SZAKDOLGOZAT

Manó Tamás

Debrecen
2007

Debreceni Egyetem

Informatika Kar

OTATÓPROGRAM A FÜGGVÉNYEK ÉS TESTEK TÉMAKÖRÉT FELDOLGOZÓ TANÓRÁKHOZ

Témavezető:

Nyakóné Juhász Katalin

tudományos főmunkatárs

Készítette:

Manó Tamás

matematika-informatika szakos hallgató

Debrecen

2007

TARTALOMJEGYZÉK

1. BEVEZETÉS

Már középiskolás koromban is vonzott a matematika és az informatika tantárgy, mindig is vágyódtam a bennük rejlő lehetőségek megismerésére, kiaknázására. Éppen ezért is választottam a matematika-informatika szakot, kihívást és kíváncsiságot éreztem e tudományok iránt.

Az egyetemen töltött évek alatt rengeteg tantárggyal találkoztam, sok tapasztalatot gyűjtöttem, és egyre inkább letisztult bennem az-az irány, amely a leginkább érdekel, a leginkább megfogott. Az informatika oldaláról a programozás az, amely a legnagyobb érdeklődést váltotta ki belőlem. Amikor elkezdődtek az első programozási órák, már akkor megfogalmazódott bennem a kérdés, hogyan lehet „saját kezűleg" elkészíteni egy grafikus felhasználói felülettel rendelkező programot. Az idő múlásával természetesen erre is fény derült. Találkoztam a Java programozási nyelvvel, és úgy éreztem ez a nyelv tökéletes arra, hogy elkészíthessem vele az első komolyabb programomat. Ekkor még fogalmam sem volt arról, hogy milyen programot kellene írnom.

Mindig is vizuális ember voltam, szeretem a dolgokat elképzelni, s a lehetőségekhez képest mindig valamilyen képet, képzetet társítok azokhoz. A tanulmányaim során, és a vizsgákra való készülések alkalmával is, mindig segítségül hívtam a vizualitást. Elérkezett a szakdolgozat ideje, amikor is megláttam a választható témák között a következőt: *Oktatóprogram készítése tetszőleges programozási nyelven*. Ekkor állt össze bennem a kép. Készítek egy oktatóprogramot, amelyben a vizuális képeket a program segítségével megvalósíthatom. Már csak azt kellett kitalálnom, hogy milyen elképzelt dolgokat képezhetnék le a programban. Úgy gondoltam, hogy a függvények és a különféle testek erre a célra tökéletesnek.

A célom tehát egy olyan oktatóprogram készítése, amelynek segítségével a tanórán, és azon kívül is hozzájárulhatok, a tanulók matematikai ismereteinek bővítéséhez. Természetesen a matematika egy igen „tág" tudományterület. Ez az oktatóprogram a matematika egy szűk ismeretkörének elsajátításában, megértésében nyújt segítséget. Amely egyrészt különböző függvények képének, alakjának a megismerésében, felfogásában, illetve azok tulajdonságainak a vizsgálatában nyújthat nagy segítséget. Másrészt néhány nehezen elképzelhető test szemrevételezésére, körbejárására, élethű képének ábrázolására is alkalmas. Illetve, a program a bemenő adatok birtokában, képes kiszámítani ezen testek felszínét és

térfogatát, természetesen a megfelelő képletek mögött megjelenítve azokat. Tehát a program alkalmas egyfajta önellenőrzésre is, mely vonatkozhat a képletek biztos tudására, illetve számítások eredményének összehasonlítására.

A program elkészítéséhez szükséges tapasztalataim egy részét, a gyakorló tanításaim közben szereztem meg. Ezen tapasztalatok alatt, nem a program magírásához szükséges szakmai tapasztalatokat, illetve tudást értem, hanem a program használata során felmerülő módszertani szükségletekre és kihívásokra gondolok. Továbbá arra, hogy a tanulókat nézve, a félkész programomba milyen további elemeket, funkciókat „csempészhetnék” bele.

Többek között az egyik órán támadt egy nagyon jó ötletem, amikor is az egyik tanuló egy szemléltető eszközt fogott a kezében, miközben forgatta, nézegette azt. A tanulónak ezt a tevékenységét úgy képeztem le a programban, hogy minden testet az egér segítségével, annak vonszolásával tetszőleges irányba és sebességgel el lehet mozdítani, ezzel mintegy megteremtve a kapcsolatot a gyerek keze és az egér, mint a számítógéphez tartozó hardveres eszköz között.

Ez csak egy módszer az órai megfigyeléseimből, melynek segítségével próbálok rávilágítani a tanítási órák és az oktatóprogramok között állítható kapcsolatokra.

A program elkészítéséhez szükséges további ismereteknél a korábban megszerzett tudásomra, illetve megtapasztalt dolgokra támaszkodom.

Az oktatóprogram felhasználói felületét, használatát, működését, jellemzőit és az itt felsorolt célokat, módszereket mutatom be a szakdolgozatomban.

KÖSZÖNETNYÍLVÁNÍTÁS

Ezúton ragadom meg az alkalmat, és a helyet arra, hogy megköszönjem témavezetőmnek, Nyakóné dr. Juhász Katalinnak a szakdolgozat elkészítése során nyújtott közreműködését, segítségét.

2. A GRAFIKUS FELHASZNÁLÓI FELÜLET ÉS JELLEMZŐI

2.1. A MEGJELENÉSI STÍLUS

A Java programozási nyelv lehetőséget biztosít arra, hogy a felület minden komponensét egyszerre valamilyen közös esztétikai szempontnak megfelelően változtassuk, ezt nevezzük a megjelenítés stílusának (LAF – Look-And-Feel). Az oktatóprogram megjelenése platformfüggő, azaz a használt operációs rendszerre jellemző megjelenési formát követi.

Az alapértelmezett megjelenési stílus a komponensek platformfüggetlen megjelenítését teszi lehetővé. Ha tehát nem állítjuk be a programkódban a megjelenítés stílusát, akkor a program futtatásakor egy minden platformon azonos kinézetű felületet kapunk.

2.2. A MENÜSOR AZ EGYES MENÜK ÉS ALMENÜK

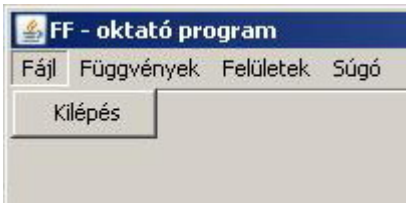
Egy program futása során, rendszerint valamilyen módon biztosítani kell a kommunikációt a program és a felhasználó között. A programok nagy részével a felhasználó interaktív módon kerül kapcsolatba, mely fokozottan érvényes az oktatóprogramok esetében. Szükség van tehát felhasználói felületre, amely megteremti a kapcsolatot a program és használója között, s mely lehetőséget biztosít adatok bevitelére, különféle műveletek elvégeztetésére, valamint az eredmények kijelzésére.

Az oktatóprogram elindítása után jelenik meg a főablak, ez az a felület, mellyel a felhasználó először találkozik. Felépítését a melléklet **1. ábrája** mutatja.
A főablak az alábbi menüsort tartalmazza:

Ez a menüsor természetesen minden, később még ismertetendő ablakból elérhető lesz.

2.2.1. A **Fájl** menü:

A *Fájl* menüből elérhető egyetlen almenü a *Kilépés*. Ezen almenü kiválasztásával lehetőségünk van a program elhagyására, bezárására.



2.2.2. A **Függvények** menü:

A *Függvények* menüből elérhető menüpontok kiválasztásával lehetőségünk nyílik különböző függvények megjelenítésére, azok módosítására és tulajdonságaik vizsgálatára.

Valamely almenüre klikkelve, a program a főablakról, a kiválasztott almenühöz tartozó ablakra vált. Ezen ablakokban a felhasználó kapcsolatba léphet a programmal, adatokat vihet be, és utasításokat adhat ki.

Azt, hogy éppen melyik függvénnyel, vagy függvénycsoporttal dolgozunk, azaz melyik függvényhez, függvényekhez tartozó modult használjuk, az ablak bal felső sarkában, a menüsor alatt lévő címke jelzi. Az ábrán látható, hogy az aktuális modul a T*rigonometrikus függvényrajzoló modul*.

Mivel az egyes függvények a felhasználótól, ugyanolyan típusú bemenő adatokat várnak - például adott függvény eltolása a koordinátarendszer x - tengelye mentén -, ezért az egyes menüpontok kiválasztásakor megjelenő felület minden esetben ugyanaz lesz. Felépítését a **2. ábra** szemlélteti.

- *Szögfüggvények* almenü:

  A szögfüggvények menüpontot választva, az alábbi trigonometrikus függvények vizsgálatára nyílik lehetőség: y=sin(x), y=cos(x), y=tg(x), y=ctg(x).

  Ahol, *x* – tetszőleges valós szám lehet.

- *Exponenciális függvény* almenü:

  Az exponenciális függvény menüpontot választva az e - alapú, valós exponenciális függvény tulajdonságait vizsgálhatjuk. Alakja: $y=e^x$.

  Ahol: $e \approx 2.72$ adott, pozitív valós szám, míg x tetszőleges valós szám.

- *Logaritmus függvény* almenü:

  A logaritmus függvény menüpontot választva a természetes alapú valós logaritmus függvény tulajdonságait vizsgálhatjuk. Alakja: y=ln(x).

  Ahol: x tetszőleges pozitív valós szám.

- *Hatványfüggvények* almenü:

  A hatványfüggvények menüpont alatt, $y=x^n$ - alakú függvények vizsgálatára van lehetőség, ahol: n - adott egész szám, x - pedig tetszőleges valós szám lehet.

  A felhasználó az n - értékét előre meghatározott { 10 , 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1 , 0 , -1 , -2 , -3 , -4 , -5 , -6 , -7 , -8 , -9 , -10 } értékhalmazból választhatja ki.

  Tehát ezt a menüpontot választva nagyon sok „ismert” függvény vizsgálatára van lehetőség.

  Néhány példa:

  - x^0, vagyis az: y=1 konstans függvény;
  - x^1, vagyis az: y=x lineáris függvény;
  - $y=x^2$, másodfokú függvény;
  - $y=x^3$, harmadfokú függvény;
  - $y=x^n$, n-ed fokú függvény, ha n - pozitív egész szám;
  - x^{-1}, vagyis az: $y = \frac{1}{x}$ függvény;
  - x^{-2}, vagyis az: $y = \frac{1}{x^2}$ függvény;
  - x^n, vagyis az: $y = \frac{1}{x^n}$ függvény, ha n - negatív egész szám.

- *Abszolútérték függvény* almenü:

  Ezt a menüpontot választva, $y=|x|$, azaz az abszolútérték függvény tulajdonságait vizsgálhatjuk, ahol: *x* - tetszőleges valós szám lehet.

Természetesen a programban minden függvény esetében, behatárolt értelmezési tartománnyal fogunk dolgozni.

2.2.3. A **Felületek** menü:

A *Felületek* menüre klikkelve az ábrán látható testek közül választhatunk. Hasonlóan a függvényekhez, tetszőleges almenüt kiválasztva a program itt is a főablakról, a kiválasztott almenühöz tartozó ablakra vált. Ezekben az ablakokban a felhasználó kapcsolatba léphet a programmal, adatokat vihet be és utasításokat adhat az oktatóprogramnak.

Mivel a gömb és a tórusz a többi testtől eltérő típusú adatokat követel, ezért ezek külön-külön önálló panelen lesznek megjelenítve. Az összes többi poliédernél azonos típusú adatokat, nevezetesen az oldal hosszát kell megadni. Ezek a testek a további *alapadatokban* sem különböznek egymástól, ezért ugyanazon a panelen lesznek ábrázolva.

Azt, hogy éppen melyik testtel dolgozunk, hasonlóan a függvényekhez, itt is az ablak bal felső sarkában, a menüsor alatt lévő címke jelzi. Az ábrán látható, hogy az aktuális modul a *dodekaéder rajzoló modul*.

A felületek menüponthoz tehát, az eddigieken kívül, további három grafikus felületet kellett felépíteni. Szerkezetüket a **3. 4. és 5. ábra** mutatja.

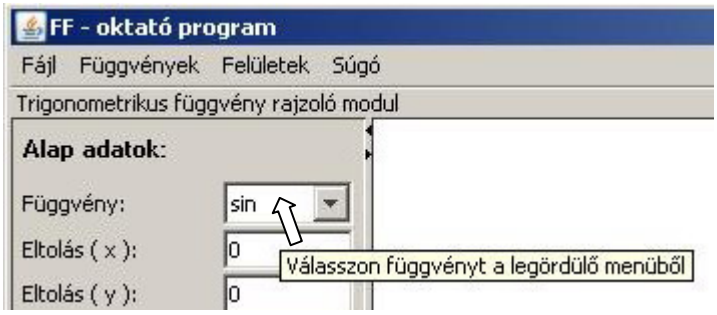
A program a bemenő adatok birtokában automatikusan kiszámítja ezen testek felszínét és térfogatát, a megfelelő képletek mögött megjelenítve azokat. A képleteket képek formájában hoztam létre, tehát a kocka, a tetraéder, az oktaéder, az ikozaéder és a dodekaéder közötti váltás során csak a címke, illetve a képleteket megvalósító képek változnak, a felület többi része változatlan marad.

2.2.4. A **Súgó** menü:

A *Súgó* menü egyetlen almenüje a *Segítség*. A program használata közben felmerülő kérdésekre reményeim szerint minden felhasználó válaszra talál a súgóban.

A segítség menüpontot választva természetesen egy újabb grafikus felület áll a rendelkezésünkre. Ez az ablak rendelkezik egy újabb menüsorral, mely két menütpontot, nyomógombot tartalmaz. A *FüggvénySúgó* nyomógomb a függvények használata közben, míg a *FelületSúgó* nyomógomb a felületek használata közben felmerülő kérdésekre ad választ. Az ablak szerkezetét a **7. ábra** mutatja.

