Szegedi Tudományegyetem
Informatikai Tanszékcsoport

GeoTransz**ND**

Szabad Transzformációk az N dimenziós Euklideszi térben

Szakdolgozat

Konzulens: **Dr. Kató Zoltán**
Egyetemi docens
Képfeldolgozás és Számítógépes Grafika Tanszék

Készítette:

Bánáti-Baumann Zsolt

Programozó-matematika szak

Témavezető: **Dr. Kurusa Árpád**
Egyetemi docens
Geometria Tanszék

Szeged
2007

Feladatkiírás

Egyszerű geometriai alakzatok felvételére és mozgatására, transzformációk tanulmányozására, illetve az N dimenziós euklideszi tér megjelenítésére alkalmas program készítése.

Tartalmi összefoglaló

- ***A téma megnevezése:***

  N dimenziós euklideszi geometria, szerkesztőprogram.

- ***A megadott feladat megfogalmazása:***

  Egyszerű geometriai alakzatok felvételére és mozgatására, transzformációk tanulmányozására, illetve az N dimenziós euklideszi tér megjelenítésére alkalmas program, kezelőfelület kialakítása.

- ***A megoldási mód:***

  Windows alkalmazás készítése MS Visual C++ környezetben. Az N dimenzió kezelésére 2 dimenziós nézeteken alapuló rendszer, a tér további érzékeltetésére 3 és n dimenziós (vetített) nézetek. Az alakzatfelvételhez szkriptnyelv kialakítása.

- ***Alkalmazott eszközök, módszerek:***

  Az alkalmazás C++, a Windowsos felhasználói felület Visual C++ nyelven, MFC felhasználásával készült. A szkriptnyelv LUA alapú.

- ***Kulcsszavak:***

  n dimenziós geometria, geometria-program, transzformációk.

Tartalomjegyzék

1 Bevezetés

Ez a munka a 2003-ban írt, GeoTransz nevű, 2D geometriai oktatóprogram kiterjesztése több dimenzióba. Több dimenziós tér megjelenítése és alakzatainak felvétele, kezelése alapvetően más megközelítést igényelt, így a két munka közötti rokonság kimerül a téma hasonlóságában.

Egyedül dokumentáció geometriai háttér c. fejezete építkezik az előző munkámra, itt ugyanis feleslegesnek tartottam ugyanazon fogalmak és definíciók megismétlését, újra-bevezetését. Ehelyett csak rámutattam, hogyan lehet bevezetni több dimenziót, és több dimenziós térre kiterjeszteni az ott tárgyaltakat.

2 Célkitűzések

E munka célkitűzése egy N dimenziós tér és alakzatainak, transzformációinak kezelésére alkalmas geometriai program megvalósítása volt.

Ez persze egy lényegétől megfosztott célkitűzés; olyan, ami méretét tekintve megvalósítható egy diplomamunka keretei között. Az igazi motiváció annak vizsgálata volt, hogyan lehet olyan programot készíteni, mely továbbfejleszthető oktatási, valamint kutatási célokra is.

Hiszek abban, hogy a geometria szemléletessége hasznos a matematikai gondolkodás fejlesztésében, hogy a megtapasztalás szükséges elvontabb témakörök mélyebb megértéséhez is. Az N dimenziós euklideszi geometria témakörében tudomásom szerint eddig nem született ilyen motivációjú oktató/szerkesztő program. Ennek a térnek a betöltésére tettem kísérletet ezzel a diplomamunkával.

Transzformációk alatt gondoltam egyrészt a speciális egybevágósági, hasonlósági transzformációkra, valamint nem ilyen keretek közé szorított, hanem „szabad transzformációkra”. Tetszőleges geometriai transzformációra: az előző diplomamunkámban tárgyalt „elrontott” transzformációkra, vagy épp függvényekkel megadható koordináta-transzformációra.

3 Összehasonlító elemzés

Ez a fejezet rámutat a főbb különbségekre és hasonlóságokra korábbi (GeoTransz) és jelenlegi (GeoTranszND) munkám között.

A sík és az N dimenziós tér megjelenítése és alakzatainak felvétele, kezelése gyökeresen eltérő probléma, így az ebből eredő különbségek is élesek a korábbi munkám és jelen szakdolgozat között.

A síkban egyetlen nézet biztosította a **kezelőfelületet**. A több dimenzió alakzatfelvételéhez több 2 dimenziós nézetet vezettem be. Ezek segítségével lehet bármelyik dimenzió mentén mozgatni a felvett alakzatokat. A tér további érzékeltetésére szolgálnak a vetített, 3 és N dimenziós nézetek. A több dimenziós alakzat(transzformáció)felvételre továbbá egy szkriptnyelv szolgál.

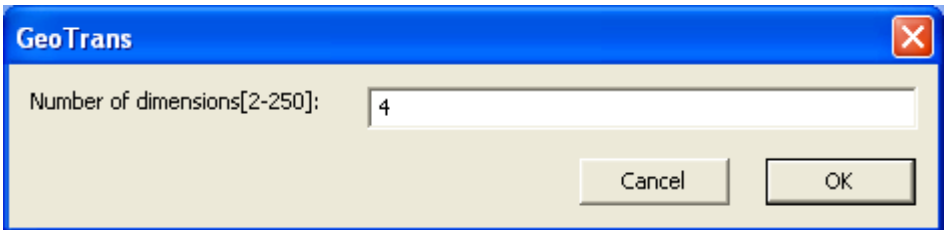
Megmaradt a **pont alapú szemlélet**. Ponthalmazokat hozunk létre, transzformáljuk, és ponthalmazokat kapunk. Ez többek között azt jelenti, hogy a program a folytonosságból adódó problémát, miszerint pl. egy szakasz végtelen pontból áll, úgy hidalja át, hogy adott pontossággal véges pontok halmazára bontja a folytonos alakzatokat, és ezekkel a pontokkal dolgozik. Erre azért van szükség, mert mindenképpen meg akartam tartani a transzformációk terén a teljes flexibilitást.

Rugalmasabbá akartam tenni a **transzformációk értékkészletét**. A 2D verzióban egy transzformáció a sík összes alakzatát transzformálta a könnyű kezelhetőség kedvéért. Több dimenziós térben már az alakzatfelvételhez is esetlegesen sok transzformációra van szükség, így elengedhetetlen, hogy egy meghatározott ponthalmazhoz kötődjön (ez lehet alakzatok halmaza is).

