Dunakavics

A Dunaújvárosi Főiskola online folyóirata 2014. II. évfolyam X. szám

Műszaki-, Informatikai és Társadalomtudományok

Dunakavics

A Dunaújvárosi Főiskola online folyóirata 2014. II. évfolyam X. szám

Műszaki-, Informatikai és Társadalomtudományok

Megjelelenik évente 12 alkalommal

Felelős szerkesztő Németh István
Tördelés Duma Attila
Szerkesztőség és a kiadó címe 2400 Dunaújváros, Táncsics M. u. 1/a.

Kiadja DUF Press, a Dunaújvárosi Főiskola kiadója
Felelős kiadó András István, rektor

A lap megjelenését támogatta TÁMOP-4.2.3-12/1/KONV-2012-0051
„Tudományos eredmények elismerése és disszeminációja
a Dunaújvárosi Főiskolán”.
http://dunakavics.duf.hu

ISSN 2064-5007

Tartalom

Dunakavics - 2014 / 10.

SCHMIDT DÁVID *

Testekkel végzett geometriai Boole-műveletek programozása és optimalizálása 2. rész

* *StruSoft Kft.*
E-mail: sydrafurynox@vip-mail.hu

Összefoglalás: Írásom témája a testekkel végzett geometriai Boole-műveletek programozása és optimalizálása. A Strusoft Kft-nél azt a feladatot kaptam, hogy írjam újra a FEM-Design programrendszerben lévő, régió és testműveletekért felelős programfüggvényeket. A cél az volt, hogy az új függvényekkel a testműveletek számítási ideje legalább a tizedére csökkenjen az eredetihez képest. Írásomban ennek a problémának a megoldását és az ehhez kapcsolódó ismeretanyagokat mutattam be. Röviden ismertettem a FEM-Design programrendszert és a testműveletekkel kapcsolatos probléma forrását. Bemutattam azokat az ismeretanyagokat, melyek a test- és régióműveletekkel kapcsolatos metódusok elméleti hátteréül szolgálnak. Ismertettem azokat testműveletekkel kapcsolatos optimalizálási lehetőségeket amelyek az új függvények gyorsulását lehetővé teszik. Sorban bemutattam azokat a fontosabb függvényeket, amelyek a testek és régiók felépítéséért és az azokkal való műveletvégzésért felelősek. Végezetül összegeztem az írás eredményességét a régi és új függvények futási idejének összehasonlításával.
Kulcsszavak: Testműveletek, Boole-algebra, optimalizálás.

Abstract: My thesis work is the Programming and Optimization of Geometrical Bool-operations of Solids at Strusoft Kft. According to the task it is necessary to rewrite the program functions what responsible for the region and solid operations in the FEM-Design program system. The goal is to make the calculation time of the solid operations at least ten times shorter. In the thesis I introduced this solution of problem and the corresponding knowledge for that. I showed shortly the FEM-Design program system and the origin of the problems of the solid operations. I introduced the knowledge what gives the theoretical background of the region and the solid operation methods. I presented the possible optimizations what makes the new solid operation functions much faster. I introduced all of the important solid and region

building and operation functions. Finally I summarized the efficiency of the thesis through comparing the running time of the old and new functions.
Keywords: Solid operations, Boole-algebra, optimization.

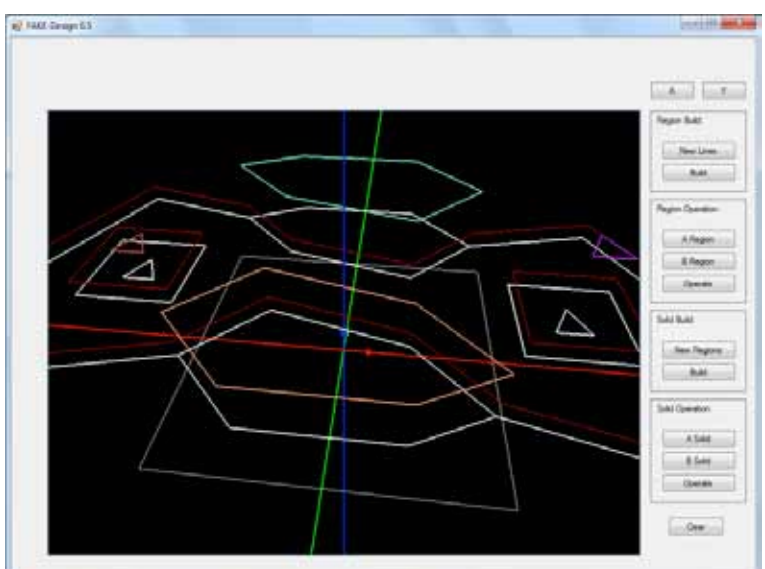
(A cikk az előző lapszámunkban megjelent 1. rész folytatása)

7. A régiófelépítő függvények bemutatása

Ebben a fejezetben a régiófelépítő műveletek megvalósítása programfüggvényeken keresztül kerül bemutatásra. A kód nagy mérete miatt csak a fontosabb függvények és az ezekhez létrehozott struktúrákat ismertetjük. A függvények részben a 2.1-es alfejezetben *(lásd: Dunakavics 2014/9. 52. old.)* tárgyalt elméleti rész leképezései, az ott szerepeltetett információk itt nem tárgyaljuk újra. Az ismertetés átfogó, de nem utasításról utasításra halad, a részletek túlzott vizsgálatába terjedelmi korlátok miatt nem bonyolódom.

A lenti ábrán látható fehér színnel jelzett rendezetlen szakaszsereg és az abból felépített – a jól láthatóság kedvéért térben eltolva – színes régiók.

7–1. ábra. A fehér színű rendezetlen szakaszsereg és a belőle generált színes régiók.



7.1. Régiókeresés

A régiókereső függvény egy központi függvény. Feladata, hogy egy kezdeti, síkban elhelyezett rendezetlen szakaszseregből megkeresse azokat a szakaszokat, amik régiókat alkotnak, majd ezen szakaszokból felépített régiókat visszaadja egy referencián keresztül.