A súgó mellett további segítséget jelent az egér megfelelő pozícióba való vonszolása. Ugyanis, ha egyes beviteli, illetve vezérlő komponensek fölé mozgatjuk az egeret, akkor az adott komponensről különböző információkat kaphatunk. Ez lehet például egy legördülő lista elemeiből való választás (lásd az ábrát), vagy egy adat, vagy eredmény mértékegységére vonatkozó információ, egy vezérlő komponens hatását leíró üzenet, illetve egy komponens által megjelenített adat további információkkal való kibővítése.



3. HASZNÁLATI ÚTMUTATÓ

3.1. HARDVERIGÉNYEK

A program működéséhez szükséges minimális erőforrás igény:

- 733 MHz-es processzor;
- 256MB RAM memória;
- 32 MB memóriával rendelkező videó kártya.

A fenti értékeket tesztelések alkalmával sikerült megállapítanom. Természetesen az egyes függvények vizsgálatához jóval kisebb erőforrásigény is megfelelne. Azonban a testek animálása, automatikus és kézzel való forgatása villámgyors számítást igényel.

Egy minimális erőforrásokkal rendelkező számítógépen az oktatóprogram természetesen már működőképes, de nagyobb felbontású és gyorsabb forgásokkal járó animációk esetében a mozgás enyhén „darabossá” válik. Tehát az optimális működéshez, mindenképpen nagyobb processzor sebesség, nagyobb kapacitású memória és több videó memória ajánlott.

Az oktatóprogram használatához 17”-os monitor, 1024 x 768-as felbontás mellett ajánlott, de természetesen ettől eltérő méretű és felbontású monitoron is használható.

3.2. PROGRAMKÖRNYEZET

A Java platform független nyelv, mely interpretált mivoltának köszönhető. Egy Java programot, így az általam írt oktatóprogramot is nem közvetlenül az operációs rendszer, hanem mindig valamilyen Java interpreter program, a Java Virtuális Gép hajtja végre. Az alkalmazás végrehajtása történhet például a Java Development Kit (JDK), azaz a Java fejlesztői környezet segítségével. Ez tartalmazza a Java programok fejlesztéséhez szükséges alapvető eszközöket, mint például a fordítót, a futtatót vagy alapvető osztályokat stb.

A program működtetéséhez szüksége lesz a Java Runtime Environmentre (JRE), azaz a Java futtató környezetre, ezt többek között a JDK is tartalmazza. A legtöbb számítógépen természetesen a JRE már telepítve van, hiszen előfordulhat, hogy a felhasználó más Java alkalmazásokat is futtat a számítógépén, de a mellékelt CD-n is megtalálható, illetve az internetről is letölthető.

3.3. TELEPÍTÉS

A mellékelt CD tartalma:

- *OktatoProgi* mappa, melynek tartalma: az oktatóprogram futtatásához szükséges állományok.
- JRE mappa, melynek tartalma: Java Runtime Environment (JRE), Windows operációs rendszer alá.

1. lépés: JRE telepítése

Amennyiben korábban még nem telepítette számítógépére a *Java futtató környezetet*, úgy ezt most, a program működése érdekében tegye meg.

2. lépés: OktatoProgi mappa másolása

Az *OktatoProgi* mappát teljes tartalmával együtt másolja a C:\Program Files\ könyvtárba, vagy egy tetszőleges helyre.

3. lépés: Parancsikon készítése:

A parancsikon létrehozásához az asztalon jobb egérgombbal klikkelve, a felbukkanó helyi menüből válassza az Új-menü Parancsikon-menüpontját. A programra mutató parancsikon helyét, az OktatoProgi\dist mappában találjuk. Itt válasszuk az OktatoProgi fájlt, ez egy jar kiterjesztésű végrehajtható állományt. Ezután, az utasításoknak megfelelően járjon el.

(A jar egy tömörített fájl formátum, amely tartalmazza a .class végrehajtható bájtkód fájlokat, a súgó betöltéséhez szükséges állományokat, illetve a felszín és térfogat képleteket reprezentáló képfájlokat).

3.4. A PROGRAM INDÍTÁSA

A létrejött parancsikon segítségével a program indítható, használatba vehető.

A programot elindíthatjuk a Start menü, programok menüpontjából is, az OktatoProgi-t választva. (Ha korábban azt ott elhelyeztük).

3.5. A PROGRAM BEZÁRÁSA

A Fájl menüből válassza a Kilépés menüpontot.

3.6. A FÜGGVÉNYEKHEZ TARTOZÓ PANEL HASZNÁLATA

Mint azt már korábban említettem, valamennyi függvényhez, ugyanazon grafikus felület van hozzárendelve. A alábbiakban ennek a felületnek a használatát, fogom ismertetni. Az ablak felépítését, szerkezetét a **2. ábra s**zemlélteti.

bal panel

A felület, menüsor alatti része egy *osztott panelből* áll. Ennek bal oldalán keresztül, (továbbiakban *bal panel,* vagy *párbeszéd panel*) vehetjük fel a kapcsolatot a programmal, ez biztosítja a kommunikációt az alkalmazás és használója között.

Míg a *jobb oldali panel*, (továbbiakban *jobb panel,* vagy *rajz panel*) a rajzi környezet összeállítására, azaz a függvények megjelenítésére alkalmas.

A bal és jobb panel közötti elválasztóval dinamikusan mozgatható, megváltoztatva ezzel a panelek felosztási arányait. Az elosztóvonal egyetlen kattintással is eltolható a minimális, illetve a maximális pozíciójába. Ezt a lehetőséget akkor célszerű alkalmazni, ha az ábrázolandó függvényt, teljes képernyőn szeretnénk megjeleníteni. Ez az opció a testek ábrázolására használt osztott panelnél is elérhető.

Az ábrán látható adatok alapértelmezett adatok. Bármely függvényt kiválasztva, rögtön megtekinthető annak képe a program által felkínált legbővebb ábrázolási tartományon és a felkínált *alap adatok* mellett.

Beviteli komponensek:

A beviteli komponensek segítségével tudunk a programnak adatokat átadni. Beviteli komponensek közé tartoznak a *legördíthető listák*, illetve a *szövegmezők*.

Vezérlő komponensek:

A vezérlő komponensek segítségével a program futását lehet vezérelni, ez lehet például egy művelet elvégeztetése. A program vezérlő komponensei a *nyomógombok*.

Megjelenítő komponensek:

A megjelenítő komponensek valamilyen információt jelenítenek meg a felhasználó felé. A programban használt megjelenítő komponensek általában a címkék, de mind a beviteli, mind a vezérlő komponensek egyben megjelenítő komponensek is lehetnek.

Az egyes beviteli, vagy vezérlő komponensek megnevezése többnyire címkével történik, mely név általában az adott komponens funkciójára utal.

3.6.1. Bal panel:

3.6.1.1. Alap adatok:

Legördíthető lista:

1. Függvény: 

A *Függvény:* címkével ellátott legördülő lista segítségével választhatunk a rendelkezésre álló függvényekből. Az, hogy a lista milyen elemeket, függvény neveket tartalmaz, attól függ, hogy a *Függvények* menüből mely menüpontot választottuk.

- A legördülő lista a s*zögfüggvények* esetében a *sin, cos, tg* és *ctg* elemeket, függvényeket tartalmazza.
- Az *exponenciális, a logaritmus és az abszolút érték függvények* esetében a lista egyetlen elemből áll, mely a kiválasztott menüponthoz tartozó függvényt reprezentálja, (*exp, log, abs*).

2. Hatvány kitevő: 

- A *hatvány függvények* esetében, a legördíthető lista nem a függvény nevét, hanem a kiválasztható hatványkitevőt tartalmazza. A lista elemek, a [10 , -10] intervallumból kiválasztható egész számok

A *Szögfüggvények, az Exponenciális függvény, a Logaritmus függvény, a Hatvány függvények és az Abszolútérték függvény* almenük tartalmáról részletesebben a **4. - 6. oldalon** olvashat.

Szövegmezők:

1. Eltolás:

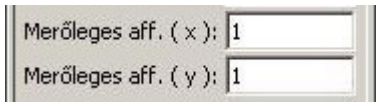
Lehetőség van a függvények x, illetve y - koordinátatengely mentén történő eltolására. A 0-érték természetesen azt jelent, hogy a függvényt nem toljuk el. Ez az alapértelmezett érték. Az eltolás egységei a következők:

Az *x - koordinátatengely* mentén két beosztás közötti távolság, π. A szövegmezőbe beírt 1-egység pedig megfelel $\frac{két\ \ beosztás\ \ közötti\ \ távolság}{\pi}$ - nek, azaz 3.14 ($\approx \pi$) egység egyenlő egy beosztással.

Az *y - koordinátatengely* mentén két beosztás közötti távolság, egy. A szövegmezőbe beírt 1-egység pedig megfelel két beosztás közötti távolságnak, azaz 1 egység egyenlő egy beosztással.

A beosztások szemléltetését, és megértését a **15. oldalon** található ábra segíti.

2. Merőleges affinitás:



A merőleges affinitás az x, illetve y - koordinátatengelyek mentén történő nyújtás, vagy zsugorítás mérőszáma.

Az 1 - alapértelmezett érték, a függvényre semmilyen hatást nem gyakorol, azaz a függvényt helyben hagyja. A 0 - értékkel való transzformálás az y=0 konstans függvényhez vezet.

A $0 < x < 1$ érték, a függvény x - koordinátatengely mentén történő nyújtásával, azaz a periódus megnövekedésével jár, míg a $1 < x$ érték hatására a függvény zsugorodik, azaz periódusa csökken.

A $0 < y < 1$ érték, a függvény y - koordinátatengely mentén történő zsugorításával, összenyomásával jár, míg a $1 < y$ érték hatására a függvény megnyúlik az y - koordinátatengely irányába.

Amennyiben az x, vagy az y értékek negatívak, úgy az eddigiek mellet a transzformáció a függvény tükrözésével is jár.

Nyomógomb:

Rajzol: 

A *Rajzol* nyomógombhoz hozzárendelt esemény, magának a kiválasztott és az *alap adatok* szerint felparaméterezett függvénynek, a megjelenítését jelenti. Ezen gombnyomás hatására adódnak át az argumentumok a programnak, melyek a számítás alapjait fogják képezni.

A gombnyomás által kiváltott esemény hatása az osztott panel jobb oldalán fog jelentkezni, hiszen ahogy említettem, a jobb panel szolgál a grafikus ábrák, azaz a függvények szemléltetésére.

3.6.1.2. Értelmezési tartomány:

Legördíthető listák:

Az ábrán látható legördíthető listák segítségével adott függvény értelmezési tartományának alsó, illetve felső határait tudjuk beállítani. Az alapértelmezett értékek a teljes értelmezési tartományt felölelik.

A logaritmus függvényt leszámítva, minden függvény értelmezési tartománya $[-5\pi, 5\pi]$, (az ábrán: [-5*PI, 5*PI]) intervallumok között, 0.5π lépésközök hozzáadásával, illetve kivonásával szűkíthető. Azaz a $[-5\pi, 5\pi]$ intervallumnál bővebb értelmezési tartományon a függvények ábrázolására nincs lehetőség. A Felső határ legördülő lista elemeit, az ábra részben szemlélteti.

A logaritmus függvény értelmezési tartománya a $[0, 5\pi]$, intervallumok között, 0.5π lépésközök hozzáadásával, illetve kivonásával szűkíthető.

3.6.1.3. Az ábrázolandó függvény:

Megjelenítő komponens: y = sin(x)

A megjelenítő komponens vagy címke alaphelyzetben, az *alap adatoknak* megfelelő függvényt mutatja, mely jelen esetben, az: y = sin(x) - függvény. Ha a párbeszéd panelen a F*üggvény:* címkével megnevezett legördíthető listából, például a cos - t választjuk, akkor a komponens tartalma: y = cos(x) - re módosul.

A komponens által megjelenített karaktersorozat tehát, dinamikusan nyomon követi a felhasználó ténykedéseit. Vagyis, egy függvény paraméterein végrehajtott bármilyen változtatás megjelenik a komponensen.

Ahhoz, hogy a komponens, az újonnan beállított adatoknak megfelelő függvényt reprezentálja, a változtatások után frissítenünk kell a programot. A frissítést, a *Rajzol* nyomógomb megnyomásával tehetjük meg.

Frissítés után a megjelenítő komponens szemlélteti, az aktuálisan felparaméterezett függvényt, a függvény grafikonja pedig megjelenik a rajz panelen. A *Rajzol* nyomógomb tehát egyszerre két eseményt is vezérel.

A megjelenítő komponens nagyon fontos információt hordoz a felhasználó számára. Hiszen az x vagy y - tengely mentén történő eltolás, illetve a merőleges affinitások beállítása után az ábrázolandó függvény felírása nem feltétlenül egyszerű. Főleg akkor nem, ha a felhasználó, az éppen vizsgált függvénnyel korábban még nem találkozott.

Továbbá a megjelenítő komponens egy megadott függvény paramétereinek beállításában is segítséget nyújthat. Ha a felhasználó nem tudja eldönteni, hogy például a sin(2x) - függvény ábrázolásához, mely paramétert kell módosítani, mindaddig próbálkozhat a paraméterek változtatásával, míg a komponensen az adott függvény meg nem jelenik.

A **16. - 17. oldalon** lévő példa szemlélteti minden egyes frissítés után, a megjelenítő komponens tartalmát.