4 Használati útmutató

4.1 A program indítása, dimenzióválasztás

Induláskor a program bekéri a munkaterület dimenziószámát, a megadott dimenziójú euklideszi térben tudunk dolgozni.



4.2 Nézetek létrehozása

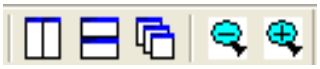
1. ábra: Új nézetek létrehozása

A tér megjelenítése nézeteken keresztül történik. Új nézeteket az 1 ábrán látható eszköztáron vehetünk fel.

Az első ikon létrehozza az N dimenziós munkaterülethez az összes 2D, és egy N dimenziós nézetet. Az ablakok címsorában a megjelenített tengelyek sorszámát látjuk. Pl. ha 4 dimenzióban dolgozunk, a (0,1), (1,2), (2,3) feliratú ablakok jelennek meg, melyek rendre az XY, YZ,ZW sík menten mutatja a munkaterületet. A negyedik ablak a 4 dimenziós nézet, ahol látható a mind a négy tengely.

A második ikonnal 2D nézeteket vehetünk fel egyesével, a harmadik ikonnal 3D nézeteket vehetünk fel. Ehhez meg kell adni a megjelenítendő tengelyek sorszámát

4.3 Nézetek kezelése



2. ábra: Nézetek kezelése

A megnyitott nézeteket elrendezésében segít a nézetek eszköztár első három ikonja. Az első függőlegesen, a második vízszintesen rendezi az ablakokat, úgy hogy kitöltsék az összes rendelkezésre álló helzet. A harmadik kupacba rendez.

Az aktuális nézetben a megjelenítés méretét (zoom) az egér görgőjével változtathatjuk, vagy a kicsinyítés / nagyítás ikonnal.

3. ábra: Nézet mozgatása ikon

Egy nézet tartalmának görgetésére használhatjuk a görgetősávokat, vagy a nézet mozgatása gombot kiválasztása után az egérgomb folyamatos lenyomásával mozgathatjuk a nézeten megjelenő munkaterületet(„szkrollozás”).

4.3.1 ND nézet mozgatása

ND nézet esetén ezzel az ikonnal tudjuk a megjelenítés egységvektorait változtatni, ha a CTRL-t és a bal egérgombot lenyomva tartjuk.

4.4 Alakzatok létrehozása, kezelése a felhasználói felületen

Az N dimenziós térben a helymeghatározás nehézkesebb felhasználói felületen, mint programkóddal, továbbá bonyolultabb alakzatok, ponthalmazok meghatározása is akadályokba ütközik; ezért a felület csak alapalakzatok felvételét támogatja.

Alakzatok létrehozására és kezelésére a 2D nézetekben van mód, hiszen ezekben egyértelmű a koordináták megadása. Egy 2D nézet két koordinátája egyértelműen meghatároz egy síkot, a (0,1), (1,2) ...(n-2,n-1) síkokban pedig egyértelműen meghatározható az N dimenziós tér egy pontja. Az N dimenziós mozgatás így több lépésben történik. Egy lépésben a meg nem jelenített koordináták alakzat felvételekor 0 értéket kapnak, mozgatáskor pedig nem

változnak.

4. ábra: Alakzatok létrehozása, kezelése eszköztár

Az alakzatok létrehozása eszköztáron lenyomott ikon határozza meg, hogyan viselkedik az egér bal gombja

Pont felvétele: 4. ikon.

Szakasz felvétele: 5. ikon.

Töröttvonal felvétele: 6. ikon.

Egyenes felvétele: 7. ikon. Nincs megvalósítva.

Félegyenes felvétele: 8. ikon. Nincs megvalósítva.

Síkidom felvétele: 9. ikon. Nincs megvalósítva.

Kör felvétele: 10. ikon.

Szabadkézi vonal felvétele: 11. ikon.Nincs megvalósítva.

Ponthalmaz felvétele: 12. ikon. Nincs megvalósítva.

Háromszöglap felvétele: 13. ikon.

Háromszöglista felvétele: 14. ikon.

Gömb felvétele: 15. ikon. Nincs megvalósítva.

Alakzatok mozgatása: 2. ikon. Ha nincs kijelölt alakzat,

Alakzatok kijelölése: 3. ikon.

4.5 Transzformációk a felhasználói felületen

5. ábra: Transzformációk eszköztár

Csakúgy mint az alakzatoknál, a transzformációk esetében is behatároltabbak a felhasználói felület adta lehetőségek, mint a szkript nyelvé.

Mindig az aktuálisan kijelölt elemekre vonatkozik a felhasználói felületen felvehető összes transzformáció.

4.5.1 Eltolás

2. ikon a transzformációk eszköztáron. Az eltolás vektorának koordinátáit kell megadni. A szkriptnyelv is lehetőséget ad rá.

4.5.2 Koordináta-transzformáció

3. ikon a transzformációk eszköztáron. Minden egyes koordinátához meg kell adni egy függvényt. Az egyes koordinátákra való hivatkozás: p[*index*], ahol index a koordináta 0 alapú sorszáma. Pl. 3 dimenziós térben a x = x + 1; y = 2 * y; z = z + sin(x + y) transzformáció megadásához rendre: p[0] + 1; 2 * p[1]; p[2] + sin(p[0] + p[1]).