7.1–1. ábra. A régiókereső függvény prototípusa.

```
544  void SearchRegions(
545    std::vector<POINTW3>  &node,
546    std::vector<FDLine>   &line,
547    LinePile              &linepile,
548    std::vector<FDRegion> &region,
549    VECTOR3D              normal=VECTOR3D(0,0,0)
550    );
```

A függvény első paramétere egy csomópontokat tároló vektor, a második paraméter egy vonalakat tároló vektor, a harmadik a függvény szempontjából fontos szakaszok indexeit tároló saját típusú változó, a negyedik az eredményrégiókat tároló vektor, az ötödik pedig egy a művelet síkjára merőleges vektort tárolja. Amennyiben meghíváskor az utolsó paraméter nem lenne megadva, egy null vektor értékét kapja meg.

A műveletek sora egy ellenőrzéssel kezdődik. Amennyiben a normal változó null vektor értékét kapta, lefut egy függvény, ami kiszámolja a műveleti síkra merőleges normálvektort.

Mivel a csomópontok koordinátái térben vannak megadva, a régiófelépítés pedig síkban értelmezett, ezért a következő művelet a csomópontok vetítése, illetve annak eldöntése, hogy melyik két koordinátatengely által kifeszített síkra lehet vetíteni. A vetítéshez, illetve később a vetített pontok eléréséhez a ProjectedNodes nevű struktúrát használom.

Pontok vetítése

A ProjectedNodes nevű struktúra feladata a térbeli pontok vetítése. Valójában a példányosításkor a vetített pontok nem jönnek létre. A struktúra eltárol magában egy mutatót az eredeti csomópontok vektoráról, majd a vetített pontokra ezen a mutatón és egy segédtömbön keresztül indexekkel hivatkozik.

7.1–2. ábra. A ProjectedNodes nevű struktúra definíciója.

```
159 struct ProjectedNodes
160 {
161   int                          pmode;
162   int                          length;
163   int                          x, y;
164   std::vector<POINTW3>::pointer node_p;
165
166   ProjectedNodes(std::vector<POINTW3> &_node, int _pmode);
167   ProjectedNodes
168     (std::vector<POINTW3> &_node, const VECTOR3D &_normal);
169
170   REAL                         X(const int &_index);
171   REAL                         Y(const int &_index);
172   PlanePoint                   operator[](const int &_index);
173   PlanePoint                   ProjectNode(POINTW3 _p3);
174   int                          size();
175   PlaneVector                  ProcjectVector(VECTOR3D _v3);
176   POINTW3                      InverseNodeProjection_NodeOnLine(
177                                  VECTOR3D _normal,
178                                  double _D,
179                                  PlanePoint pp);
180   POINTW3                      InverseNodeProjection_NodeOnLine(
181                                  NodeIndex a,
182                                  NodeIndex b,
183                                  PlanePoint pp);
184   BoundingRectangle            BoundingCuboidProjection(
185                                  const BoundingCuboid &bc);
186   bool                         IsRegionNotParallelToProjection(
187                                  VECTOR3D &_normal);
188   bool                         IsPointHigherThanOther(
189                                  POINTW3 &p1,
190                                  POINTW3 &p2);
191 };
```

A vetítésre használt sík kiválasztása egy, a konstruktorban paraméterként átadott, normálvektor alapján dől el. Amelyik a legnagyobb a normálvektor x, y és z koordinátája közül, a vetítés során az a koordinátát hagyjuk el a csomópontok koordinátája közül.

Ennek a struktúrának nagy jelentősége van a program egészének szempontjából, számtalan függvény használja a régió- és testműveleteknél egyaránt.

A struktúra függvényei lehetővé teszik, hogy az aktuális vetítési irány szerint más külső pontokat, vektorokat és befoglaló téglatesteket vetítsünk síkra. Nekünk nem kell foglalkoznunk azzal, hogy a vetítés milyen síkra történik, a struktúra önmagában, egységes módon lekezel minden vetítési szituációt. A struktúra ugyanazon példányán keresztül elvégzett vetítések eredményei garantáltan egy síkba kerülnek. A [] operátor kiterjesztésével pedig lehetőségünk van úgy (indexszel) hivatkozni a vetített pontokra, mint az eredeti háromdimenziós csomópontvektor elemeire.

A vetítés után meg is kezdődhet a szakaszsereg rendezése az erre létrehozott függvény segítségével. Valójában nem a vonalakat rendezzük sorba, hanem a vonalak indexeit a linepile nevű változóban. Értelemszerűen ez kevesebb művelettel jár, mintha magát a vonalvektort rendezgetnénk.

A következőkben egy ciklusművelet kezdődik, amelyen belül a régiók és kontúrjaik keresése megtörténik. Az iteráció addig tart, amíg minden szakasz feldolgozásra nem kerül.

A ciklusmagban lévő műveletek egy kezdőszakasz vizsgálatával indulnak. A kezdőszakasz a sorba rendezés szerinti első olyan szakasz, ami még nem tartozik régiókontúrhoz.
A következő művelet során megvizsgáljuk, hogy a kezdőszakaszunk helyileg hol található. Amennyiben a kezdőszakasz egy már részlegesen felépített régióban található, akkor azon a régió egy belső kontúrja kezdőszakaszát fogja képezni. Ha a kezdőszakasz egyik meglévő régiónak sem tagja, akkor egy új régió külső kontúrjának kezdőszakasza lesz.

A feltételvizsgálat után a régió kontúrkeresését végző függvény meghívódik, majd elágazástól függően megtörténik a régió normálvektorának beállítása, a régió és kontúr befoglaló téglatesteinek kiszámítása.

A ciklusművelet befejezése után a függvény kilép.

7.2. A szakaszok sorba rendezése

A szakaszrendező függvény felel a vonalak irányának beállításáért és a vonalak sorba rendezéséért.

7.2–1. ábra. A szakaszrendező függvény prototípusa.

```
549 void OrderLines(
550   ProjectedNodes       &pnode,
551   std::vector<FDLine> &line,
552   LinePile             &linepile
553   );
```

A függvény a vetített pontokat, a vonalakat, és a vonal-indexeket tároló paraméterek alapján dolgozik.

A műveletsor első felében, egy iterációs művelet során végighaladunk az összes szakaszon, aminek az indexe szerepel a linepile nevű változóban és úgy állítjuk be a szakaszok végpontjait, hogy mindig a szakaszt tároló objektum a-változója tárolja a referenciapont indexét.

A műveletsor második felében sorba rendezem a szakaszokat a 2.1-es alfejezetben *(lásd: Dunakavics 2014/9. 52. old.)* ismertetett feltételek szerint.