3.6.2. Jobb panel:

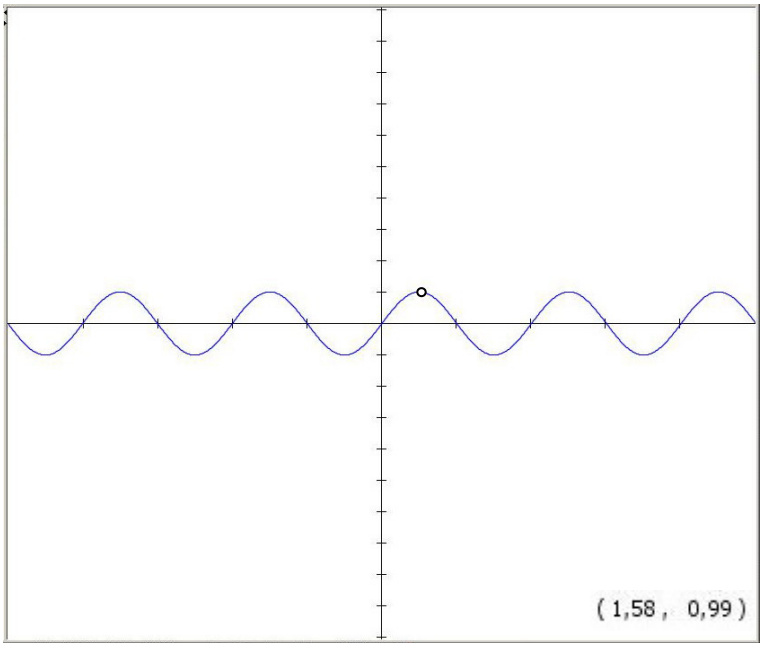
Alaphelyzetben a jobb oldali panelen semmilyen ábra nem látható. A jobb panelt, az osztott panel elválasztó vonalától jobbra lévő fehér alapszínű terület testesíti meg.

Ahogy említettem a jobb panel és a bal panel vezérlő komponense szoros kapcsolatban vannak. A felhasználó minden gombnyomására, az éppen aktuális paraméterek szerint frissül a rajzterület.

3.6.2.1. A koordináta rendszer:

A rajz panelhez hozzá van rendelve egy kétdimenziós koordinátarendszer, melynek origója az x, illetve y - koordinátatengelyek metszéspontjában van. Ha egy függvény megjelenik a rajzterületen, úgy egy adott függvénypont pontos (x, y) - koordinátáit megtudhatjuk, ha az egérrel rámutatunk a megfelelő pontra. A pont koordináták a rajz panel jobb alsó sarkában zárójelek között kaptak helyet. Természetesen az x, y - koordináták által meghatározott sík tetszőleges pontjának - azaz a függvények képétől különböző pontjának - koordinátái is megjelenítésre kerülnek. Az egér pozíció csak akkor kerül megjelenítésre, ha az egér a jobb panel felett helyezkedik el.

Az ábrán szemügyre vehetjük a kétdimenziós koordinátarendszert, az origó helyét, valamint egy aktuális pont koordinátáit.



Továbbá összevethetjük a beosztásokat az *eltolásnál* (**lásd 12. oldal**) elmondottakkal.

Természetesen, az egér pozícióhoz tartozó (x, y) koordináta értékek, összhangban vannak az eltolásnál ismertetett egységekkel.

A függvényekhez tartozó panel használatát egy olyan komplex példán keresztül mutatom be, mely a korábban megismertetett összes komponens, paraméterezési és beállítási lehetőségét magában foglalja, szemlélteti.

3.6.2.2. **Példa:** Ábrázolja az: $y = -2 \cdot \sin\left[2 \cdot \left(x + \frac{\pi}{4}\right)\right] + 2$ függvényt, a $[-4 \cdot \pi \ , \ 3 \cdot \pi]$ intervallumon.

Mielőtt elkezdenénk a feladatot, ellenőrizzük le, hogy a bal panelen a *Függvény:* címkével ellátott legördülő listában a sin - függvény van e kiválasztva. Mivel ez az első lista elem ezért erre csak akkor van szükség, ha ezt korábban nem állítottuk el.

A feladat megoldást célszerű több lépésre bontani:

- Első lépésben ábrázoljuk magát, az $y = \sin(x)$ - függvényt.

  Ekkor a *megjelenítő komponens* tartalma: y = sin(x)

  Ennél a lépésnél, az alapértelmezett beállítások szerint tekinthetjük meg a függvény képét, azaz semmilyen argumentumot nem kell elállítani.

- Második lépésben transzformáljuk az x - változót. Azaz, az x - koordinátatengely mentén történő változásokat szemléltetjük. Ezt a lépést bontsuk két részre:

  1. Ábrázoljuk, az $y = \sin(2x)$ - függvényt.

     Ekkor a *megjelenítő komponens* tartalma: y = sin(2(x))

     Azaz ennél a résznél a bal panel, *Merőleges aff. (x):* címkével ellátott szövegmezőbe 2 - t írunk.

  2. Ábrázoljuk, az $y = \sin\left[2 \cdot \left(x + \frac{\pi}{4}\right)\right]$ - függvényt.

     Ekkor a *megjelenítő komponens* tartalma: y = sin(2(x+0.785))

     Ekkor az *Eltolás (x):* címkével ellátott szövegmezőbe, a $\frac{\pi}{4} \approx 0.785$ értéket írjuk.

- A harmadik lépésben, a függvény értékét manipuláljuk, azaz érték transzformációt hajtunk végre. Vagyis az y - koordinátatengely mentén történő változásokat szemléltetjük. Ezt a lépést bontsuk három részre:

  2. Ábrázoljuk, az $y = -\sin\left[2 \cdot \left(x + \frac{\pi}{4}\right)\right]$ - függvényt.

Ekkor a *megjelenítő komponens* tartalma: y = -1sin(2(x+0.785))

Azaz következő lépésként, a *Merőleges aff. (y):* címkével ellátott szövegmezőbe írjunk -1 - t.

2. Ábrázoljuk, az $y = -2 \cdot \sin\left[2 \cdot \left(x + \frac{\pi}{4}\right)\right]$ - függvényt.

Ekkor a *megjelenítő komponens* tartalma: y = -2sin(2(x+0.785))

Ennél a pontnál, szintén a *Merőleges aff. (y):* címkével ellátott szövegmezőt szerkesszük, de most az értéke -2 legyen.

2. Ábrázoljuk, az $y = -2 \cdot \sin\left[2 \cdot \left(x + \frac{\pi}{4}\right)\right] + 2$ - függvényt.

Ekkor a *megjelenítő komponens* tartalma: y = -2sin(2(x+0.785))+2

Vagyis, az *Eltolás (y):* címkével ellátott szövegmezőbe írjunk 2 - t.

- Utolsó lépésként, az értelmezési tartomány résznél, a legördülő listák segítségével állítsuk be a feladat szövegében szereplő intervallumnak megfelelő alsó és felső határokat.

Célszerű minden lépés után, a *Rajzol* nyomógomb segítségével frissíteni a függvény grafikonját. Ezt azért ajánlom, mert ekkor minden egyes beállított paraméter érték esetén, figyelemmel kísérhetjük az adott argumentumnak a függvényre gyakorolt hatását, illetve a *megjelenítő komponens* frissítésére csak így van lehetőségünk.

Ha a bal oldali párbeszéd panelen, minden adatot jól állított be, akkor a rajz panelen a **6. ábra** által szemléltetett függvény grafikonjának kell megjelenni.

3.7. A FELÜLETEKHEZ TARTOZÓ PANELEK HASZNÁLATA

A **6. oldalon** az egyes menük ismertetésénél említettem, hogy a felületek vizsgálatához három önálló panelt kellett létrehozni. Egyet a *gömb*, egyet a *tórusz*, egyet pedig az összes többi *poliéder* vonatkozásában.

A gömbhöz tartozó bal panel

Az ábra a gömb elemzését segítő felület bal oldalát, azaz a *bal panelt* szemlélteti.

Mivel a gömb, a tórusz és a többi test elemzéséhez szükséges panelek csak az *alap adatok* résznél különböznek, ezért a *gömb animálása* vagyis az animációs rész, továbbá *a felszín és térfogat számítási* rész, a három panel esetében együtt lesz tárgyalva.

A felület **(3. ábra)** menüsor alatti része egy *osztott panelből* áll. A panel szerkezete teljesen megegyezik a függvényekhez tartozó osztott panel szerkezetével. Azaz, a dinamikusan változtatható elosztóvonal bal oldalán a *bal*, míg jobb oldalán a *jobb panel* vagy *rajz panel* helyezkedik el, a korábban elmondott funkcionalitásokat megtartva. Vagyis a bal panel a program és a felhasználó közötti kapcsolatot és információközlést, míg a jobb panel a szemléltetést segíti.

Azonban, a bal panel felépítése a függvényeknél megismertekhez képest megváltozott. Ez annak a következménye, hogy a testek vonatkozásában más típusú adatok és más jellegű vizsgálatok szükségesek. Természetesen a panel továbbra is tartalmaz alapértelmezett adatokat, segítve ezzel a felhasználó kezdeti lépéseit.

A jobb panel annyiban változott, hogy az egérmutató által kijelölt pont koordinátái nem lesznek megjelenítve, mert ez a vizsgálatok szempontjából felesleges lenne.

3.7.1. Bal panel:

3.7.1.1. GÖMB:

3.7.1.1.1. ***Alap adatok:***

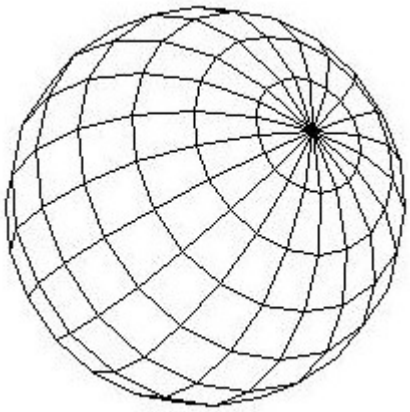
Legördíthető lista:

Felbontás: 

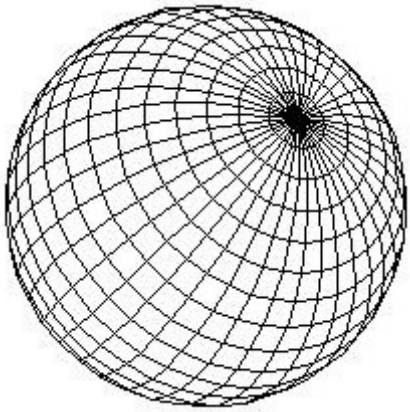
A *Felbontás:* címkével ellátott legördíthető lista elemeit választva, megváltoztathatjuk a gömb és a körgyűrű felületi hálójának sűrűségét. A háló sűrűségét a test méretéhez viszonyítva célszerű beállítani. Hiszen egy viszonylag kisméretű gömb, vagy körgyűrű esetében, kisebb felületi háló sűrűség mellett életszerűbb képet kaphatunk.

A felbontás két fokozatban állítható. Az alapértelmezett érték 20, emellett választható a 40-es érték, mely nagyobb felbontást sűrűbb felületi hálót eredményez. A 20-as érték azt jelenti, hogy a gömböt a z - tengely irányából nézve 20 szeletre osztjuk fel, és a z - tengely mentén az xy - síkban a szeleteket szintén felosztjuk 10 részre.

A felületháló fokozatainak szemléltetése:



1. A felületháló sűrűsége 20-as felbontás mellett.



2. A felületháló sűrűsége 40-es felbontás mellett.

Szövegmezők:

2. Eltolás: 

Lehetőség van a testek x, illetve y – koordinátatengely mentén történő eltolására. Ez gyakorlatilag azt a célt szolgálja, hogy a felparaméterezett testet a képernyő közepére, vagy annak tetszőleges helyére mozgathassuk. Mivel a koordinátarendszer origója a rajz panel bal felső sarkában van, ezért az alapértelmezett értékek úgy vannak beállítva, hogy az adott test teljes egészében a rajzterületen helyezkedjen el. Hiszen, az x=0 és y=0 értékek mellett, például a gömbnek, csak a jobb alsó negyede lenne látható.

Az eltolás egységei:

Mind az x, mind az y - koordinátatengely mentén történő eltolás esetén 1 - egység, 1 - pixelnek felel meg. Azaz, az alapértelmezettként beállított értékek a gömb 250 - pixellel való eltolását jelentik, mindkét koordinátatengely pozitív irányában.

2. Sugár: 

A *Sugár (r):* címkével ellátott szövegmezőben adhatjuk meg a gömb sugarát. Az ábrán a szövegmező az alapértelmezett értéket tartalmazza. Egy-egység itt is, mint ahogy az eltolásnál, egy pixelnek felel meg.

Nyomógomb:

Rajzol: 

A *Rajzol* vezérlő komponenst megnyomva az *alap adatoknak* megfelelő, aktuálisan vizsgált test, huzalvázas képe jelenik meg a rajz területen.

A nyomógombról, további információkat a **13. oldalon** olvashat.

3.7.1.2. TÓRUSZ:

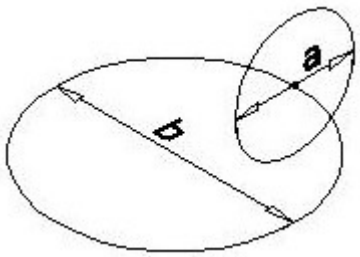
3.7.1.2.1. ***Alap adatok:***

Az ábrát összehasonlítva a gömb bal paneljén található *alap adatok* résszel **(18. oldal)**, láthatjuk, hogy az első három adatban teljesen megegyeznek.

Ezért a *Felbontás:* címkével ellátott legördíthető listára, illetve az E*ltolás (x): és Eltolás (y):* címkével

megnevezett szövegmezőkre a gömbnél elmondottak érvényesek. Továbbá a *Rajzol* nyomógomb vonatkozásában sincs semmilyen új információ.