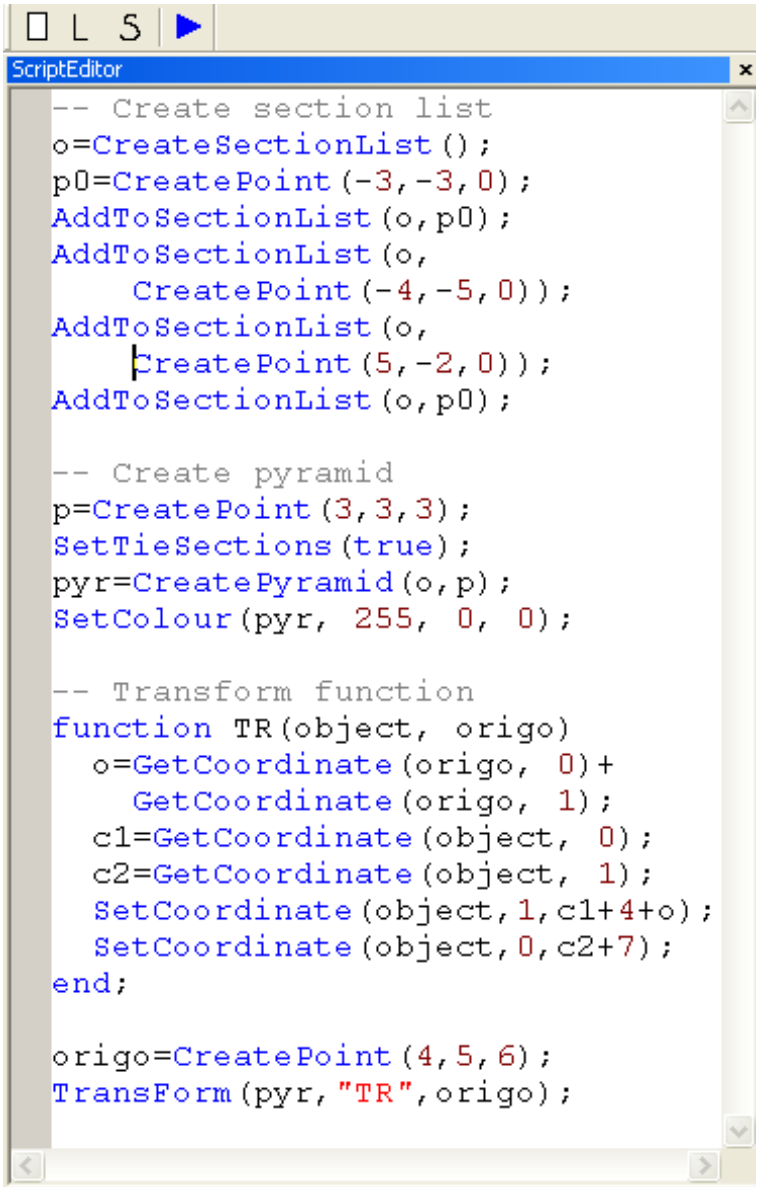
A szkriptnyelv is lehetőséget ad rá.

4.5.3 Forgatások

4. ikon a transzformációk eszköztáron. Alapforgatás felvételéhez azt a két dimenziót kell megadni, amelyben a forgatás történik, majd a forgatás szögét. A szkriptnyelv is lehetőséget ad rá.

4.6 Szkript

A programban a legtöbb lehetőséget a szkriptnyelv biztosítja mind az alakzatok, mind a transzformációk felvételében, így ennek tárgyalása a legrészletesebb.



6. ábra: Szkript szerkesztő

Ezért a szkriptek írására, szerkesztésére, a dokkolható ablakban megjelenő, szkript szerkesztő biztosít felületet. A kód áttekintését szintaktikus színezés segíti: a felismert kulcsszavak kékkel, a megjegyzések szürkével jelennek meg.(6. Ábra)

4.6.1 Megnyitás, mentés, futtatás

Új szkriptet a szkript eszköztáron balról az első ikonnal kezdhetünk(6. ábra).

Szkriptek megnyitása, mentése a második, harmadik ikon.

Az aktuális kód futtatása pedig a jobbszélső gomb.

4.6.2 Lua alapok

A program szkriptnyelve LUA alapokon nyugszik, ezért ez az alfejezet röviden bemutatja ezt a nyelvet.

A LUA egy szabad liszenszű, alkalmazásokba könnyen beágyazható szkriptnyelv. Az itt tárgyaltaknál sokkal komolyabb lehetőségeket biztosít, az interneten bőséges dokumentáció áll rendelkezésre róla(www.lua.org).

4.6.2.1 Változók

Deklarálás:

```
local változónév;
A local kulcsszó inicializálásnál elhagyható:
változónév=5;
```

Alaptípusok:

```
szám, szöveg, boolean
n=5; -- szám típusú változó deklarálása értékadással
s=”szoveg”; -- szöveg típusú változó deklarálása értékadással
b=true;
```

Asszociatív tömb:

```
tomb1={2,4,6,8};
tomb2 = {x = 200, y = 300, nev = "a" tipus = "pont"};
```

Hivatkozás tömbelemre:

```
tomb1[1]        tomb1 1. eleme, erteke 2;
tomb2["x"]      tomb2 x eleme
tomb2.x tomb2 x eleme
```

Megjegyzés:

```
-- karakterek után irt szöveg
```

4.6.2.2 Vezérlési szerkezetek

For-ciklus szintaktikája:

```
for i=start, end, count do
        ...
end;
```

i a ciklusváltozó, start-tól end-ig count lépésközzel

While ciklus szintaktikája:

```
while F do
        …
end;
```

Feltételes elágazás

```
if F1 then
        ...
elseif F2 then
        ...
else
        ...
end;
```

Függvény megadása:

```
function FN(param1, param2, …)

end;
```

Matematikai függvények a math könyvtárban:

```
math.sin
math.cos
math.abs
```

4.6.3 GeoTransz szkript nyelve

4.6.3.1 Egyszerű alakzatok létrehozása

OBJECT CreatePoint(N1, N2,...)

Leírás: Létrehoz egy pontot a megadott koordinátákkal
Paraméterek:
N1: a pont 1. koordinátája
...
Visszatérési érték: a létrehozott pont

OBJECT CreatePoint(VECTOR)

Leírás: Létrehoz egy pontot a megadott vektor koordinátáiból
Paraméterek:
VECTOR: a vektor.
Visszatérési érték: a létrehozott pont

VETOR CreateVector(N1, N2,...)

Leírás: Létrehoz egy vektort a megadott koordinátákkal
Paraméterek:
N1: a vektor 1. koordinátája
...
Visszatérési érték: a létrehozott pont

SECTION CreateSection(POINT1, POINT2)

Leírás: Létrehoz egy szakaszt a két pont között.

TRIANGLE CreateTriangle(POINT1, POINT2, POINT3)

Leírás: Létrehoz egy kitöltött háromszöget.