7.3. Régiótagság-vizsgálat

A régiótagság megállapítására megalkotott függvény azt vizsgálja, hogy egy szakasz egy másik régión belül található-e. Amennyiben egy másik régióban található, a függvény visszaadja a tartalmazó régió indexét. Ellenkező esetben -1-es értékkel tér vissza.

7.3–1. ábra. A régiótagságot vizsgáló függvény prototípusa.

```
497  int IsRegionPart(
498    ProjectedNodes              &pnode,
499    const std::vector<FDLine>   &line,
500    const std::vector<FDRegion> &region,
501    const int                   &rvs_size,
502    const LineIndex             &observed_line
503    );
```

Az első feltételes vizsgálatnál megnézzük, hogy létezik-e már részlegesen felépített régió. Amennyiben nem létezik, a függvényünk azonnal -1-es visszatérő értékkel kilép, ha létezik, akkor folytatja futását.

A következő műveletek során előállítjuk a vizsgált szakasz felezőponti középpontjának vetületét, majd egy ciklus segítségével végigjárjuk a meglévő régiókat és megnézzük, hogy a pont melyik régióvetületben található. A bentlévőség akkor teljesül, ha a felezőpontból húzott végtelen félegyenes páratlan számúszor metszi egy régió falát. A metszéseket egy másik függvénnyel számoltatjuk meg. Az első páratlan számú metszés esetén a függvény visszatér az aktuálisan vizsgált régió indexével. Ha a bentlévőség feltétele egyik régiónál sem teljesül, akkor -1 a visszatérő érték.

7.4. Régiómetszések keresése

A SearchCutsInRegion(…) nevű függvény (*7.4-1. ábra*) megszámolja, hogy egy paraméterként kapott pontból húzott végtelen félegyenes hányszor metszi a vizsgált régió falát alkotó szakaszsereget.

7.4–1. ábra. A végtelen félegyenes régiómetszéseit számoló függvény prototípusa.

```
531  int SearchCutsInRegion(
532    ProjectedNodes              &pnode,
533    const std::vector<FDLine>   &line,
534    const FDRegion              &region,
535    const PlanePoint            &halfline
536    );
537
```

A függvény elején deklarálunk egy cuts nevű, int típusú változót és 0 kezdőértékkel látjuk el.

Egymásba ágyazott ciklusok segítségével bejárjuk a vizsgált régió kontúrjait és azok szakaszait. A metszésvizsgálatok előtt a régió és kontúrjainak befoglaló téglatesteivel előzetes vizsgálatokat végzünk. Amennyiben nincs átfedésben a befoglaló és a végtelen félegyenesünk, fölösleges lenne metszéseket keresgélnünk. Ezzel a módszerrel sok műveletet spórolunk meg.

Mivel a végtelen félegyenesünk az Y tengellyel párhuzamos és –Y irányban végtelen, ezért a szakaszokon keresett metszéspont kutatása egyszerű tartományvizsgálatokra és aránypárokkal való számításokra redukálódik. Ha a végtelen félegyenes végpontjának x koordinátája a vizsgált szakasz végpontjainak x koordinátái közé esik, és a félegyenes végpontja a szakasz felett található, akkor a félegyenesünk metszi a szakaszt. Minden ilyen metszésnél növeljük eggyel a cuts nevű változónk értékét.

A függvény visszatérési értéke a cuts nevű változó értéke lesz.

7.5. Régiókontúr keresése

A régiókontúrkereső-függvény (*7.5–1. ábra*) feladata, hogy egy régiókontúrt felépítsen egy rendezett szakaszsereg egymáshoz csatlakozó szakaszaiból.

7.5–1. ábra. A régió kontúrt feltérképező függvény prototípusa.

```
563  void SearchRegionContour(
564    ProjectedNodes            &pnode,
565    std::vector<FDLine>       &line,
566    const LinePile            &linepile,
567    std::vector<bool>         &regionpart,
568    const int                 &startingline,
569    const bool                &isholeelement,
570    std::vector<LineIndex>    &contour_lines
571    );
```

Első lépésben megállapítjuk, hogy a készülő kontúrunk külső vagy belső kontúr lesz-e. Ezt az isholeelement nevű paraméterváltozó mondja meg nekünk. Ha külső kontúrt építünk, akkor a régió normálvektorával szemben állva az óramutató járásával ellentétes irányba haladunk a kontúrszakaszokhoz csatlakozó szakaszok keresésében, ha belső kontúrt építünk, akkor az óramutató járásával megegyező irányba haladunk.

A kontúrkeresés egy iterációs műveleten keresztül valósul meg. Az iteráció mindig a következő kontúrszakasz felismerése után ismétlődik, egészen addig, amíg vissza nem érünk a kiindulási ponthoz.

Minden újonnan felismert kontúrelemnél egy belső ciklus segítségével egy vektorban összegyűjtjük az új kontúrelem szabad végéhez csatlakozó, még fel nem használt szakaszok indexeit.

Az összegyűjtött csatlakozó-szakaszok közül a GetNextLine(...) függvény segítségével kiválasztjuk a megfelelő csatlakozó-szakaszt, ami a kontúr új elemét képezi.

7.6. A csatlakozó-szakasz kiválasztása

Amennyiben egy félkész régiókontúr végpontjához több szakasz is csatlakozik a GetNextLine(...) függvény feladata, a 2.1-es alfejezetben tárgyalt tézis alapján annak a szakasznak a kiválasztása, amellyel a kontúr építése folytatható.

7.6–1. ábra. A megfelelő csatlakozó szakasz kiválasztására megírt függvény prototípusa.

```
510 void GetNextLine(
511   ProjectedNodes              &pnode,
512   const std::vector<FDLine>   &line,
513   const LinePile              &linepile,
514   const std::vector<int>      &actualconnectedlines,
515   const NodeIndex             &actualendpoint,
516   int                         &actualline
517   );
```

A függvényben első lépésben meghatározzuk a régiókontúr utolsó szakaszának irányvektorát, úgy, hogy a szakasz szabad végéből a másik végébe mutasson.