A körgyűrű felület származtatása:

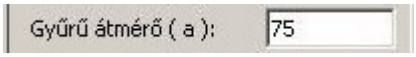


A körgyűrű felület származtatható úgy, hogy egy b - átmérőjű körvonalon, (nevezzük vezérgörbének, vagy *vezérkörnek*) körbeforgatunk, egy a - átmérőjű kört, (nevezzük generáló - görbének, vagy *generáló - körnek*) úgy, hogy a generáló - kör középpontja a forgatás alatt mindvégig a vezérkörön helyezkedik el. Továbbá a generáló - kör szimmetria tengelye mindvégig érinti a vezérkört. Ahol: $a \leq b$.

Szövegmezők:

1. Belső átmérő: 

   A *Belső átmérő (b):* címkével ellátott szövegmező, a *vezérkör* átmérőjét adja meg.

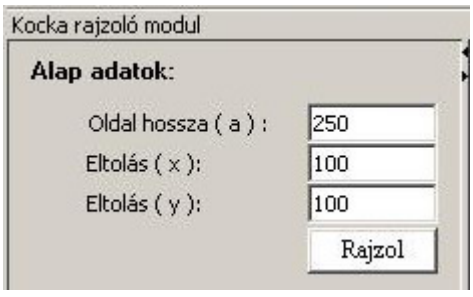
2. Gyűrű átmérő: 

   A *Gyűrű átmérő (a):* címkével jelölt szövegmező, a *generáló - kör* átmérőjét adja meg.

Mind a *belső átmérő*, mind a *gyűrű átmérő* esetében 1 - egység a szokásos módon, 1 - pixelnek felel meg.

3.7.1.3. A TÖBBI POLIÉDER:

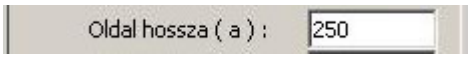
3.7.1.3.1. ***Alap adatok:***



A kocka, a tetraéder, az oktaéder, az ikozaéder és a dodekaéderhez tartozó közös panel *alap adatok* része kívánja meg a legkevesebb argumentumot a felhasználótól. A program által várt egyetlen újadat, az

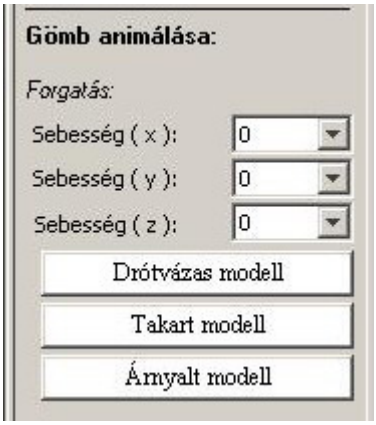
egyes poliéderek oldalának hossza, vagyis az *Oldal hossza (a):* címkével jelölt *szövegmező*. Az *eltolás* és a *rajzol* komponensek szerepe, itt is megegyeznek a gömbnél elmondottakkal.

Szövegmezők:

Oldal hossza: 

Az *Oldal hossza (a):* címkével megnevezett szövegmező segítségével, az aktuális poliéder oldalának hosszát paraméterezhetjük fel, pixelekben megadva.

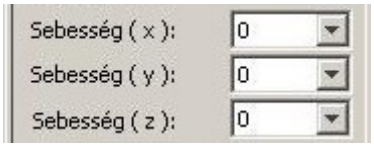
3.7.1.3.2. ***A gömb animálása:*** (Az itt leírtak az összes többi testre is érvényesek.)



Az animáció lényegében az egyes testek forgását, keringését fogja jelenteni. Erre utal, a *Forgatás:* megjelenítő komponens. Ezáltal a program lehetőséget biztosít a testek „körbejárására”, 3 - dimenziós térbeli képének szemügyre vételére.

Az animált kép különféle megjelenítési stílusokban tárulhat elénk. Bármely kiválasztott test esetén a hozzá tartozó ablakban megtekinthetjük annak drótvázas, takart, illetve árnyalt modelljét, melyet az erre szolgáló vezérlő komponensek *Drótvázas modell*, *Takart modell*, *Árnyalt modell* igénybevételével tudunk szabályozni. Az animáció mellett ebben a részben erről is szó lesz.

Legördíthető lista:

Sebesség: 

A sebesség beállításával lehetőségünk van a testek folyamatos forgásának megtekintésére. A *Sebesség (x):* címkével megjelölt legördíthető lista segítségével tudjuk szabályozni, a test x - koordinátatengely körüli forgásának sebességét. Hasonlóan, az y és z - koordinátatengelyek körüli forgást, a megfelelő legördíthető listák segítségével vezérelhetjük.

A sebesség beállítására 5 - fokozat áll rendelkezésre. Az alapértelmezett, 0 - értékek mellett a test nem kering, mozdulatlan. Ezt akkor célszerű alkalmazni, ha a testet különféle

megjelenési stílusokban szeretnénk ábrázolni keringetés nélkül. Hiszen, a *Rajzol* nyomógomb megnyomásával, csak a drótvázas modell megtekintésére van lehetőségünk.

A test keringetése történhet az egyes koordinátatengelyek körül külön - külön, illetve egyszerre is. Ezzel lehetőségünk van a test bármilyen irányból való szemrevételezésére.

Az interaktív térbeli megtekintés:

A forgatás saját kezűleg az egérmutató közvetítésével, annak vonszolásával is megvalósítható. Ez az interaktív térbeli megtekintés, amikor a test nem folyamatosan kering, hanem a mutatóeszköz segítségével a felhasználó vezérli a forgás irányát és sebességét.

Az egeret használva a folyamatos keringetés a felhasználó beavatkozásával bármikor megszakítható. Továbbá, az egér segítségével a mozdulatlan test bármikor „életre kelthető”.

Nyomógombok:

Az animációs részben három nyomógomb található: a *Drótvázas modell*, a *Takart modell* és az *Árnyalt modell* nyomógombok.

Bármely vezérlő komponenst lenyomva, egyrészt, átadódnak a programnak az alap adatok részben megadott argumentumok. Tehát ebből a szempontból működésük megegyezik, a *Rajzol* nyomógombéval. Másrészt, az animációs részben beállított forgási sebességek is átkerülnek a programhoz. Az összes átadott argumentum fogja a grafikus ábra megjelenítéséhez szükséges számítások alapját képezni.

Továbbá bármely gombnyomás hatására az osztott panel jobb oldalán, azaz a rajz panelen megjelenik a grafikus ábra. Ebben a vonatkozásban is a Rajzol nyomógomb működését követi.

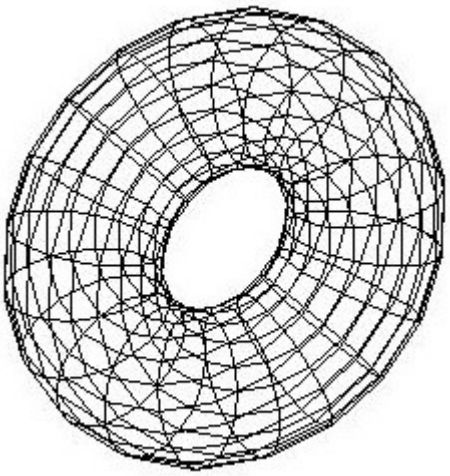
1. Drótvázas modell: 

A *Drótvázas modell,* vezérlő komponenst megnyomva, az aktuálisan vizsgált test, huzalvázas képe jelenik meg a rajz területen.

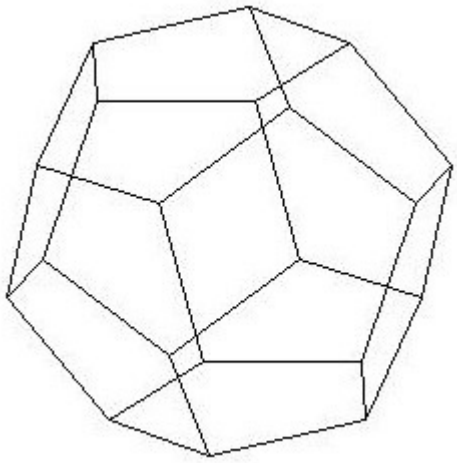
Huzalvázas modell, vagy drótvázas modell

Az objektumok huzalvázas vagy drótvázas megjelenítése a legegyszerűbb, ugyanakkor a legkevésbé valósághű. Ekkor a testeket éleikkel ábrázoljuk, azaz úgy jelenítjük meg azokat

mintha drótból készültek volna. Az ábrán nincsenek takart vonalak, minden él teljes egészében megjelenítésre kerül. A huzalvázas megjelenítés műveleti igénye nagyságrendileg kisebb a különböző árnyalt képeket előállító algoritmusokénál.



A tórusz drótvázas modellje



A dodekaéder drótvázas modellje

2. Takart modell: 

A *Takart modell,* vezérlő komponenst megnyomva, az aktuálisan vizsgált test „kitakart" képe jelenik meg a rajz területen.

Takart modell

A takart megjelenítés során nem az objektumok átlátszó vázát, hanem az élek által meghatározott határoló felületek „kitakart" képét rajzoljuk ki.

Vagyis ekkor a nézőpont és az objektum modelltérbeli elhelyezkedése alapján a kirajzolásnál a takarási viszonyokat is figyelembe vesszük. Ez azt jelenti, hogy a képen a látható lapok által eltakart élek és testrészek nem fognak megjelenni.

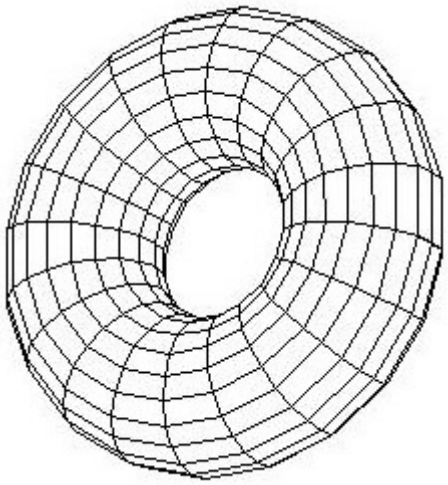
A takarási viszonyok kialakításának matematikai háttere:

A tóruszt leszámítva minden test konvex, azaz a test bármely két pontját összekötő szakasz az adott testen belül helyezkedik el.

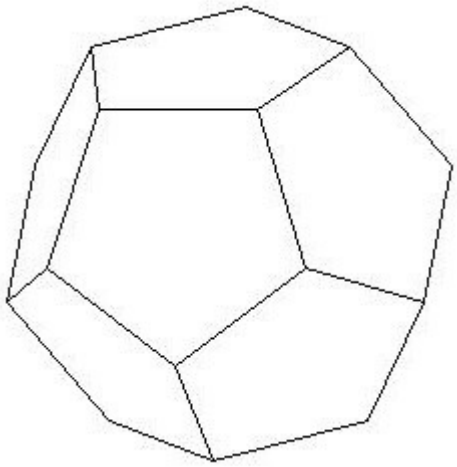
Az egyes testek zárt sokszögekből, vagy poligonokból épülnek fel. Ha ezeknek a poligonoknak a normálvektorait úgy állítjuk be, hogy az objektumból kifelé mutassanak, akkor azok a poligonok vagy lapok melyek normálvektorai nem a nézőpont felé mutatnak,

biztosan takarva lesznek a test közelebbi lapjai által. A konvex poliéderek esetében elegendő ezeknek az úgynevezett hátrafelé néző lapoknak az eltávolítása.

Mivel a tórusz konkáv, ezért ott a látható lapok is takarhatják egymást. Azonban, ha a látható lapokat legkisebb z - koordinátájuk szerint csökkenő sorrendbe rendezzük, akkor megjelenítésük után a takarási viszonyok megfelelőek lesznek.



A tórusz takart modellje



A dodekaéder takart modellje

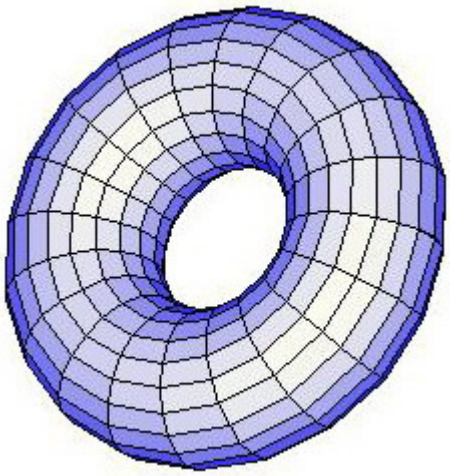
3. Árnyalt modell: 

Az *Árnyalt modell,* vezérlő komponenst megnyomva, az aktuálisan vizsgált test árnyalt képe jelenik meg a rajz területen.

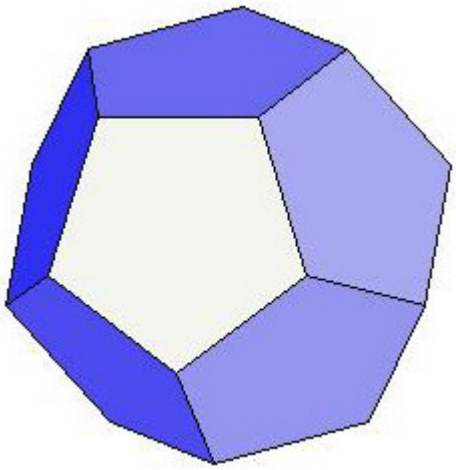
Árnyalt modell

A jelenetek sokkal realisztikusabb képi megjelenítését kapjuk, a testek felületének árnyalt ábrázolásával.