OBJECT CreateWireTriangle(POINT1, POINT2, POINT3)

Leírás: Létrehoz egy kitöltetlen háromszöget

CIRCLE CreateCircle(POINT, dim1, dim2, N)

Leírás: Létrehoz egy kört
Paraméterek:
POINT: a kör középpontja
dim1, dim1: a kör befoglalósíkjának dimenziói
N: a kör sugara

SECTION_LIST CreateSectionList()

Leírás: Létrehoz egy üres töröttvonalat.

AddToSectionList(SECTION_LIST, POINT)

Leírás: Hozzáad egy pontot a megadott töröttvonal végéhez.

OBJECT CreateTriangleList()

Leírás: Létrehoz egy üres háromszöglistát.

OBJECT AddToTriangleList(POINT)

Leírás: Hozzáad egy pontot a megadott háromszöglista végéhez.

4.6.3.2 Összetett alakzatok létrehozása

OBJECT CreatePyramid(SECTION_LIST, POINT)

Leírás: Létrehoz egy gúlát a megadott töröttvonalra, mint alapra, POINT csúccsal.

OBJECT CreateSet()

Leírás: Létrehoz egy üres objektum halmazt.

AddToSet(SET, OBJECT)

Leírás: Hozzáad egy objektumot a megadott halmazhoz.

OBJECT CreateClone(OBJECT)

Leírás: Létrehozza a megadott objektum egy másolatát.
Visszatérési érték: a létrehozott másolat.

OBJECT CreateTieClone(OBJECT)

Leírás: Hasábosítás. A hasábosítás tulajdonságait a SetTie... parancsokkal allithatjuk be.
Meg nem teljes funkcionalitasu!!!! ----------TODO

OBJECT MakePointCollection(OBJECT, bool)

Leírás: Az objektumból ponthalmazt készít. Bool true esetén törli az eredeti objektumot.

OBJECT **GetParts**()

Leírás: Egy objektum részeit adja vissza halmaz formájában. Pl: egy szakasznak a két végpontját.

OBJECT **GetPoints**()

Leírás: Egy objektum részei közül csak a pontokat adja vissza egy halmazban.

4.6.3.3 Összetett alakzatok viselkedését módosító függvények

SetTiePoints(bool)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. true esetén a megfelelő pontokat összeköti szakasszal, false esetén nem.

SetTieSections(bool)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. true esetén a megfelelő szakaszokat összeköti egy felülettel, false esetén nem.

SetTieAccuracy(N)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. Az összekötő objektumok iterációsűrűségeét adja meg.

SetTieLineWidth(N)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. Az összekötő objektumok vonalvastagságát adja meg.

SetTiePointColour(r, g, b)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. Az összekötő szakaszok színét adja meg.
Paraméterek:
r: Piros komponens (0-255)
g: Zöld komponens (0-255)
b: Kék komponens (0-255)

SetTiePointShowPoints(bool)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. Az összekötő szakaszok ShowPoint tulajdonságát állítja be.

SetTieSectionColour(r, g, b)

Leírás: CreateTieClone és a CreatePyramid viselkedését állítja be. Az összekötő felületek színét adja meg.
Paraméterek:
r: Piros komponens (0-255)
g: Zöld komponens (0-255)
b: Kék komponens (0-255)

SetTieSectionShowPoints(bool)

Leírás: CreateTieClone és a CreatePyramid viselkedését módosítja. Az összekötő felületek ShowPoint tulajdonságát állítja be.

4.6.3.4 Objektumok tulajdonságai

SetColour(**OBJECT**, r, g, b)

Leírás: Az objektum színét beállítja a megadott RGB szerint.
Paraméterek:
r: Piros komponens (0-255)
g: Zöld komponens (0-255)
b: Kék komponens (0-255)

SetShowPoints(bool)

Leírás: Beállítja, hogy láthatóak legyenek az objektum pontjai. Pl szakasz esetén egybefüggő vonalként jelenjen meg, vagy adott sűrűségű pontsorozatként.
Paraméterek:
bool: true esetén pontok halmazaként jelenik meg.

SetLineWidth(**OBJECT**, N)

Leírás: Vonalvastagság. Pont esetén a pontábrázolás típusa. (csak ShowPoints hamis esetben)
Paraméterek:

SetHidden(**OBJECT**, bool)

Leírás: Elrejti vagy láthatóvá teszi az objektumot.
Paraméterek:
bool: true esetén rejtett, false esetén látható.

SetPointsPerUnit(**OBJECT**, N)

Leírás: az iteráció sűrűségét N pont/egységre állítja. Ha S egy 1 egység hosszú szakasz, a SetPointsPerUnit(S, 10) hívás után a szakaszt 10 pont reprezentálja.

SetName(**OBJECT**, STRING)

Leírás: beállítja egy objektum cimkéjét.

4.6.3.5 Objektumok kezelése

OBJECT **GetObject,**(name)

Leírás: címke alapján visszaad egy objektumot----------TODO
Paraméterek:
N1: a pont 1. koordinátája
...
Visszatérési érték: az adott cimkéjű elem

OBJECT **GetSelectedObjects**()

Leírás: Az éppen kijelölt alakzatot adja vissza.

Visszatérési érték: egyszeres kijelölés esetén a kijelölt objektumot, egyébként a kijelöltek halmazát.

STRING **GetClass**(**OBJECT**)

Leírás: visszaadja egy objektum tipusat (POINT, SET, SECTION_LIST, TRIANGLE,...)

OBJECT SetCoordinate(POINT, dim, value)

Leírás: Beállítja egy pont-vector egy koordinátáját.
Paraméterek:
POINT: a pont
dim: a koordináta sorszáma (dimenziószám)
value: a beállítandó érték

OBJECT GetCoordinate(POINT, dim)

Leírás: Visszaadja egy pont vagy vektor egy koordinátáját.
Paraméterek:
POINT: a pont
dim: a koordináta sorszáma (dimenziószám)
Visszatérési érték: a kérdéses koordináta értéke.

4.6.3.6 Transzformációk létrehozása

SetTransformImage(bool)

Leírás: Transzformációk viselkedését állítja be.
True esetén létrehoz egy képobjektumot és egy transzformációt; az eredeti alakzat mozgatása esetén a képhalmaz mindig a transzformált objektum aktuális állapotának transzformáltja.
false esetén létrehoz egy, az eredeti objektum aktuális állapotából egy képhalmazt, mely a transzformáció után nem követi az eredeti objektum változásait, egy független ponthalmaz lesz.
----------TODO

OBJECT **Move**(OBJECT, VECTOR)

Leírás: Eltolás adott vektorral.