Következő lépésben egy ciklusműveleten keresztül egyesével összehasonlítjuk az egyes csatlakozó szakaszok irányvektorai és a régió utolsó felismert szakaszának irányvektora által bezárt szögeket. A szögeket az óramutató járásával megegyező irányban az utolsó felismert kontúrszakasztól számítjuk. Amelyik szakaszhoz a legkisebb szög tartozik, az kerül kiválasztásra. A kiválasztott szakasz indexével felülírjuk az actualline nevű változót.

7.7. Sík normálvektorának előállítása szakaszseregből

Feladatunk egy szakaszsereg síkjára merőleges vektor előállítása. Ezt a GetLinePileNormalVector(…) nevű függvény végzi el nekünk.

Normálvektorunk előállításához rendelkezésünkre áll egy egész szakaszsereg minden tulajdonsága. Persze a feladat elvégzéséhez számunkra elegendő két szakaszt kiválasztani, majd ezek irányvektorának vektoriális szorzata szolgáltatná nekünk a normálvektort. Az első problémát az jelenti, hogy nem tudjuk, hogy a szakaszaink hogyan helyezkednek el. Nem választhatunk ki véletlenszerűen két szakaszt a szakaszseregből, mert lehet, hogy bizonyos szakaszaink irányvektora párhuzamos, ami lehetetlenné teszi, hogy normálvektort származtassunk belőlük. Második problémánk, hogy numerikus pontosság szempontjából használható normálvektort kell előállítanunk, amihez minél nagyobb szöget kell bezárnia vektorainknak. A jó eredmény előállításához meg kell keresnünk azt a két szakaszt, amelyek a legnagyobb szöget zárják be egymással.

A függvényünk szakaszpáronként kiszámolja a szakaszok egymással bezárt szögét és megkeresi azt a két szakaszt, ahol a bezárt szög a lehető legközelebb áll a derékszöghöz. A kiválasztott két szakasz irányvektorai tárolásra kerülnek.

7.7–1. ábra. a síkban elhelyezkedő szakaszsereg normálvektorát előállító függvény prototípusa.

```
535  VECTOR3D GetLinePileNormalVector(
536    const std::vector<POINTW3> &node,
537    const std::vector<FDLine> &line,
538    const LinePile &linepile
539    );
```

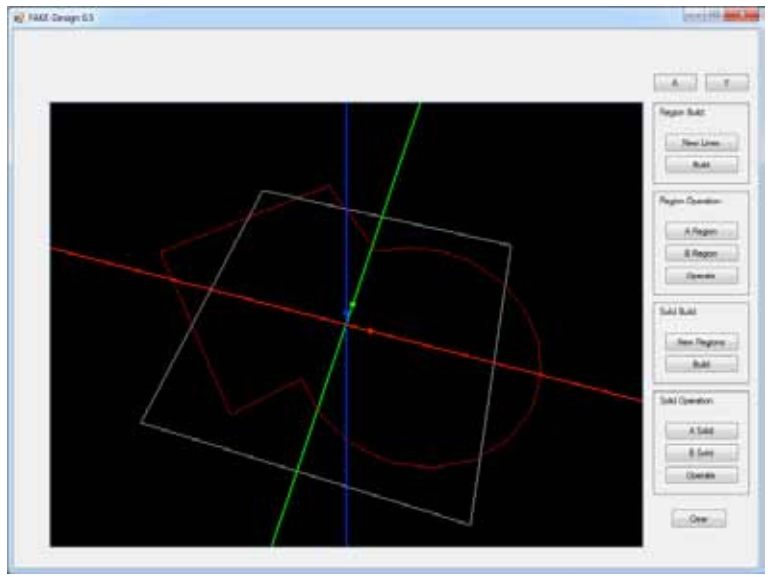
A függvény visszatérő értéke a két eltárolt vektor vektoriális szorzatának normáltja, amit a VectorProduct(…) és a NormalizedVector(…) függvények segítségével számolunk ki.

8. A régióműveletekért felelős függvények bemutatása

Ebben a fejezetben a 2.2-es alfejezetben *(lásd: Dunakavics 2014/9. 50. old.)* ismertetett régióműveletek programkódbeli leképezését mutatjuk be. Az ismertetés átfogó, de nem túlzottan részletes. A dolgozat terjedelmi korlátozásai miatt csak a fontosabb műveletek és utasításokat ismertetjük. A függvényprototípusok kódját ábrákon tekinthetjük meg, melyek a forráskódban történő navigációt szeretnék megkönnyíteni.

Az alábbi ábrán megtekinthető a régióműveleteket végző függvények által megvalósított unióművelet, ami egy négyzet és egy kör között lett elvégezve.

8–1. ábra. Egy négyzet és egy kör uniója.



8.1. Régióműveletek

A régióműveletekért a RegionOperator(...) nevű függvény (*8.1–1-es ábra*) felelős. Ez a központi függvény, innen történnek azoknak a függvényeknek a meghívásai is, amiket a későbbi alfejezetben ismertetjük.

8.1–1. ábra. A régióműveletekért felelős függvény prototípusa.

```
570  void RegionOperator(
571    std::vector<POINTW3>  &node,
572    std::vector<FDLine>   &line,
573    FDRegion              &region1,
574    FDRegion              &region2,
575    SolidOperation        operationtype,
576    std::vector<FDRegion> &region
577    );
```

A függvény paramétermezőjében láthatjuk a csomópontokat tároló vektort, a régiók szakaszainak tárolóvektorát, az első régióttároló objektumot, a második régiót tároló objektumot, a művelet típusát meghatározó változót és az eredményrégiók tároló vektorát.

A függvénytörzs első lépése, hogy megállapítsuk a befoglaló téglatestek alapján létrejöhet-e művelet a két régió között. Ha nem, akkor azonnal kilépünk.

A következő műveletcsoport során a CutContourLines(…) függvény segítségével megkeressük a régiók összemetsződéseinél létrejövő metszéspontokat, majd a LineFragmentation(…) nevű függvénnyel létrehozzuk a „széttördelt" kontúr szakaszokat. A metszéspontok jelöléséhez és a szakasztöredékek indexeinek tárolásához a RegionRemains2 nevű struktúrát vezettük be.

Ezek után a kért műveletnek megfelelően ki kell válogatnunk, hogy melyik szakaszokra és szakasztöredékekre van szükségünk ahhoz, hogy az eredményrégióinkat felépíthessük. Egy switch case-utasítás, és a paraméterben kapott, művelettípust meghatározó változó segítségével döntjük el, hogy milyen kiválogatást szeretnénk. Itt vagy az FDRegionOperate(…), vagy az FDRegionExcludedOr(…) nevű függvényt hívjuk be. A kiválogatott szakaszok indexeit a linepile nevű változóban lesznek eltárolva.