Az árnyalással azt próbáljuk kifejezni, hogy a természetben látható megvilágított felületek a fényforrások, a felület térbeli helyzetétől valamint a nézőpont elhelyezkedésétől függően különböző fényességűnek látszanak. A takarási viszonyokat természetesen ennél az ábrázolásnál is figyelembe vesszük.

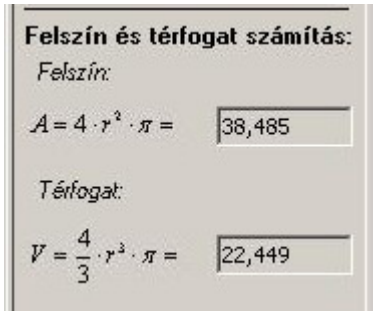


A tórusz árnyalt modellje



A dodekaéder árnyalt modellje

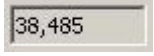
3.7.1.3.3. ***Felszín és térfogat számítás:***



A program, bizonyos bemenő adatok birtokában képes kiszámítani a testek felszínét és térfogatát, természetesen a megfelelő képletek mögött megjelenítve azokat. Ezzel lehetőségünk van egyfajta önellenőrzésre, mely vonatkozhat a képletek biztos tudására, illetve számítási eredmények összehasonlítására.

A számításokat a program automatikusan, a felhasználó beavatkozása nélkül végzi el. Kezdetben az alapértelmezett értékek kerülnek behelyettesítésre a képletbe, majd ezt követően már a felhasználó által megadott adatok. Egy újonnan bevitt adatot követően, a bal panelen tetszőleges komponensre klikkelve, frissülnek a számítási eredmények. (Az ábra, a gömb felszín és térfogat számítási részét szemlélteti).

A felszín és térfogat képletekben nyomon követhető, hogy az *alap adatok* részből mely adatok kerülnek behelyettesítésre. Az ábrán, a képletekben szereplő **r** - betű, az *alap adatok* részben a megfelelő címkében zárójelek között szerepel (lásd. *Sugár(r)*). Azaz, a *Sugár (* **r** *):* címke mögött álló szövegmező értéke lesz behelyettesítve a formulákba.

Természetesen, az eredményeket megjelenítő szövegmező csak információt közvetít a felhasználó felé, szerkesztésére nincs lehetőség. Ezért kapott szürke hátteret. 

Az *eredmény* legalább egy, és legfeljebb három tizedes jegy *pontosságú*.
A számítás alapjául szolgáló értékek, a felhasználásuk előtt 100-al osztva lettek. Ezt azért tartottam fontosnak, mert így az eredmények sokkal esztétikusabbak, olvashatóbbak.

Például a gömb esetében, ha az eredeti értékekkel számolnánk, úgy a térfogat eredménye: 22449297,5 egység^3 lenne. Ha a felhasználó, az alapértelmezettként megadott sugárnál nagyobb értéket ad meg, úgy az eredmény még tovább növekedhet.

Így, *a számítási eredmények mértékegysége* a felszín esetében: $(100 \cdot egység)^2$, míg a térfogat esetében: $(100 \cdot egység)^3$.

4. A „BOLOND BIZTOS” PROGRAM, AVAGY A HIBÁS BEMENŐ ADATOK KEZELÉSE

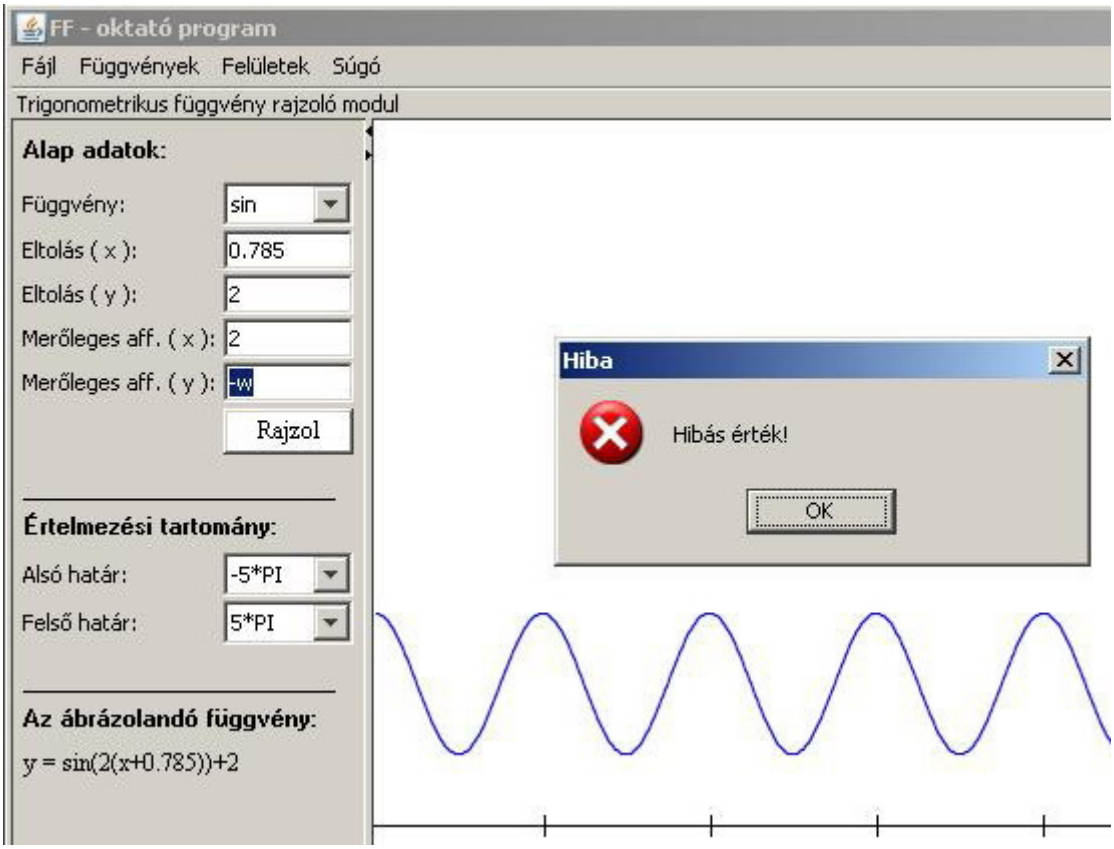
4.1. ELSŐ HIBA CSOPORT:

Amikor a felhasználó még csak ismerkedik a programmal, könnyen előfordulhat, hogy a program számára értelmezhetetlen adatokat ad át. Persze, a hibás input megadása, lehet akár egy egyszerű félre gépelés eredménye is, nem feltétlenül a programmal szembeni „tudatlanságunkból” fakadhat.

A fent leírt hibához vezethet, ha például egy szám helyet véletlenül egy betűt ütünk le, vagy tizedespont helyett, tizedes vesszőt írunk.

Ezekkel, az adatokkal, a program nem tud számolni, átadásuk hibás működéshez vezethet.

A hiba kezelése:



Az ábra a **16. oldalon** megfogalmazott feladatot szemlélteti. A feladat harmadik lépésének 2. pontjában a *Merőleges aff. (y):* címkével ellátott szövegmezőt kell

szerkesztenünk, ahogy az, az ábrán is látható. Az átadandó érték, -2 helyett „véletlenül” -w - t írtunk.

A *Rajzol* vezérlő komponenst megnyomva, egy úgynevezett hibakijelző dialógusdoboz figyelmezteti a felhasználót, a helytelenül megadott argumentumról. Az üzenet: *„Hibás érték!”*, a hiba jellegére utal. A dialógusdoboz felugrásával egy időben a hibás adatot tartalmazó szövegmező kapja meg a beviteli fókuszt, amelyet a program vizuálisan is megjelöl (**lásd, ábra**).

Az OK - gombot megnyomva, a felhasználó köteles a kijelölt mező tartalmát helyesen felparaméterezni, ellenkező esetben ismét felbukkan a hiba üzenet és az előző eseménysor mindaddig ismétlődik, amíg a program számára át nem adjuk a megfelelő adatot.

Természetesen ez a probléma a testek esetében is felmerülhet. A felhasználó által okozott hasonló hibák kezelése, és a dialógusdoboz által közvetített hibaüzenet, a testek esetében teljesen megegyezik az előbb elmondottakkal. A különbség az, hogy a függvényeknél a hiba vizsgálatot csak egy panel vonatkozásában kellett megvalósítani, míg a testeknél a gömb, a tórusz és a többi testhez tartozó közös panelre külön - külön meg kellett oldani.

A körgyűrű vonatkozásában további problémák is felmerülhetnek, melyről a második hiba csoport tartalmaz információkat.

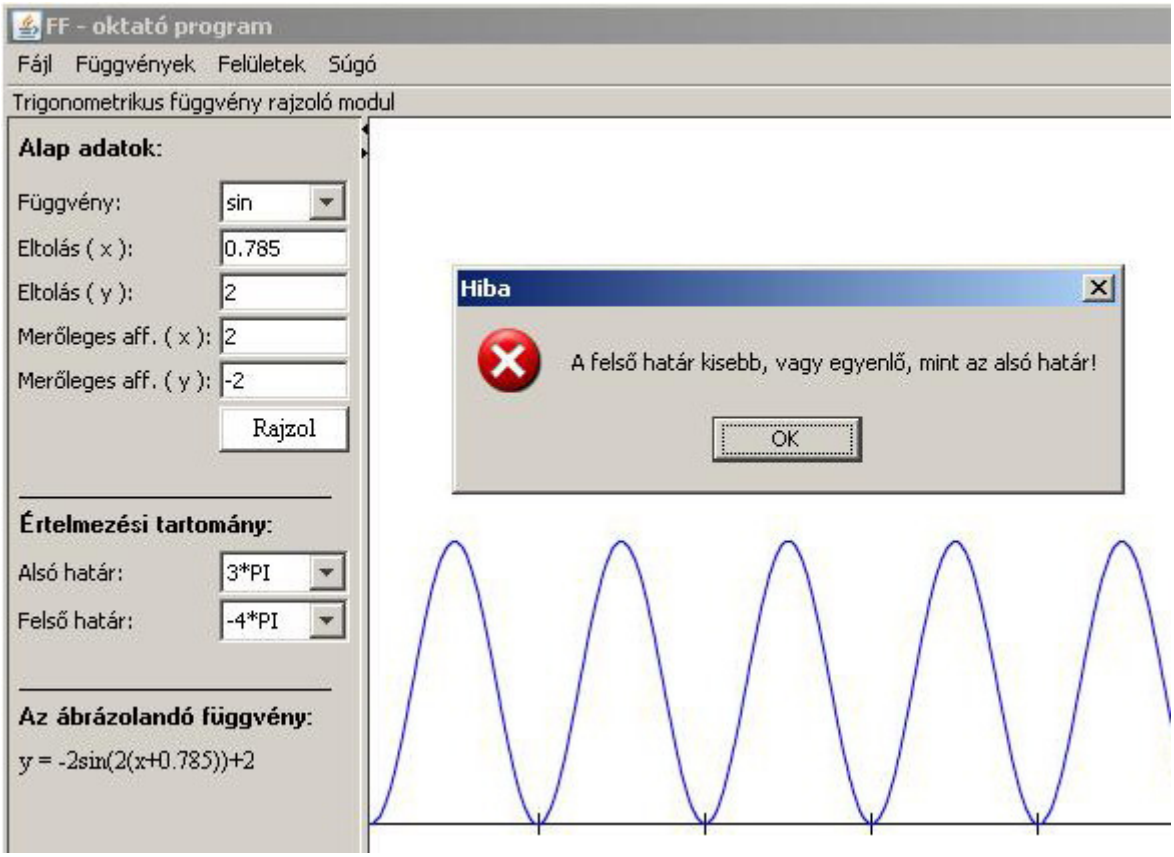
4.2. MÁSODIK HIBA CSOPORT:

Megalapozatlan matematikai tudásunk vagy figyelmetlenségünk folytán megadhatunk olyan argumentumokat, mellyel a program ugyan képes lenne műveleteket végrehajtani, de mindez csak idő és energiapazarlás, mert a számítás nem jár érzékelhető eredménnyel, vagy az eredmény a felhasználó megtévesztéséhez vezet.

Példa:

1. *Ilyen lehet például, egy rosszul beállított intervallum (pl. [* $3 \cdot \pi$, $-4 \cdot \pi$ *]).*
2. *Ha a tóruszhoz tartozó grafikus felületen, a Belső átmérő (b): és a Gyűrű átmérő(a): címkével jelölt szövegmezőket úgy paraméterezzük fel, hogy: a>b, akkor a test ezen argumentumok mellett ábrázolható, de a felszín és térfogatszámítás hibás eredményhez vezet. Vagyis a program rossz eredményeket közvetítene a felhasználó felé.* ***(lásd 31. oldal, ábra)****.*

1. A hiba kezelése (lásd 1. példa).



Az ábrán a **16. oldalon** megfogalmazott feladat szerint adtunk meg az *alap adatokat*. A feladatban utolsó lépésként az értelmezési tartományt kell beállítani, azonban az alsó és a felső határ fordítva lett megadva. Nyílván való, hogy ezen az intervallumon a függvény nem ábrázolható, amelyről a felhasználót is figyelmeztetni kell.