OBJECT **Rotatde2D**(OBJECT, dim1, dim2, angle)

Leírás: 2d forgatás megadott koordináták mentén adott szöggel.

OBJECT **Transform**(OBJECT, FN)

Leírás: a megadott objektumot transzformálja a megadott függvénnyel.
A transzformáló függvény szintaxisa: FN(VECTOR). Az egyetlen paraméter a transzformálandó pont vektorként. A függvénynek nincs visszatérési értéke, a vektort helyben módosíthatja.

Paraméterek:
OBJECT: a pont
FN: a transzformáló függvény neve (idézőjelek között megadva)
Visszatérési érték: a transzformált objektum.

Az alábbi **példa** a pyr objektum első és második koordinátáját kicseréli.

```
-- Transform function
function TR(object)
  c1=GetCoordinate(object, 0);
  c2=GetCoordinate(object, 1);
  SetCoordinate(object,1,c2);
  SetCoordinate(object,0,c1);
end;

TransForm(pyr,"TR");
```

OBJECT **Transform**(OBJECT, FN, OBJECT)

Leírás: a megadott objektumot transzformálja a megadott függvénnyel
A transzformáló függvény szintaxisa: FN(VECTOR, OBJECT). Az első paraméter a transzformálandó pont vektorként, a második a transzformációhoz tartozó objektum. A függvénynek nincs visszatérési értéke, a vektort helyben módosíthatja.

Paraméterek:

OBJECT: a pont

FN: a transzformáló függvény neve (idézőjelek között megadva)

A transzformációhoz tartozó objektum (pl középpontos tükrözésnél a középpont)

Visszatérési érték: a transzformált objektum.

Az alábbi **példa** a pyr objektum első koordinátáihoz hozzáadja a középpont első és második koordinátájának összegét.

```
-- Transform function
function TR(object, origo)
  o=GetCoordinate(origo, 0)+GetCoordinate(origo, 1);
  c1=GetCoordinate(object, 0);
  c2=GetCoordinate(object, 1);
  SetCoordinate(object,1,c1+o);
end;

origo=CreatePoint(4,5,6);
TransForm(pyr,"TR",origo);
```

ForEach(OBJECT, FN)

Leírás: Egy halmaz minden elemére végrehajtja a megadott függvényt.

(N, VECTOR) **GetDistance**(OBJECT, POINT)

Leírás: Egy objektum es egy pont távolságát adja vissza, valamint az objektumhoz legközelebbi pontot. Elsősorban tükrözések megvalósítására szánt fv.

5 Geometriai háttér

Hogy lehet háromnál több dimenziójú térről beszélni? Hogyan építhető fel az n dimenziós geometria axiomatikusan? Mi történik a megszokott vektorműveletekkel, transzformációkkal több dimenzióban? Ezekre a kérdésekre ad választ ez a fejezet először a geometria, majd a lineáris algebra eszközeivel.

A fejezetnek nem célja az N dimenziós geometria egzakt matematikai bevezetése, csupán ízelítőt ad abból, hogyan lehetne ezt megtenni.

5.1 Az N dimenziós tér geometriai bevezetése

Ez az alfejezet azt mutatja be, hogyan lehetne felépíteni N dimenziós geometriát az euklideszi axiómákra támaszkodva.

Szándékosan nem a szokásos, lineáris algebra útján történő bevezetést választottam, ehelyett előző munkámhoz hasonlóan megpróbáltam csak az általános és középiskolai ismeretekre támaszkodni. A következő fejezet említést tesz a szokásos útról.

5.1.1 N dimenziós tér fogalma

A sík és a tér fogalmához tapasztalásból, absztrakcióval jutunk el; az n dimenziós tér fogalmához pedig a sík és a tér általánosításával, dimenziós analógia révén.

Egy pont meghatározásához például egy dimenzióban egy, két dimenzióban két három dimenzióban három koordinátára van szükségünk. Máshogy kifejezve: 1 dimenzióban jobbra/balra tudunk mozogni, két dimenzióban már fel/le, háromban előre/hátra is. A felsorolt három tengely-menti mozgás közül egyiket sem tudja pótolni másik kettő. Azt is megállapíthatjuk, hogy ezen tengelyek egymásra merőlegesek. Általánosságban mindig annyi tengely-mentén tudunk mozogni, ahány dimenziós térben vagyunk.

Negyedik dimenzióban tehát egy negyedik tengelyt képzeljünk el, aminek a mentén olyan mozgást tudunk végezni, ami az eddigi háromféle mozgással nem pótolható. Olyat, amely az

eddigiekre merőleges; egy olyan irányba, amit mi nem látunk.

N dimenziós térben n darab tengelyt képzeljünk el, melyek mindegyike mentén olyan mozgás lehetséges, ami a fennmaradó n-1 -féle mozgással nem pótolható, és ezek egymásra páronként merőlegesek.

Abból az intuitív kérdésfeltevésből indultunk ki, hogy hányféle kiterjedése van az adott térnek. Ez, mint később látni fogjuk, pontosan is megfogalmazható a lineáris algebra eszközeivel.

5.1.2 Alapfogalmak, axiómák

A sík és a tér alapfogalmaihoz, axiómáihoz ugyancsak tapasztalásból, absztrakcióval jutunk el. N dimenziós geometria felépítésénél nem támaszkodhatunk tapasztalatainkra, **a magasabb dimenziójú terekhez a sík és a tér vektorainak általánosításával jutunk el.**

Olyan n dimenziós geometriát szeretnénk építeni, ahol a két és három dimenziós geometria eddig megszokott szabályai érvényesek, ezért ugyanazokból az alapfogalmakból és axiómákból indulunk ki, és kibővítjük azokat analógia révén.

Szükségünk lesz az n dimenziós teret, térelemeket leírni képes szókincsre. Ami két dimenzióban az egyenes, három dimenzióban a sík, n-dimenzióban az **n-1 dimenziós altér**. Ezzel a szóhasználattal az egyenes a 3 dimenziós tér 1 dimenziós altere.