Végül már csak az előző fejezetben ismertetett SearchRegions(…) nevű régiókereső és -felépítő függvényt kell meghívnunk, hogy létrejöhessenek az eredményrégióink.

8.2. Kontúrszakaszok elmetszése

A CutContourLines(…) nevű függvény (*8.2–1-es ábra*) felelős a régiók összemetsződéseinél a kontúrszakaszok metszéspontjainak tárolásáért.

8.2–1. ábra. A kontúrszakaszok elmetszéséért felelős függvény prototípusa.

```
591  void CutContourLines
592    (
593    std::vector<POINTW3>  &node,
594    std::vector<FDLine>   &line,
595    RegionRemains2        &regionr1,
596    RegionRemains2        &regionr2,
597    SolidOperation        operationtype
598    );
```

A műveleteink során két egymásba ágyazott cikluson keresztül bejárjuk a két régiónkat és párosával megvizsgáljuk, hogy a kontúrszakaszaink metszik-e egymást és ha igen, akkor hol. A régiók metsződéseinek könnyű kezeléséhez és a számítások során keletkező eredmények tárolásához a RegionRemains2 nevű struktúra objektumait használjuk.

A szakaszok metszésvizsgálatánál először megnézzük, hogy a két szakasz egymásra esik-e, ha nem, akkor kereshetjük a metszéspontot. A lehetséges metszéspontot itt is síkon állítjuk elő, majd a kétdimenziós pontot visszavetítjük térbe. A metszéspontot mindkét szakaszhoz tartozó segédobjektumban tároljuk.

8.3. Szakasztördelés

A szakasztördelő függvény (*8.3–1-es ábra*) felelős azért, hogy az előző alfejezetben tárgyalt metszéspontok alapján előállítsa a metszések után keletkező szakasztöredékeket, mint új szakaszokat.

8.3–1. ábra. A szakasztördelő függvény prototípusa.

```
615  void LineFragmentation(
616    std::vector<POINTW3>  &node,
617    std::vector<FDLine>   &line,
618    LineAddon2            &actual
619    );
```

A paraméterben megkaptuk azt az objektumot, ami az aktuális szakasz tördeléséhez szükséges információkat tárolja. Ez az objektum az erre a célra létrehozott LineAddon2 nevű struktúra egy példánya.

A függvény első felében iterációs műveleteken keresztül sorba rendezzük a szakaszon keletkezett metszéspontokat, a szakasz kezdőpontjától mért távolságuk szerint növekvő sorrendben.

A függvény második felében egy újabb ciklus segítségével a metszéspontokon sorban végighaladva, a pontokból párosával szakaszokat képezünk.

8.4. Szakaszválogatás műveletek szerint

Az eredményrégiót képező szakaszok kiválogatását az FDRegionOperate(…) és az FDRegion Excluded-Or(…) nevű függvények végzik el. A kizáró-vagy művelet kivételével minden műveletnél az FDRegion-Operate(…) függvényt (*8.4–1-es ábra*) használjuk.

A kizáró-vagy szerinti válogatást végző FDRegionExcludedOr(...) függvény azért lett külön megírva, mert ennél a műveletnél nem kell vizsgálnunk a szakaszaink régió szerinti bentlévőségét.

8.3–1. ábra. Az unió, kivonás és közösrész műveletek szerinti kiválogatást végző függvény prototípusa.

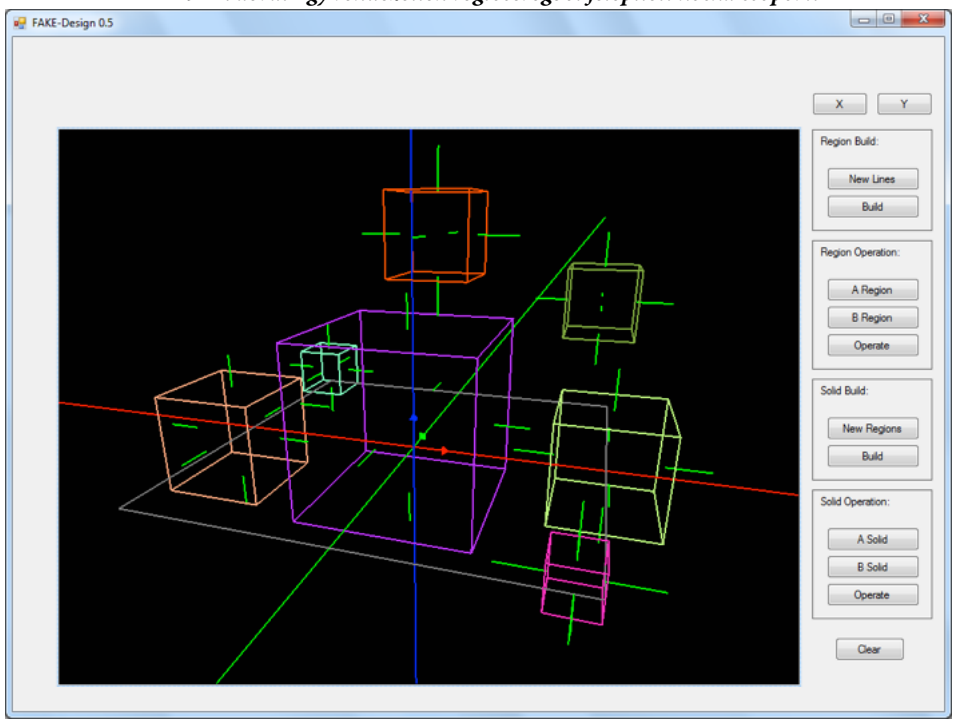
```
622  void FDRegionOperate(
623    ProjectedNodes          &pnode,
624    std::vector<FDLine>     &line,
625    FDRegion                &region1,
626    FDRegion                &region2,
627    RegionRemains2          &regionr1,
628    RegionRemains2          &regionr2,
629    bool                    b1,
630    bool                    b2,
631    LinePile                &linepile
632    );
```

A függvény szerkezete viszonylag egyszerű: ciklusok segítségével bejárjuk az elmetszett régióinkból létrejött szakaszseregünket és minden szakasznál megvizsgáljuk, hogy a szakasz a másik régión belül található-e. Adott művelettípustól függ, hogy éppenséggel a szükséges szakaszainknak épp kívül, vagy belül kell-e lenniük a másik régión. A szükséges szakaszaink indexeit egy paraméterben kapott vektorban tároljuk.