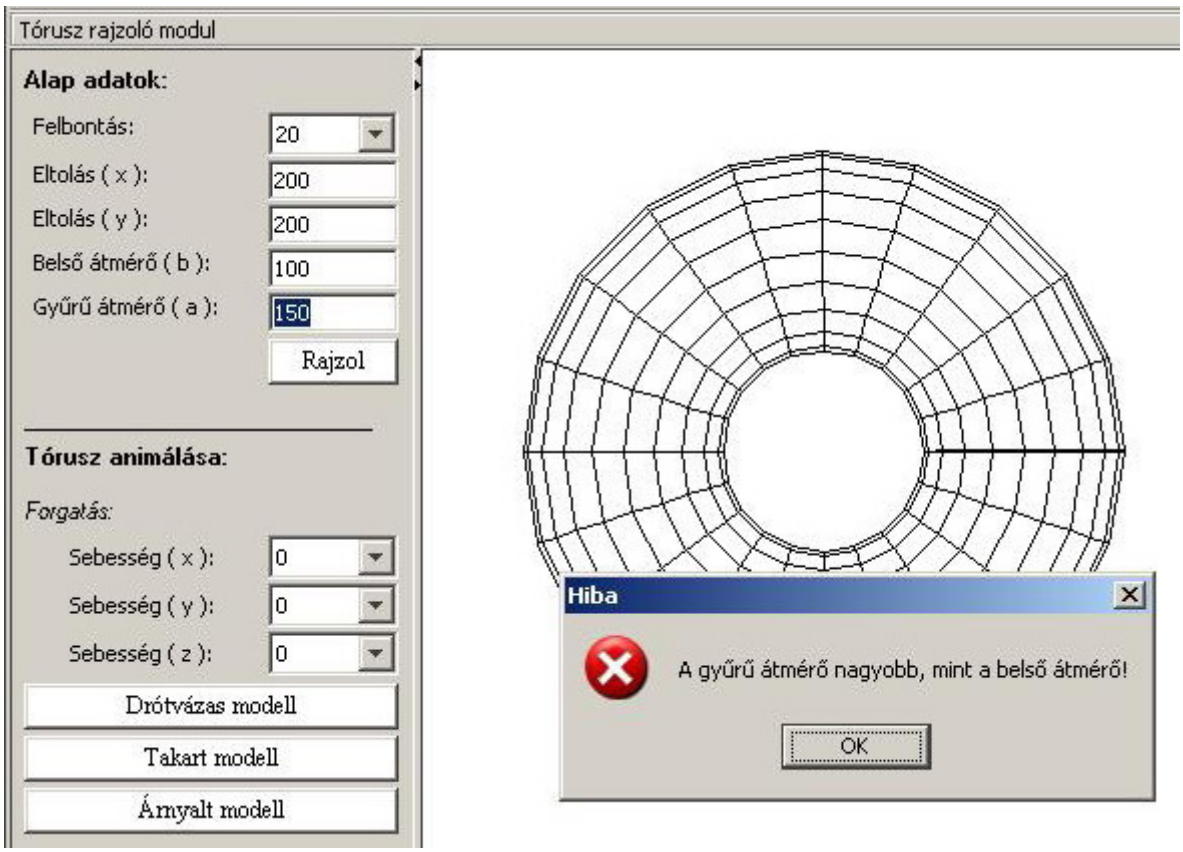
A *Rajzol* vezérlő komponenst megnyomva a felbukkanó dialógusdobozban *„A felső határ kisebb, vagy egyenlő, mint az alsó határ!”* üzenetet olvashatjuk.

Az OK - gomb megnyomása után a „hibás adatot” tartalmazó, legördíthető lista kapja meg a beviteli fókuszt. Ez esetben, ez a *Felső határ:* címkével ellátott lista lesz, mert ezt állítottam be utoljára. A program a hibásan beállított elemet, itt is vizuálisan kijelöli, de csak az OK- gomb lenyomása után nem pedig a felbukkanó dialógusablakkal együtt.

Amennyiben előbb a felső határt állítjuk be, az aktuális alsó határ értékéhez képest helyesen, majd ezután az alsó határt, de a már aktuálisan beállított felső határhoz képest hibásan, akkor *„Az alsó határ nagyobb, vagy egyenlő, mint a felső határ!”* üzenetet kapjuk.

Természetesen a fenti eseménysor itt is mindaddig ismétlődik, amíg az értelmezési tartományt helyesen meg nem adtuk, vagyis a feladat csak értelmesen megadott intervallum mellett folytatható.

2. A hiba kezelése: (lásd 2. példa).



Az ábrán látható, hogy a felhasználó a *gyűrű ármérőjét* a *belső átmérőnél* nagyobbra állította, azaz: (a>b). Ilyen argumentumok mellett a test ábrázolható lenne, azonban, a megjelenő grafikus kép nem körgyűrűt ábrázolna.

A nagyobb problémát mégis a felhasználó, hibás felszín és térfogat eredményekkel való megtévesztése jelentené. Ennek megelőzése érdekében a program megakadályozza, hogy az *a*, illetve a *b* paraméterek értéke között fennálljon az *a>b* összefüggés.

A *vezérlő komponensek bármelyikét* (*Rajzol, Drótvázas modell, Takart modell, Árnyalt modell*) megnyomva, a felbukkanó dialógusdoboz *„A gyűrű átmérő nagyobb, mint a belső átmérő!”* hiba üzenettel figyelmezteti a felhasználót, a helytelenül megadott argumentumról. A dialógusdoboz felugrásával egy időben, a hibás adatot tartalmazó szövegmező kapja meg a beviteli fókuszt, amelyet a program vizuálisan is megjelöl (**lásd, ábra**).

Az OK - gombot megnyomva, a felhasználó köteles a kijelölt mező tartalmát helyesen felparaméterezni, ellenkező esetben ismét felbukkan a hiba üzenet, és az előző eseménysor mindaddig ismétlődik, amíg a program számára át nem adjuk a megfelelő adatot.

Amennyiben előbb a gyűrű átmérőjét állítjuk be, az aktuális belső átmérőhöz képest helyesen, majd ezután a belső átmérőt, de a már aktuálisan beállított gyűrű átmérőhöz képest hibásan, akkor *„A belső átmérő kisebb, mint a gyűrű átmérő!”* üzenetet kapjuk.

5. A PROGRAM BŐVÍTHETŐSÉGE

A program bővítése egy függvénnyel vagy egy testtel az elkészített panelek felhasználásával egyszerűen és gyorsan megvalósítható. A függvények esetében az összes függvényhez tartozó, korábban már megismertetett panelt szerkeszthetjük, míg a testek esetében három panel közül is választhatunk attól függően, hogy az újonnan létrehozandó test milyen paramétereket kíván. A panelek szerkesztése az egyes komponensek, azaz a címkék, a szövegmezők vagy a nyomógombok átnevezését, illetve a legördíthető listák elemeinek megváltoztatását jelentheti.

Mivel a vezérlő komponensek minden függvény vagy test esetében ugyanazt a funkciót látják el, és a vizsgálatukhoz szükséges paraméterek is általában átfedik egymást, ezért a panelek szerkesztése például a függvények esetében, általában csak a *Függvény:* címkével megnevezett legördíthető lista elemeinek az átnevezését jelenti.

Az alábbi programrészlet, a függvényekhez tartozó panelen a legördíthető lista elemeinek a beállítását végzi attól függően, hogy a panelt mely függvény (vagy függvények) vonatkozásában kívánjuk használni. A kódrészletet a Fuggveny - osztály tartalmazza. (Ebben az osztályban építjük fel a panelt, illetve itt vizsgáljuk a felhasználó által átadott adatok helyességét).

```
If( fuggveny.equals( „Exp” ) )
   FuggvenyComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[]{„exp”}));
else if( fuggveny.equals( „Log” ) )
   FuggvenyComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[]{„log”}));
else if( fuggveny.equals( „Abs” ) )
   FuggvenyComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[]{„abs”}));
else if( fuggveny.equals( „Hatvany” ) )
   FuggvenyComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[]
{„10”,”9”,”8”,”7”,”6”,”5”,”4”,”3”,”2”,”1”,”0”,”-1”,”-2”,”-3”,”-4”,”-5”,”-6”,”-7”,”-8”,”-9”,”-10”}));
else if( fuggveny.equals( „Trigonometria” ) )
   FuggvenyComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[]
{„sin”,”cos”,”tg”,”ctg”}));
```

A programkód működése: Ha a String típusú függvény objektum az „Exp” sztringet reprezentálja, akkor a FuggvenyComboBox legördíthető lista egyetlen eleme az exp lesz, különben ha a „Log”

sztringet reprezentálja, akkor a legördíthető lista egyetlen eleme a log lesz, és így tovább. Egy újabb függvény felvétele esetén létre kell hoznunk egy **else if** ágat.

Például, ha a fuggveny objektum a „*Trigonometria*” sztringet reprezentálja, úgy a FuggvenyComboBox legördíthető lista elemei az alábbiak lesznek: (sin, cos, tg, ctg).

Azt, hogy mely modul van használatban az alábbi kódrészlet segítségével állíthatjuk be:

```
if( fuggveny.equals( „Exp” ) )
    modulLabel.setText( „ Exponenciális függvény rajzoló modul” );
else if( fuggveny.equals( „Log” ) )
    modulLabel.setText( „ Logaritmus függvény rajzoló modul” );
else if( fuggveny.equals( „Abs” ) )
    modulLabel.setText( „ Abszolútérték függvény rajzoló modul” );
else if( fuggveny.equals( „Hatvany” ) )
    modulLabel.setText( „ Hatvány függvény rajzoló modul” );
else if( fuggveny.equals( „Trigonometria” ) )
    modulLabel.setText(„ Trigonometrikus függvény rajzoló modul „);
```

A kódrészlet működése hasonló az előző oldalon lévő programkód működéséhez, azzal a különbséggel, hogy itt a megfelelő feltétel teljesülése esetén a modulLabel, azaz a modul címke által megjelenített szöveg, a kerek zárójelek között megadott sztring lesz.

Például, ha a fuggveny objektum az „*Exp*” sztringet reprezentálja, úgy a modulLabel komponens által megjelenített szöveg: 

Amennyiben végrehajtottuk a panelen a szükséges módosításokat, definiálnunk kell egy metódust, mely az adott függvény grafikonjának kirajzolását végzi. A metódust a FuggvenyClass osztályban kell létrehoznunk, melynek felépítése a következő:

```
public void függvényNév(  /* A szükséges paraméterek */ )          /* függvény specifikáció */
{
        /* implementáció */
}
```

Természetesen a fentieken kívül még néhány kisebb módosításra is szükség van a forráskódban, melynek megértése a programkód alaposabb ismeretét követelné meg az olvasótól, ezért ennek részletezésétől eltekintek.

Látható tehát, hogy az oktatóprogramban egy új függvény vagy test felvétele nem kíván a programozótól több órás fejlesztést, mindösszesen néhány tíz soros programkód megírásával jár.

6. ÖSSZEFOGLALÁS

Az oktatóprogram hosszadalmas munka eredménye, melyben ötvöződnek korábban megszerzett tapasztalataim, illetve a fejlesztés során felhalmozódott új ismereteim. A program készítésekor különösen nagy figyelmet fordítottam a magamban megfogalmazódó célok elérésére, mely rengeteg kihívást jelentett számomra a munka során.

Az oktatás során nagyon fontos a tanulók motiválása, az önellenőrzés és egyes területeken a tananyaghoz kapcsolódó szemléltetés. Különösen a függvények és a testek vizsgálata az, mely megkívánja a minél hatékonyabb demonstrációt. A szemléltetés már több száz éve a pedagógia szerves része, mely napjainkban is igen hatékony módszer. A megértést könnyíti, ha a tanuló látja a térbeli elemeket, közvetlenül észleli a függvények változásait, s nem csak a tanári előadást hallgatja. Célom volt tehát egy olyan oktatóprogram készítése, mely hasznos kiegészítő lehet a megfelelő matematikai terület elsajátításában, elmélyítésében, illetve hozzá kapcsolódó grafikus ábrák generálásában.

Úgy vélem az oktatóprogram funkcionalitása, használhatósága, hatékonysága és az oktatásban vagy éppen az önálló tanulásban betöltendő szerepe kielégítheti az itt és a bevezetőben megfogalmazott célokat. Sikerült megvalósítani egy olyan alkalmazást, mely a hibakezelés és a felhasználóval való folyamatos kapcsolattartás miatt nem bizonytalansághoz, hanem élményekkel teli tanuláshoz vezethet.

Az alkalmazás főképpen középiskolás tanulók számára készült. Azonban mivel az oktatóprogram tervezése során szem előtt tartottam a fejlesztési lehetőségeket, ezért úgy gondolom, hogy a program magasabb igények kielégítésére is alkalmas lehet a szükséges modulok hozzáadásával. További komponensekkel vagy grafikus felületekkel bővítve, a program az eddigieken túl, más funkciókat és elvárásokat is megvalósíthat. Vagyis nem csak az ajánlott korosztály, hanem akár az oktató fejlődéséhez is hozzájárulhat.

Azonban ne feledjük, hogy a számítógéppel segített oktatás növelheti ugyan a tanítás-tanulás színvonalát, továbbá hozzájárulhat a tananyag eredményes elsajátításához, de fontos, hogy az irányítás mindvégig a tanár illetve a felhasználó kezében maradjon.

