])  
 normal2 := getNormal(simplex[1]–next\_point, next\_poin –simplex[0])  
 np\_to\_origin = -next\_point  
 Ha normal1 \* np\_to\_origin > 0:  
 simplex[1] = next\_point  
 Különben ha normal2 \*np\_to\_origin > 0:  
 simplex[0] = next\_point  
 Különben:  
 foundTriangle = igaz  
 Amíg foundValidPoint és nem foundTriangle  
 is\_contact = foundTriangle