9. A testfelépítő függvények bemutatása

Ez a fejezet a 3.1-es alfejezetben *(lásd: Dunakavics 2014/9 52. old.)* bemutatott testfelépítő eljárás megvalósítását mutatja be az általam írt programfüggvényeken keresztül. Az ismertetés átfogó, de nem merül túlzott részletekbe a szakdolgozat terjedelmi korlátai miatt. A lenti 9–1. ábrán látható egy rendezetlen régióseregből felépített kockacsoport. A különböző testeket a program különböző színnel jelez. Itt az egyes régiókhoz tartozó normálvektorokat is megjelenítettük zöld színnel.

9–1. ábra. Egy rendezetlen régióseregből felépített kocka csoport.



9.1. Testfelépítés

A testfelépítő függvény (*9.1–1-es ábra*) keresi meg és építi fel a testeket egy rendezetlen régióhalomból. Ez egy központi függvény, amelyen belül számtalan másik függvényhívás történik.

9.1–1. ábra. A testfelépítő függvény prototípusa.

```
554 void SearchSolids(
555   std::vector<POINTW3> &node,
556   std::vector<FDLine> &line,
557   std::vector<FDRegion> &region,
558   std::vector<FDSolid> &solid
559   );
```

A függvény első paramétere egy csomópontokat tároló vektor, a második a szakaszokat tároló vektor, harmadik a régiókat tároló vektor, a negyedik pedig a függvényben előállított testek tárolója.

Első lépésben rendezzük a régiókat. Ezt az OrderRegions(…) nevű függvény segítségével oldjuk meg. Ezek után deklaráljuk a későbbi műveletekhez szükséges változókat és tárolókat. A következő fontos állomásnál egy olyan vektor feltöltése történik, amiben LineAddon típusú objektumokat tárolunk. Közvetett módon ebben a tárolóban jelennek meg a régió szomszédságokat tartalmazó információk, amikre később szükségünk lesz.

Vonalakhoz tartozó segédinformációk tárolása

A testfelépítés során, amikor régióról régióra haladunk a frontvonalunk kiterjesztésével, ismernünk kell, hogy egy régió éleihez milyen másik régiók csatlakoznak. Amennyiben minden egyes régióélnél kereséseket kellene végrehajtanunk, hogy megtaláljuk a csatlakozó régiókat, az bizony elég sok erőforrást felemésztene. Előbbieket belátva a szomszédságokat érdemes előre feltérképezni.

9.1–2. ábra. A szakaszokhoz tartozó segédinformációkat.

```
192 struct LineAddon
193 {
194   std::vector<LineIndex>   sibling;
195   int ownregion;
196
197   LineAddon();
198 };
```

A bevezetett LineAddon nevezetű struktúrában (*9.1–2-es ábra*) el tudjuk tárolni egy szakaszon fekvő másik szakaszok indexeit és annak a régiónak az indexét, amely az adott szakaszt tartalmazza. Ha minden szakaszhoz létrehozunk egy az ismertetett struktúrából létrehozott példányt, akkor közvetetten megkapjuk a régiószomszédságokat tartalmazó információkat.

Ez után egy iterációs művelet következik, ami egészen addig ismétlődik, amíg van olyan régiónk, ami nem egy testhatároló felületéhez tartozik.

A ciklusmagban először megkeressük a következő kezdő régiót, amivel az új határolófelület felépítése megtörténhet.

A kezdőrégiót meg kell vizsgálnunk abból a szempontból, hogy helyileg egy részlegesen felépített test belsejében található-e. A bentlévőségre vonatkozó információt eltároljuk, hogy a később meghívott függvényeink ezt figyelembe vehessék.

Ezek után meghívjuk a testkontúr-felépítő függvényt, majd beállítjuk a határoló felületeket és a testek befoglaló téglatesteit. Ezután a ciklusmagban nincs több művelet.

Az utolsó művelet a testek tárolóvektorának feltöltése. Itt fontos megjegyeznünk, hogy a felépítő függvényünkön belül nem az FDSolid nevű struktúra felhasználásával tároltuk az adatokat, mivel erre a célra itt egy Solidoid nevű struktúrát vezettük be. Az FDSolid vektor feltöltése a Solidoid vektor által tárolt adatok alapján történik.

„Solidoid”

A Solidoid struktúra (*9.1–3-as ábra*) egy test adatainak tárolására lett megalkotva. A testműveleteket végző függvény olyan paramétereket használ, amikben egy helyen vannak letárolva a csomópontok, és egy helyen vannak letárolva a szakaszok. Mivel az FDSolid-nak saját pontjai és szakaszai vannak, problémás volna a köztes műveletekben ilyen formában tárolni a félkész testeket, mert folyamatosan váltogatnunk kellene a különböző tárolók között, másrészt sok esetben nem tudnánk kihasználni az egységes indexhivatkozások előnyeit.

9.1–3. ábra. A testek adatait csak részlegesen tároló struktúra definíciója.

```
225 struct Solidoid
226 {
227   std::vector<SolidoidContour>  contour;
228   BoundingCuboid                bounding_cuboid;
229
230   Solidoid();
231   void SetBoundingCuboid();
232 };
```

A Solidoid annyiban különbözik az FDSolid-tól, hogy nincsenek saját pontjai és szakaszai, a belső objektumai külső tárolókban levő adatok indexeit tárolják.

9.2. A régiók sorba rendezése

A régiórendező függvény (*9.2–1-es ábra*) felelős a régiók irányának beállításáért és a régiók sorbarendezéséért.

9.2–1. ábra A régiók sorbarendezéséért felelős függvény prototípusa.

```
561  void OrderRegions(
562    const std::vector<POINTW3> &node,
563    std::vector<FDLine> &line,
564    std::vector<FDRegion> &region,
565    std::vector<RegionIndex> &regionpile
566    );
```

A sorbarendező függvény a csomópontok, a szakaszok, a régiók és régió-indexek tárolóit használja bemeneti paraméterként. A sorbarendezés eredménye ténylegesen csak a régióindexeket tároló vektorban jelenik meg. A többi régió felépítéshez tartozó függvény ezen, az indexeket tároló regionpile nevű vektoron keresztül hivatkozik a régiókra.