7. IRODALOMJEGYZÉK

1. Nyékyné Gaizler Judit - JAVA 2 útikalauz programozóknak: 1.3. I.- kötet
   ELTE TTK Hallgatói Alapítvány Budapest 2001

2. Nyékyné Gaizler Judit - JAVA 2 útikalauz programozóknak: 1.3. II.- kötet
   ELTE TTK Hallgatói Alapítvány Budapest 2001

3. Nyékyné Gaizler Judit - Referencia 1.3.
   ELTE TTK Hallgatói Alapítvány Budapest 2001

4. Dr. Nyakóné dr. Juhász Katalin - Az informatika iskolai alkalmazásai,
   Debreceni Egyetem, Természettudományi Kar, Matematikai és Informatikai Intézet, 2000

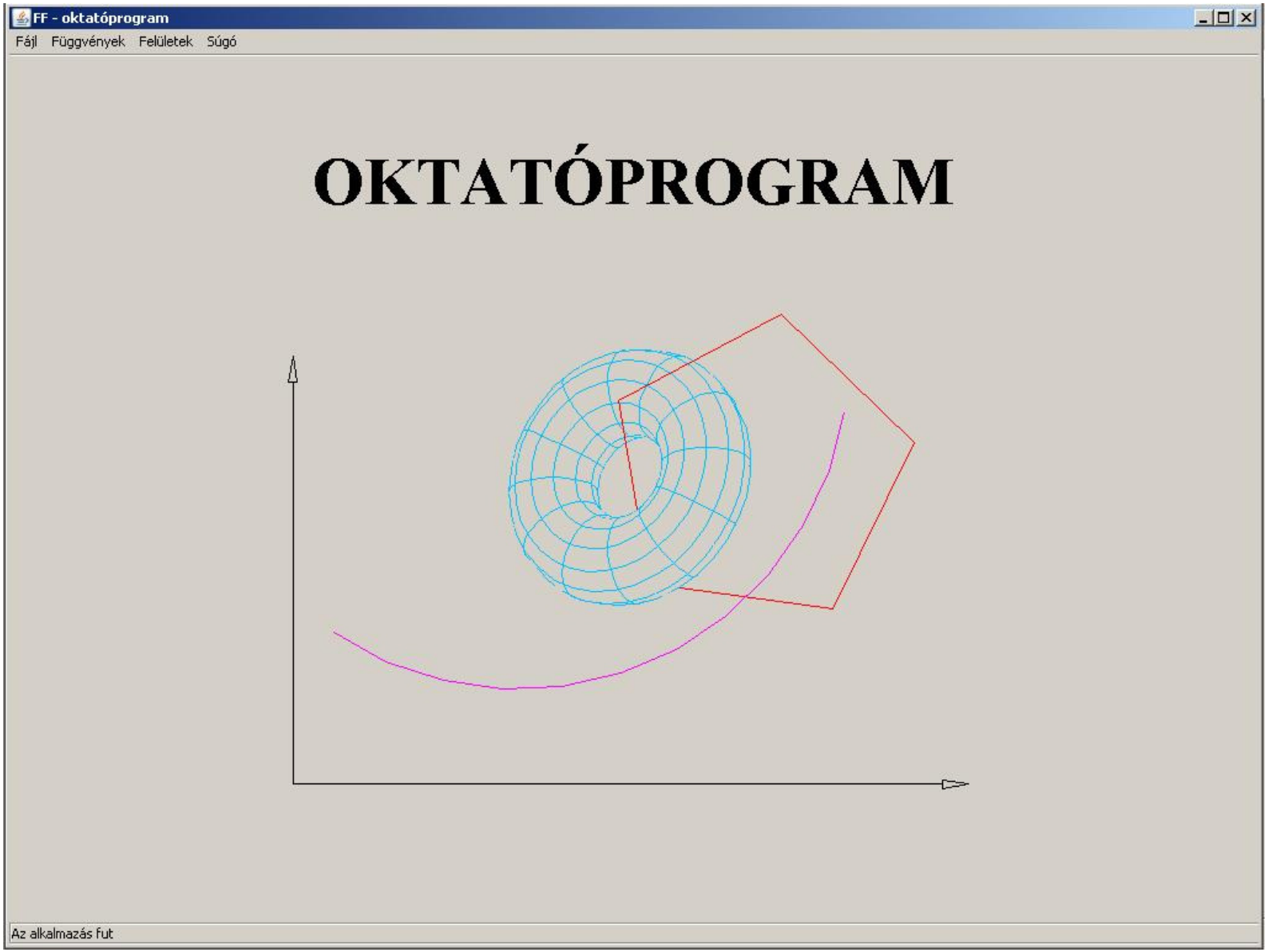
5. Dr. Schwarcz Tibor - Bevezetés a számítógépi grafikába tantárgy előadásjegyzete

6. Czapáry Endre - Matematika I. osztály
   Nemzeti Tankönyvkiadó, Budapest

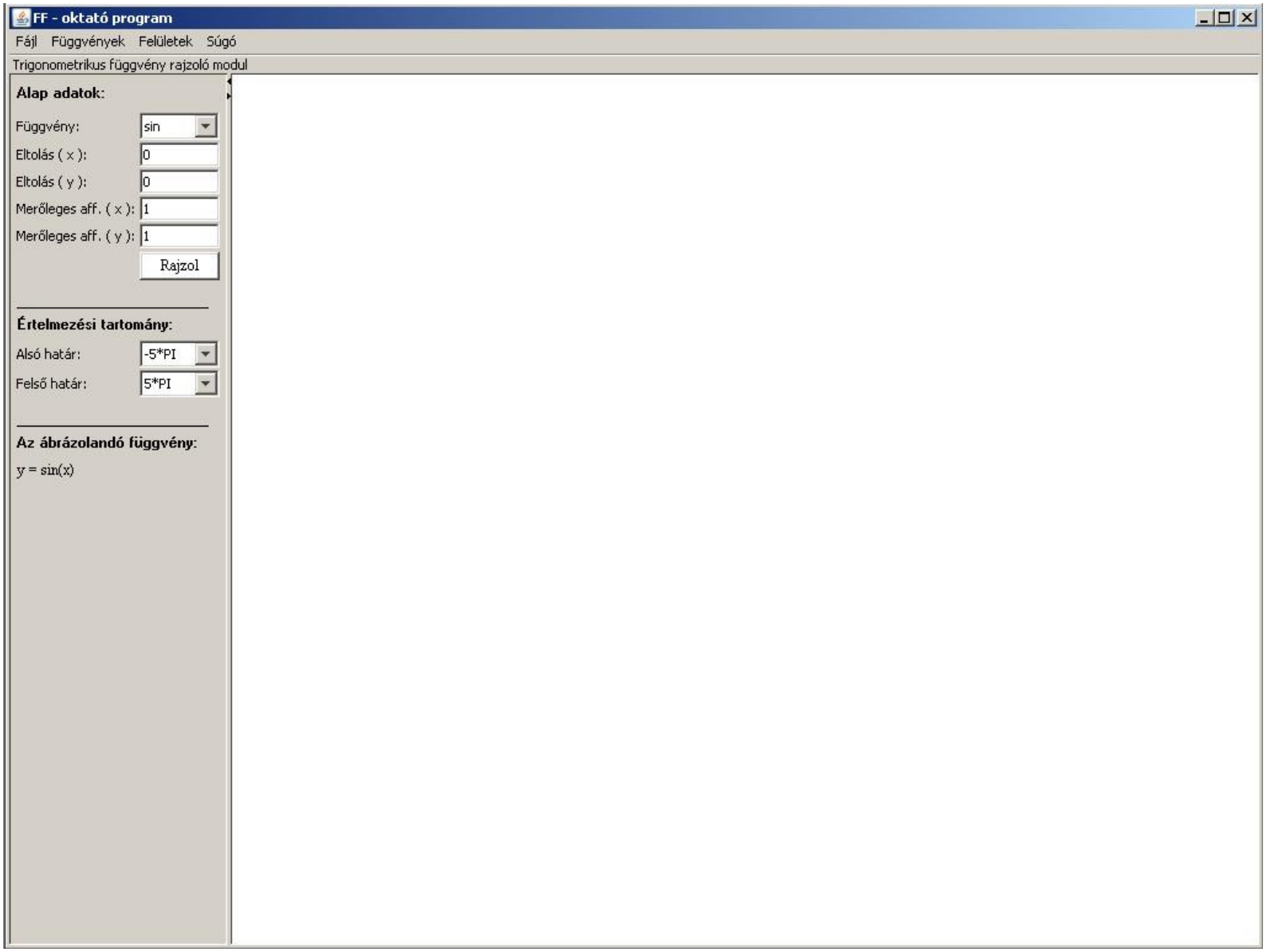
7. Czapáry Endre - Némethy Katalin - Matematika II. osztály
   Nemzeti Tankönyvkiadó, Budapest