A félegyenes, félsík, féltér fogalmát általánosan **n dimenziós féltérnek** hívjuk, a félsík tehát 2 dimenziós féltér.

Ahol az egyértelműség nem csorbul, n dimenziós tér(altér) helyett egyszerűen azt mondjuk, tér(altér).

Nézzük a térelemek illeszkedésére vonatkozó axiómákat Hajós György: Bevezetés a geometriába c. könyvéből:

I. Két ponthoz egy és csak egy egyenes illeszkedik.

II. Ha három pont nincs egy egyenesen, akkor egy és csak egy sík illeszkedik hozzájuk.

III. Ha egy sík tartalmazza egy egyenes két pontját, akkor tartalmazza a teljes egyenest is.

Szükségünk van tér, több dimenziós tér illeszkedéséhez is axiómára:

- Ha négy pont nincs egy síkon, akkor egy és csak egy 3-dimenziós tér illeszkedik hozzájuk.

Általánosan:

- Ha n+1 pont nincs egy n-1 dimenziós altéren, akkor egy és csak egy n-1 dimenziós altér illeszkedik hozzájuk.

Félegyenesre, félsíkra, féltérre bontás axiómái:

IV. Egy pont a rajta áthaladó egyenest két félegyenesre bontja fel. Az egyenes e ponton áthaladó szakaszának a végpontjai más-más félegyeneshez tartoznak. Az egyenes minden más szakaszát az egyik félegyenes tartalmazza.

V. Egy egyenes a rajta áthaladó síkot két félsíkra bontja fel. Az egyenest metsző síkbeli szakasznak a végpontjai más-más félsíkhoz tartoznak. A sík minden más szakaszát legalább az egyik félsík tartalmazza.

VI. Egy sík a rajta áthaladó teret két féltérre bontja fel. A síkot metsző térbeli szakasz végpontjai más-más féltérhez tartoznak. A tér minden más szakaszát legalább az egyik féltér tartalmazza.

- Egy n-1 dimenziós ***altér*** a rajta áthaladó n dimenziós ***ter***et két n dimenziós ***féltér***re bontja fel. Az *alteret* metsző térbeli szakasz végpontjai más-más *féltér*hez tartoznak. A *tér* minden más szakaszát legalább az egyik *féltér* tartalmazza.

A mozgásra vonatkozó megállapítások:

VII. A mozgás két pont összekötő szakaszát a két elmozgatott pont összekötő szakaszába, az egyenest egyenesbe és a síkot síkba viszi.

VIII. Egy és csak egy olyan térmozgás van, amely egy adott félsíkot és ennek határán adott félegyenest megadott helyzetbe, egy adott félsíkba és annak határán adott félegyenesbe visz át.

- A mozgás két pont összekötő szakaszát a két elmozgatott pont összekötő szakaszába, az n dimenziós *alteret* n dimenziós *altérbe* viszi.
- Egy és csak egy olyan térmozgás van, amely egy ***adott helyzetű*** *félteret* ***megadott helyzetbe**** visz át.

Egy n dimenziós féltér *féltér* ***megadott*** helyzetű, ha a határán adott egy n-1 dimenziós féltér,..., és ennek határán egy 2 dimenziós féltér.

Az alábbi megállapítások nem szorulnak kiegészítésre:

IX. Egy szakaszt bármely belső pontja két olyan szakaszra bont fel, amelyek hosszának összege az eredeti szakasz hossza.

X. Ha a hosszegység adott, akkor bármely A kezdőpontú félegyenesen egy és csak egy olyan B pont található, amelyre nézve az AB távolság egy adott pozitív valós szám.

Végül az ötödik posztulátumként ismert

XI. Párhuzamossági axióma: Egy ponton át egy és csak egy olyan egyenes halad, amely egy a ponton át nem haladó egyenessel párhuzamos.

5.1.3 Vektor-fogalom

A vektort a geometria irányított szakaszként vezeti be. Ahhoz, hogy a vektorokat számokkal jellemezzük, szükségünk van egymásra merőleges egységvektorokra. Síkban $\mathbf{e_1}$, $\mathbf{e_2}$, térben $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}$, analógia révén az n dimenziós térben $\mathbf{e_1}$, $\mathbf{e_2}$... $\mathbf{e_n}$. Az n dimenziós tér tetszőleges vektorát ezen vektorok lineáris kombinációjaként előállíthatjuk: $\mathbf{v}= v_1*\mathbf{e_1} + v_2*\mathbf{e_2}+...+ v_n*\mathbf{e_n}$. Ezért, ha adottak az egységvektorok, a **v** vektort a v_1, v_2... v_n számokkal jellemezhetjük,

röviden $\mathbf{v}(v_1, v_2... v_n)$.

5.1.4 Vektorműveletek

Analógia révén a következő alapvető vektorműveletekhez jutunk (definíció):

Összeadás:

$\mathbf{v}(v_1, v_2...v_n) + \mathbf{w}(w_1, w_2, w_n)=\mathbf{z}(v_1+w_1, v_2+w_2...v_n+w_n)$

Skalárral való szorzás:

$a*\mathbf{v}(v_1, v_2...v_n)=\mathbf{v}(a*v_1, a*v_2...a*v_n)$

5.2 Vektorterek, az euklideszi tér

A lineáris algebra foglalkozik a vektorok általánosításával, a vektorterekkel. Az intuitív bevezetőhöz hasonlóan képzelhetjük el a vektortereket. Egy vektortér nem más, mint absztrakt elemek halmaza, melyek a geometriai vektor-fogalomból általánosított tulajdonságokkal rendelkeznek, és a vektorok között megszokott műveleteket lehet rajtuk végezni.

A vektortér bázisa tartalmazza az irányokat, amik mentén mozoghatunk: olyan vektorok, amelyek lineáris kombinációjából a tér minden pontjához eljutunk. Egy n dimenziós tér bázisa n egymásra merőleges vektor, az egységvektorok. A szükséges vektorműveleteket a vektortér definíciója biztosítja.

A vektorterek segítségével könnyen és szépen definiálható az N dimenziós euklideszi tér és vektorai, vektorműveletei. Mindehhez egy valós számtest feletti, N dimenziós vektorteret kell definiálni egy skaláris szorzással, ezt nevezik Euklideszi térnek.