A függvény első lépése a régiók normálvektorok által meghatározott módon történő oldalbeállítása, amit a SetRegionsByNormal(…) nevű függvény meghívásával érünk el.

A további lépésekben létrehozunk egy pregion vektort, ami egy RegionAddonA nevű segédstruktúra objektumait tárolja, majd feltöltjük a vektort egy iterációs művelet segítségével, amiben minden objektumra meghívjuk a konstruktorát.

A RegionAddonA nevű segédstruktúrát (*9.2–2. ábra*) azért vezettük be, mert szükségünk volt bizonyos adatok letárolására, amit csak ebben a függvényben használunk, és sok időt elvenne minden feltételvizsgálatnál az újraszámolásuk. További jelentős indok, hogy a segédstruktúránk egyszerűbb, mérete sokkal kisebb, mint az FDRegion-é, vagyis a vektoron belüli sorbarendezés kevesebb memória és processzorműveletet igényel.

9.2–2. ábra. A régiókhoz tartozó segédstruktúra definíciója.

```
200 struct RegionAddonA
201 {
202   NodeIndex                    min;
203   RegionIndex                  ownindex;
204   REAL                         nx_angle;
205   REAL                         ny_angle;
206
207   RegionAddonA(
208     const std::vector<POINTW3> &node,
209     const  std::vector<FDLine> &line,
210     const FDRegion             &region,
211     const int                  &index
212     );
213
214   void SetMinNode(
215     const std::vector<POINTW3> &node,
216     NodeIndex                  n
217     );
218 };
```

A RegionAddonA segédstruktúra tárolja az eredeti régió referenciapontját, indexét és a normálvektorának X és Y tengellyel bezárt szögét. Ezen változók értékét a konstruktorfüggvény állítja be.

Egymásba ágyazott ciklusok segítségével rendezzük a pregion vektorban található elemeket. A rendezés feltételei figyelembe veszik a régiók referenciapontjait, és a normálvektorok dőlésszögeit.

Végezetül feltöltjük a regionpile nevű vektort a sorbarendezett függvényekkel.

9.3. Testtagság vizsgálat

Az IsSolidPart() nevű függvény (*9.3–1. ábra*) hasonlóan van megoldva, mint a régiótagságot vizsgáló függvény, csak eggyel több dimenzióban. Ha egy régió egy (kész vagy félkész) testen belül található, akkor a függvény visszaadja a befoglaló test indexét, ellenkező esetben -1 értékkel tér vissza.

9.3–1. ábra. A testtagságot vizsgáló függvény prototípusa.

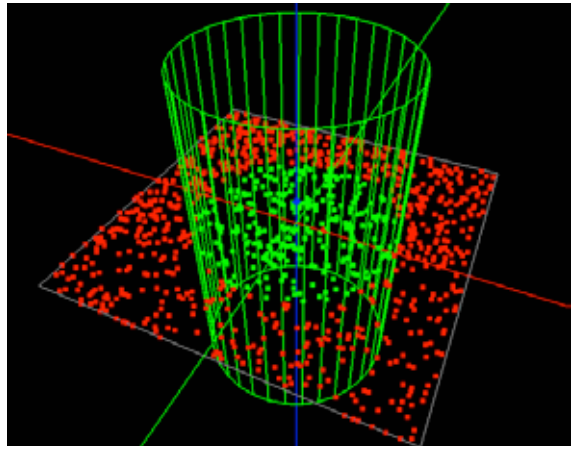
```
589  int IsSolidPart(
590    std::vector<POINTW3>          &node,
591    const std::vector<FDLine>     &line,
592    const std::vector<FDRegion>   &region,
593    const std::vector<Solidoid>   &prosolid,
594    const RegionIndex             &observedregion
595    );
```

Az első vizsgálatnál azt nézzük meg a függvényben, hogy létezik-e már (kész vagy félkész) test. Ha nem, akkor a függvény -1 értékkel tér vissza.

További lépésekben egy ciklusművelet segítségével minden létező (kész vagy félkész) testre vizsgálatokat folytatunk. Először megnézzük, hogy a vizsgált régió befoglaló téglateste benne van-e az adott régió befoglaló téglatestében. Ha a feltétel teljesül, akkor a SearchSolidCuts(…) nevű függvény segítségével megszámoljuk, hogy a régiónk egy véletlenszerű pontjából húzott végtelen félegyenes hányszor metszi az adott test határolófelületeit. Az első olyan test esetében, ahol a metszések száma páratlan, a függvény a test indexével tér vissza.

A lenti ábrán (*9.3–2. ábra*) egy pont testben lévőségéért felelős kódrészlet tesztelése látható. A programnak ezer darab véletlenszerűen elhelyezett pontot kell aszerint kiszíneznie, hogy a hengeren belül, vagy kívül található. Ugyanilyen elven működő kód vizsgálja, hogy a régió egy belső pontja egy adott testen belül található-e.

9.3–2. ábra. A pontok bentlevőségi vizsgálatának tesztelése ezer darab pontra.



Amennyiben a ciklusmagban meghatározott feltétel szerint egyetlen olyan testet sem találunk, ami a régiónkat körülvenné, a függvény -1 értékkel tér vissza.

9.4. Testmetszések keresése

A SearchSolidCuts(...) nevű függvény (*9.4–1. ábra*) feladata megvizsgálni, hogy egy régió véletlenszerűen kiválasztott belső pontjából húzott végtelen félegyenes hányszor metszi egy test határolófelületét. Három dimenzióban a bent lévőség vizsgálata jóval bonyolultabb, mint két dimenzióban.