8. Hajnal Imre - Matematika III.
   Nemzeti Tankönyvkiadó, Budapest

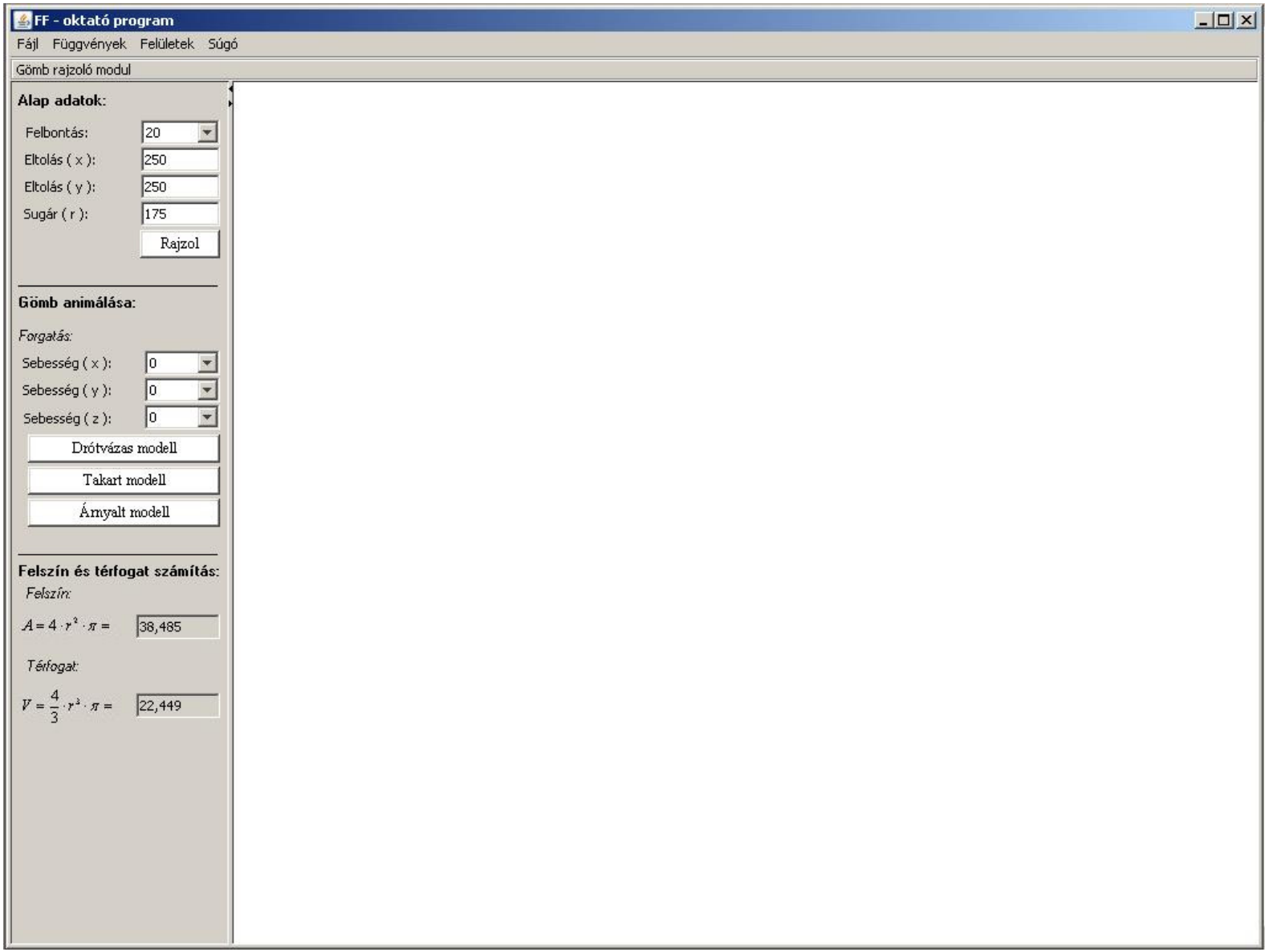
8. MELLÉKLET



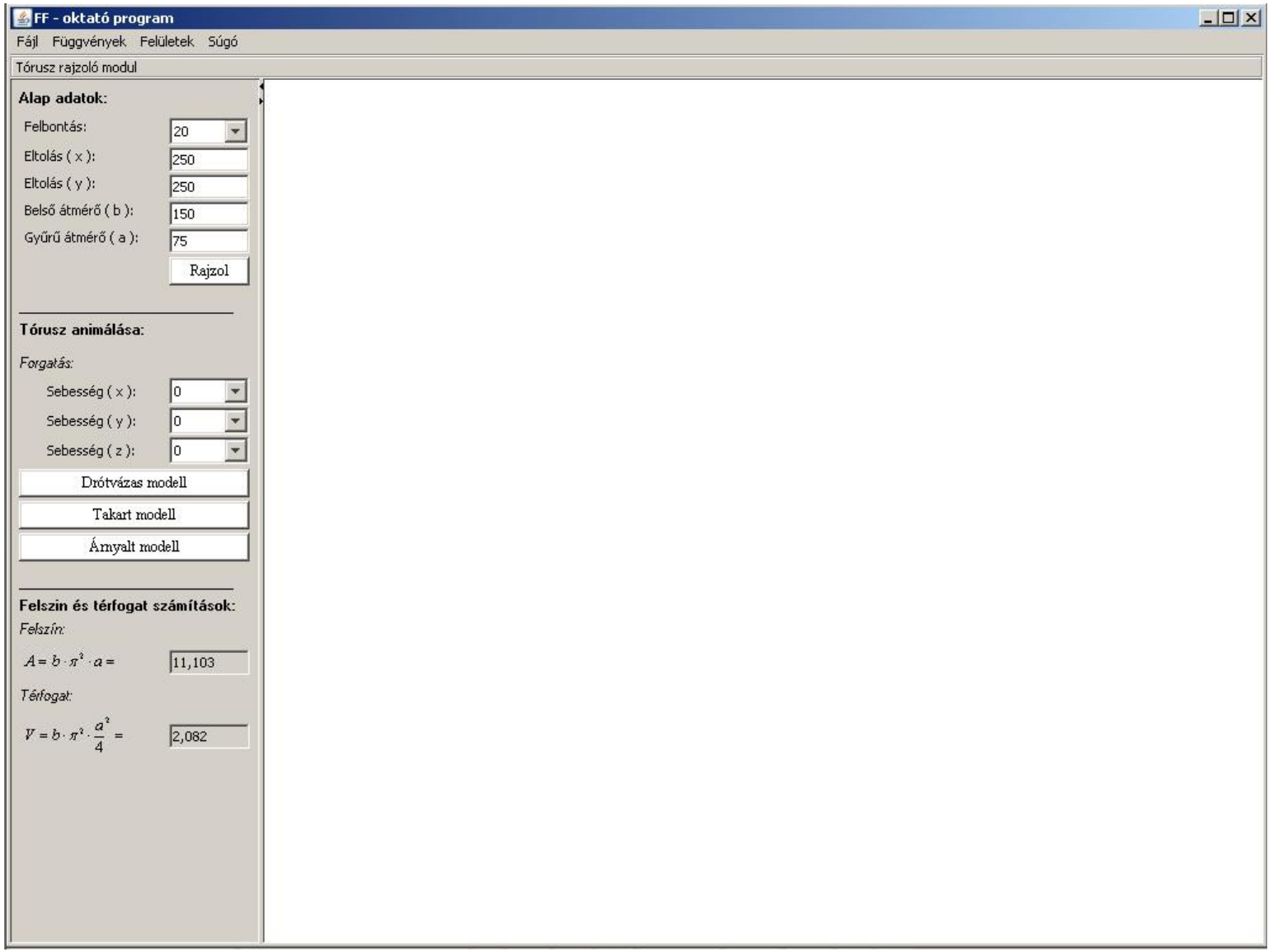
1. ábra: Fő ablak



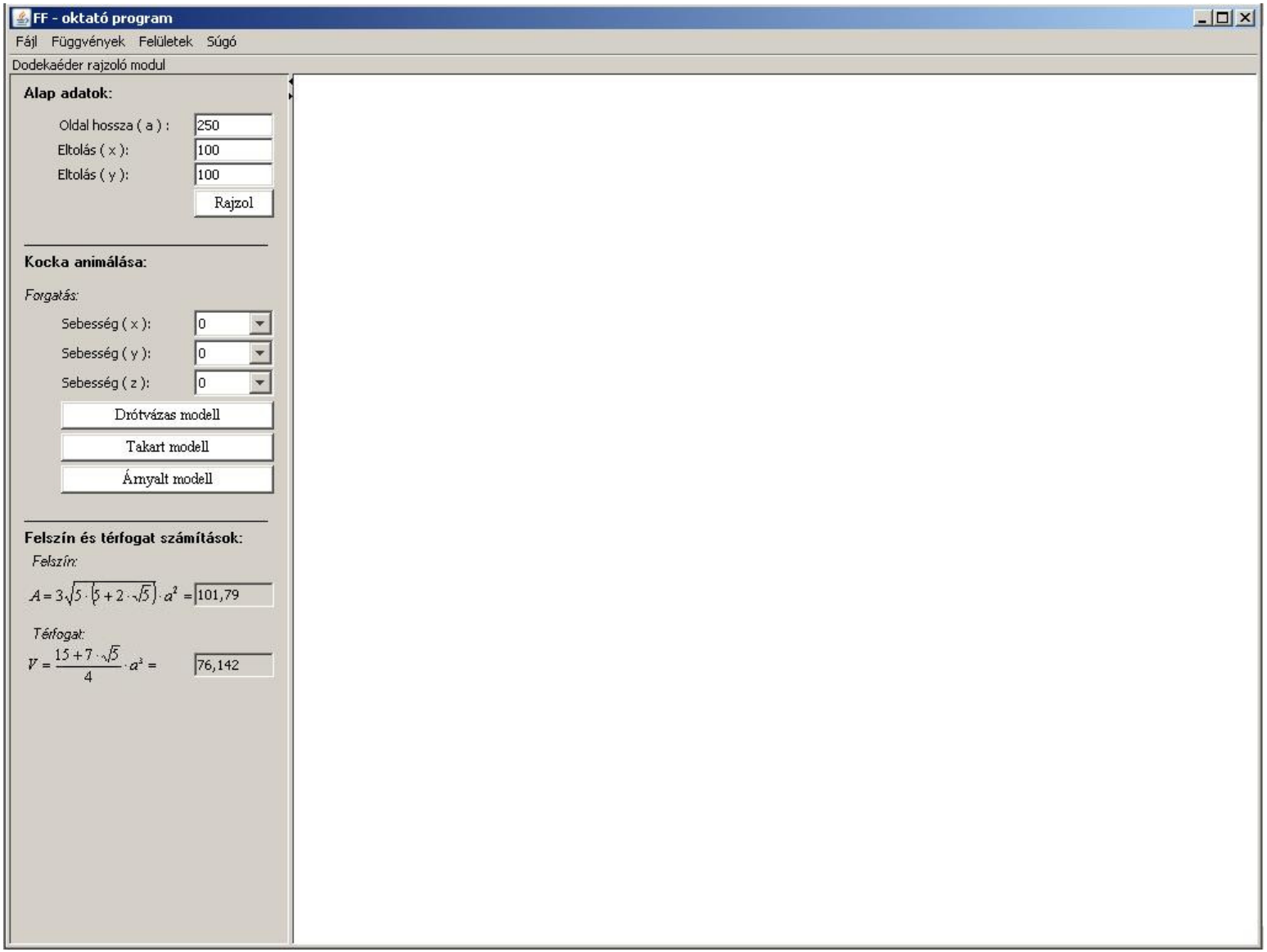
2. ábra: A függvények vizsgálatához tartozó grafikus felület



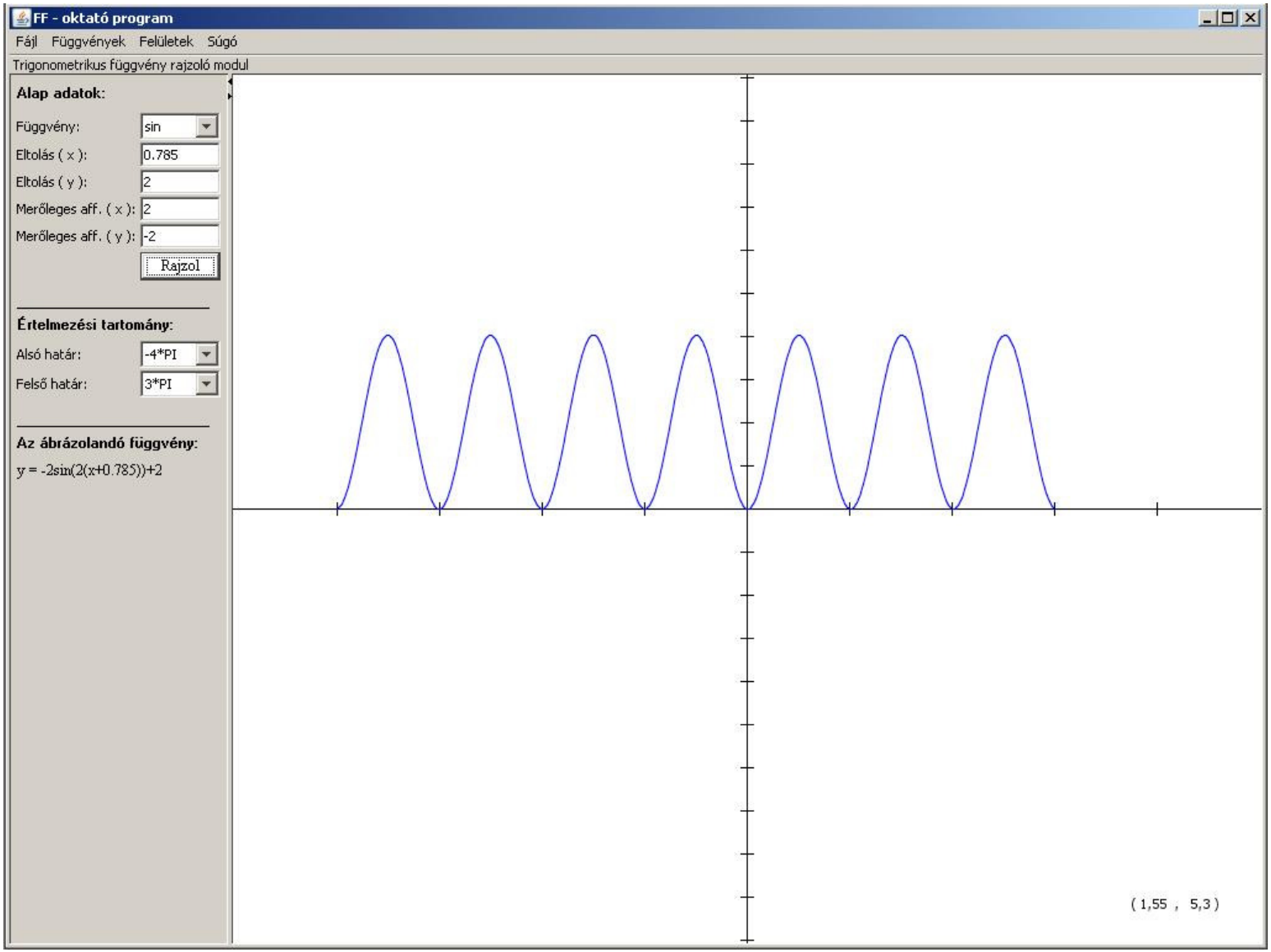
3. ábra: A gömb vizsgálatához tartozó grafikus felület



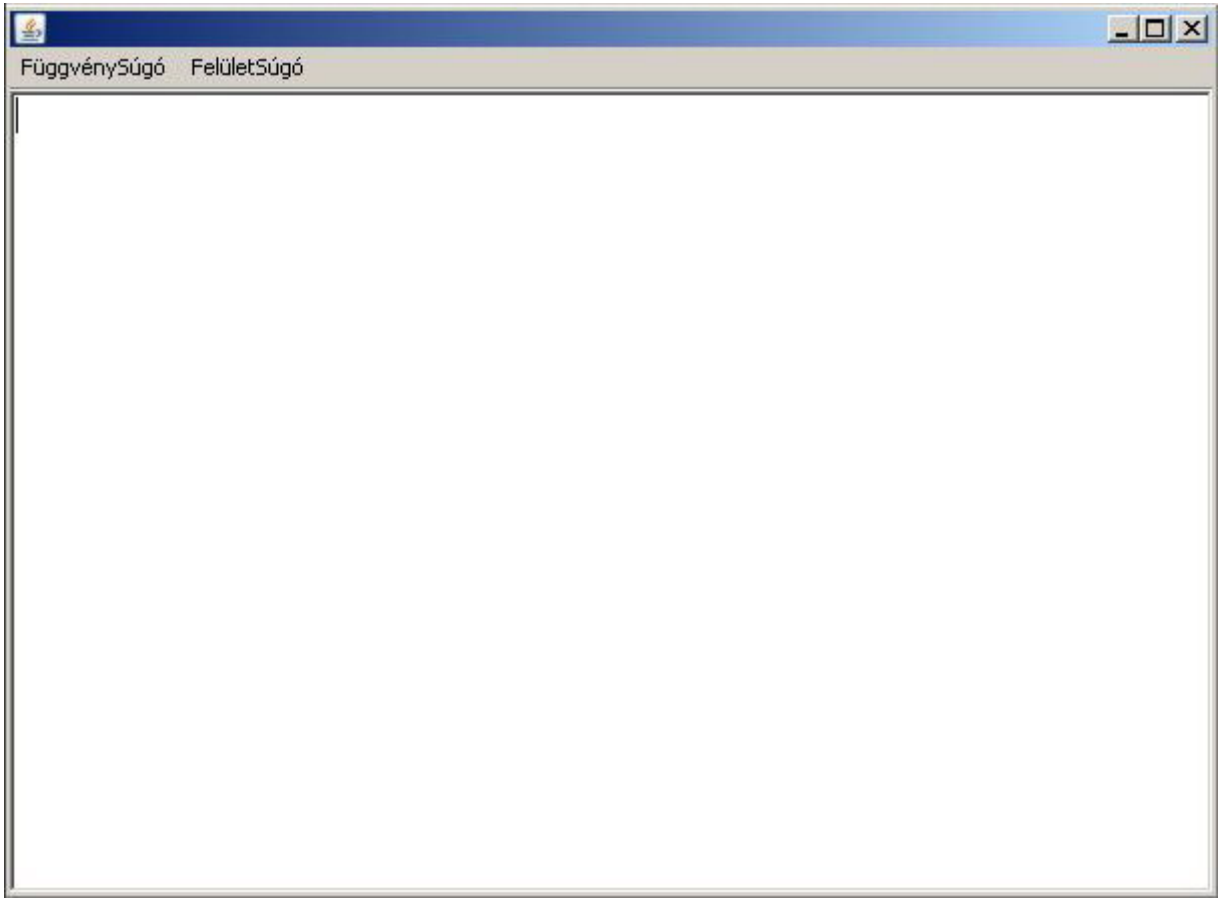
4. ábra: A tórusz vizsgálatához tartozó grafikus felület



5. ábra: A dodekaéder vizsgálatához tartozó grafikus felület



6. ábra: A példa végeredményének szemléltetése



7. ábra: A súgó ablak