Ennek részletei megtalálhatóak a http://en.wikipedia.org/wiki/Euclidean_space lapon.

5.3 Transzformációk

Ez a fejezet ízelítőt ad a legalapvetőbb geometriai transzformációk N dimenziós térben való értelmezéséből.

5.3.1 Eltolás

Teljesen analóg a két/három dimenziós térben való forgatáshoz:

Definíció: Adott egy n dimenziós v(v1, v2...vn) vektor. Az eltolás egy olyan transzformáció, melyben egy tetszőleges pont képét a P kezdőpontú v vektor végpontja adja meg.

Definíció (koordinátageometriai): Egy p(p1,p2...pn) pont v(v1, v2...vn) vektorral való eltolás utáni képe a p'(v1+p1, v2+p2...vn+pn) pont.

5.3.2 Forgatások N dimenzióban

A forgatás intuitív dimenziós analógiája valahogy így hangozna: két dimenzióban egy pont körül forgatunk, három dimenzióban egy egyenes-tengely mentén, négy dimenzióban egy sík mentén... De hogy lehet egy sík mentén vagy egy tér mentén forgatni? Ez már idegen a hétköznapi szemléletünktől.

Célravezetőbb a három dimenziós forgatást máshogy értelmezni. Egy pont elforgatása egy tengely mentén tulajdonképpen egy maximum két dimenziós művelet: ha a pont nincs a tengelyen, abban a síkban zajlik, amely a tengelyre merőleges, és áthalad a ponton. A forgatás középpontja a tengely és a sík metszéspontja. A forgatásnál egy pont mindig egy körvonal mentén halad. Persze különböző pontok forgatása nem ugyanazon a körvonalon, nem feltétlenül ugyanabban a síkban, ugyan akörül a pont körül zajlik. Ebből már érezhető, hogy mi is az a forgatás tetszőleges dimenziójú térben.

Az analógiából feltűnő, hogy n dimenzióban egy n-2 dimenziós altér körül forgatunk, ezt nevezzük a forgatás tengelyének. Ez azon pontok halmaza, amelyek helybenmaradnak, másképpen **fixpontjai** a transzformációnak.

Minden n dimenziós forgatáshoz van tehát egy n-2 dimenziós altér, mely helybenmarad.

Ez azt is jelenti, hogy ha egységvektorainkat megfelelően vesszük fel, a transzformáció az alakzatok n-2 db koordinátáján nem változtat. A másik oldalról, azon pontok, melyek nem fixpontok, illeszkednek egy 2 dimenziós altérre, azaz egy síkra. Azon pontok tehát, melyeknek az előbbi bázissal a fennmaradó két koordinátája nem nulla. Illetve, a tér bármely pontára igaz, hogy az adott forgatás csak e szóban forgó két koordinátáján változtat.

Így már egyszerű n dimenzióban elképzelni a forgatást. Egy n-2 dimenziós forgástengelyt meghatározhatunk egy két dimenziós altér komplementertereként. Ha adott egy N dimenziós tér és ennek egy bázisa, azaz n egymásra merőleges egységvektora, ennek alapforgatásai tetszőlegesen választott két tengely mentén működnek, tehát annyi van, ahányféleképpen ki tudunk választani n-ből kettőt. Ez az altér, egyszerűbben sík, amelynek pontjai nem maradnak helyben. Ezen alapforgatások szorzataként pedig a tér összes, origón áthaladó tengelyű forgatása előállítható. Ha a tengely nem halad át az origón, egy eltolással átvihető.

Összegezve az n dimenziós tér minden forgatása megkapható a fenti értelemben vett 2 dimenziós forgatások és eltolások szorzataként.

5.3.3 Tükrözések

A tükrözések ponttranszformációs definíciói teljesen analógok a két dimenziós testvéreikéhez. Mivel a programban sem kaptak nagy hangsúlyt a tükrözések, a definíciókat itt nem részletezem.

5.3.4 Koordináta-transzformáció

Definíció (koordinátageometriai): Adott f_1, f_2... f_n $\mathbf{R^n \to R}$ n változós valós függvény. Egy tetszőleges $P(x_1, x_2 \ldots, x_n)$ pont képe a $P'(x_1', x_2' \ldots, x_n')$, ahol $x_i' = f_i(x_1, x_2 \ldots, x_n)$, $i=1..n$.

6 Megvalósítás

Ez a fejezet röviden ismerteti a program megvalósítását.

6.1 Fejlesztő környezet, nyelv

A fejlesztés MS Visual C++ (2003) környezetben zajlott, felhasználói felület kivételével sztenderd C++ (ISO C++) nyelven készült. A nyelv választásában döntő szerepet játszott az esetleges portolhatóság és a számítások sebességigénye. Felhasználói felület egyelőre csak Windows alá készült, ez MFC alapú. Nem használja az MFC document-view architektúráját sem, erre saját, portolható c++ megoldást ad.

6.2 Lokalizáció

Egyedi a lokalizációs mechanizmus is. Tervezésénél főszempont volt, hogy azonnal használni lehessen, de a hosszú távú fejlesztéshez biztosítsa a következőket: dinamikus nyelvi bővíthetőség (fordítás nélkül, utólag hozzáadható/szerkeszthető nyelvi adatbázisokkal), szavak/kifejezések kezelése strukturált legyen, gyorsaság, gyors és egyszerű fejlesztés, áttekinthető kód, nagyobb projektekhez automatizálható a kódból az ID gyűjtés és karbantartás.

Bővebben: x\L10N\index.html

6.3 Template felépítés

Annak érdekében, hogy a programban dinamikusan változtatható legyen a számítások pontossága (nagyobb pontosság a sebesség rovására), a számítások típusa template paraméterként végigvonul a legtöbb osztályon.

A pontosság változtathatósága utólag nem tűnik fontosnak, és sok esetben átláthatatlanná teszi a kódot; így további fejlesztések előtt érdemes kiszedni.

6.4 A program részei

Az alábbiakban ismertetem a forráskód főbb részeit és ezek funkcióit.

6.4.1 Math

A programban használt matematikai típusok, műveletek megvalósítása. Itt található az n dimenziós vektor osztály, ennek műveletei, és ezekre épülő típusok, segédfüggvények, konverziós függvények, továbbá az n dimenziós tér, alterek és ezek projekciói 2 dimenzióra.

6.4.2 View

A nézet megvalósítása. A görgethető nézet biztosítja a 2 dimenziós logikai-fizikai koordináták közötti konverziót, kezeli a kicsinyítést-nagyítást, görgetést, a munkaterület tetszőleges megjelenítését. A ScrollView projektív nézetekek kezelésére alkalmas, a programban jelenleg minden nézet projektív. Látványos 3D megjelenítéshez e mellé lehet majd beilleszteni például egy OpenGL-es nézetet.

6.4.3 Display

A Display osztály és leszármazottai a fizikai megjelenítésért felelősek. Ponttíposok, vonal, háromszög, szöveg kirajzolása adott színnel, vonalvastagsággal, 2 dimenzióban. Az MFCDisplay az MFC megvalósítás, más platformra ehhez hasonlóan kell a BaseDisplay-t implementálni.

Itt található a Visualizer is, ami a logikai és a fizikai megjelenítést köti össze: n dimenziós alakzatokat jelenít meg egy Display objektum segítségével. Ennek az egyik megvalósítása a ProjectiveVisualizer, projektív nézetekhez.. Itt jelenleg a megjelenítés azonnali, ha objektumokat takarással kívánunk megjelíteni, ide beilleszthető egy mélység szerinti rendezés.

Egy N dimenziós alakzat megjelenítésének folyamata ezek alapján a következőképpen történik:

ND alakzat --(**Visualizer**)--> --(**View**)-->*2D alakzat* --(**Display**)-->*kirajzolt alakzat*

6.4.4 GTDoc

Az egyedi dokumentum-nézet architektúra dokumentum osztálya. Egy munkaterülethez tartozó alakzatokat, transzformációkat és egyéb adatokat tárolja. Az itt tárolt objektumokat több nézet megjelenítheti.

6.4.5 GTGeoObjects

A program geometriai objektumai. Minden megjeleníthető objektum kirajzolja magát, ezen felül minden megjeleníthető objektum kezeli az őt érintő eseményeket (alakzatfelvétel, mozgatás) is. Erre szolgál a Displayable osztály.

Az alakzatok, ponthalmazok ősosztálya a BaseGeoObject. Ebből származnak a geometriai alakzatok, a pont, szakasz, háromszög, háromszöglista, töröttvonal, kör, halmaz, képhalmaz.

Minden objektumnak biztosítania kell egy iterátort, amely az alakzatot adott pont/egység pontossággal(PointPerUnit) pontonként adja vissza. Ez az iterátor biztosítja többek között a pontonkénti transzformálást, kirajzolást, általános távolságfüggvényt.

6.4.6 LuaVM

A lua alapú scriptnyelv megvalósítása. Itt található a Lua 5.1 libre épülő GeoTrans specifikus lua függvények regisztrációja, c++ implementációja, és egyéb lua-val kapcsolatos segédfüggvények, objektumok.

A LuaGC a geometriai objektumok kezelésére szolgál: paraméterek átadását, objektumok

szemétgyűjtését végzi. A lua függvények a c++ objektumokat referenciaszámlált shared_ref nyers c++ mutatójaként kapják meg. Paraméter átadáskor új shared_ref objektumot hoz létre, és tárolja a mutató típusát, így gondoskodik az objektum létezéséről, típus szerinti helyreállításáról, és típusos törléséről.

A LuaVM pedig a lua virtuális gép objektuma, mely egy szkript futtatása során fellépő hibákat ablakban megjeleníti.

6.4.7 GeoTranszND

A felhasználói felület. MFC alapú, de az MFC Document-view architektúráját nem használja.

6.4.8 Egyéb

- STLX – hasznos Standard Template Library kiegészítések
- MFCX – hasznos MFC kiegészítések
- Unicode - lib az UTF kódolású szövegek kezelésére
- L10N – lokalizációs könyvtár
- YTL – saját template lib. Itt található egy referenciaszámlált smart-pointer, a shared_ref (boost:shared_ptr-en alapul.)
- FSManager – portolható file-system manager az alapvető fileműveletekhez. Nincs megvalósítva.
- Serializer – A munkaterület szerializálása; boost alapokra terveztem. Nincs megvalósítva.

7 A program továbbfejlesztési lehetőségei

A tervezésben nagy szerepet kapott a moduláris felépítés, a fejlesztés során pedig igyekeztem olyan platformfüggetlen megoldásokat kidolgozni egy-egy problémára, mely már használható az adott szinten is, de kompromisszumok nélkül továbbfejleszthető.

A program továbbfejlesztési lehetőségei:

- további alapalakzatok felvétele: gömb, kocka, henger, kúp
- polárkoordináták kezelése
- magasabb szintű algoritmusok alakzatfelvételre:
  - konvex lezárt generálása
  - testek felvétele határolósíkokkal
  - forgástestek generálása adott pontossággal
- magasabb szintű algoritmusok alakzatok kezelésére:
  - alakzat egy pontjának mozgatásával a környező pontok is mozogjanak
  - síkidomok háromszögekre bontása
  - határolósíkok kezelése
  - testek metszete, uniója
- projektív nézetekhez z-order algoritmus
- további nézetek:
  - OpenGL-es 3D nézet
  - forgatható több dimenziós nézetek
- szkriptnyelv fejlesztése:
  - vektorműveletek
  - mátrixműveletek

- lokalizáció
- szerializáció

8 Források

8.1 Felhasznált irodalom

1. Hajós György: Bevezetés a geometriába, Nemzeti Tankönyvkiadó, Budapest 1999
2. Wikipedia (en.wikipedia.org)
3. www.lua.org – A Lua alapok fejezethez
4. GeoTransz dokumentáció, Bánáti-Baumann Zsolt, 2003

8.2 A program fejlesztéséhez felhasznált könyvtárak, szabad forráskódok

1. Microsoft Foundation Class Library (MFC) – felhasználói felületek fejlesztésére
2. Boost (www.boost.org) – C++ könyvtár
3. Lua 5.1 (www.lua.org) – szkript nyelv
4. CSizingControlBar (Cristi Posea) – dokkolható ablakok
5. CrystalEdit (Stcherbatchenko Andrei) – szintaktikus szövegszerkesztő