9.4–1. ábra. A testmetszéseket számoló függvény prototípusa.

```
597  int SearchSolidCuts(
598    std::vector<POINTW3>          &node,
599    const std::vector<FDLine>     &line,
600    const std::vector<FDRegion>   &region,
601    const Solidoid                &prosolid,
602    const RegionIndex             &observedregion
603    );
```

Mivel a végtelen félegyenesünk párhuzamos az X tengellyel és mínusz X irányába végtelen, ezért előszűrést végezhetünk a test határolórégiói között arra vonatkozóan, hogy szóba jöhetnek-e az egyes régiók a metszésvizsgálatok során. Kigyűjtjük azokat a testhatároló régiókat, amelyek befoglaló téglatesteinek vetülete átfedésben van a vizsgált régiónk befoglaló téglatestének vetületével az Y és Z koordinátatengelyek által kifeszített síkon. Ezzel nagymértékben csökkentettük a későbbi bonyolultabb vizsgálataink számát.

A következő fontos lépés a vizsgált régiónk egy belső pontjának kiválasztása, amit a RandomPointInRegion(…) nevű függvény ismételt hívogatásával addig generáltatunk, amíg nem kapunk olyan pontot, aminek a vetülete nem esik rá a potenciális befoglaló test szóba jöhető határoló-régiói pontjainak és szakaszainak vetületére. Ezzel garantáljuk az olyan szituációk elkerülését, ahol a végtelen félegyenesünk a test sarokpontjain, vagy a régiók találkozásánál lévő élen, vagy a régiók felületén haladna át, amik zavart keltenének a metszések számlálásában.

Miután megvan a belső pontunk, vetítjük az összes csomópontot a Z és Y koordinátatengelyek által kifeszített síkra. A kiválogatott határoló-régióink közül megvizsgáljuk, hogy melyik régió vetületén belül található a félegyenesünk vetített végpontja (a 7.3 alfejezetben bemutatott függvény segítségével). Amennyiben a bentlévőség fenn áll és a félegyenes végpontja az X tengelyen a határoló régió síkjában, vagy afölött található, akkor a régiómetszés esetével van dolgunk. Az ilyen régiómetszéseket összeszámoljuk és a végösszeg képezi a függvény visszatérő értékét.

9.5. A testkontúr keresése

A SearchSolidoidContour(…) nevű függvény (*9.5–1. ábra*) feladata, hogy egy testkontúrt építsen fel egy rendezett régiósereg egymáshoz csatlakozó régióiból.

9.5–1. ábra. A testkontúr kereső függvény prototípusa.

```
617  void SearchSolidoidContour(
618    const std::vector<POINTW3>     &node,
619    std::vector<FDLine>            &line,
620    const std::vector<LineAddon>   &pline,
621    std::vector<FDRegion>          &region,
622    std::vector<RegionIndex>       &regionpile,
623    std::vector<bool>              &solidpart,
624    const int                      &startingregion,
625    const bool                     &isholeelement,
626    std::vector<RegionIndex>       &contour_regions
627    );
```

Amennyiben a kezdő régiónk belső testkontúrhoz tartozik, kezdőlépésként meg kell fordítanunk a régió körüljárási irányát és a normálvektorának inverzét kell képeznünk. Ezeket a műveleteket a RegionInvert(...) nevű függvény meghívásával hajtjuk végre.

A kezdőrégiót betesszük az új régiókontúrt tároló objektum régiói közé, majd deklaráljuk a frontvonalszakaszok indexeit tároló vektort és megadjuk a kezdő indexeket.

A következő lépésben egy ciklusművelet jön, ami addig tart, amíg a frontvonal vektorunk tartalmaz indexeket.

A ciklusmagban mindig egyetlen frontvonalszakaszt vizsgálunk, aszerint, hogy melyik másik régiók élei csatlakoznak hozzá, és hogy ezek közül melyik élhez tartozó régióval kell folytatnunk a kontúr építését. Ezt a GetNextRegionsConnectedLine(...) függvény dönti el nekünk, ami egy szakaszindexet ad vissza nekünk.

A megtalált szakasz régióját ha kell invertáljuk aszerint, hogy hogyan illeszkedik helyesen a szomszédos kontúrelemekhez (3.1-es alfejezetben leírt módon, *lásd: Dunakavics 2014/9 52. old.*), majd beillesztjük a kontúrrégiók közé.

A beillesztés után meghívjuk az ExpandFrontLines(...) nevű függvényt, ami korrigálja a frontvonalunkat az új régió kontúrvonalai szerint. Itt a ciklusmag véget ér.

9.6. Frontvonal kiterjesztése

Az ExpandFrontLines(...) nevű függvény (*9.6–1. ábra*) kezeli a frontvonal-terjeszkedést, csökkenést, amikor egy új régióval bővül az eddig meghódított felületünk. Ilyenkor az új régió éleit hozzá kell adni a frontvonalhoz, a meghódított területen belülre került éleket pedig meg kell szüntetni.

9.6–1. ábra. A frontvonal kiterjesztő kereső függvény prototípusa.

```
629  void ExpandFrontLines(
630    const std::vector<FDLine>      &line,
631    const std::vector<LineAddon>   &pline,
632    const FDRegion                 &region,
633    std::vector<int>               &frontlines
634    );
```

Első lépésben egy szimpla ciklusművelet segítségével a régió éleinek indexeit berakjuk a frontvonal-indexeket tartalmazó vektorba.

Második lépésben megnézzük, hogy keletkeztek-e duplikációk és ha igen, akkor azokat kivesszük. Bevezetünk egy segédvektort és iteratívan végignézzük, hogy két frontvonalszakasz egymásra került-e. Csak azon szakaszok indexeit rakjuk be a segédvektorba, amelyeken az előbbi feltétel nem teljesül. Végül a segédvektor tartalmával helyettesítjük az eredeti frontvonal-indexeket tároló vektorunk tartalmát.

9.7. Csatlakozó régió kiválasztása

Amikor egy testkontúr építésénél egy adott frontvonalszakaszhoz több régió éle is illeszkedik, akkor el kell tudnunk dönteni, hogy melyik régióval folytassuk a terjeszkedést. Mivel a szakaszok szomszédságait már korábban feltérképeztük, és minden szakaszról tudjuk, hogy melyik régió alkotóeleme, ezért itt tulajdonképpen elég visszaadnunk a megfelelő régió csatlakozó szakaszának indexét. Ezt a GetNextRegionConnectedLine(…) nevű függvény (*9.7–1. ábra*) teszi meg nekünk.

9.7–1. ábra. A régió kiválasztó függvény prototípusa.

```
637  int GetNextRegionsConnectedLine(
638    const std::vector<POINTW3>     &node,
639    std::vector<FDLine>            &line,
640    const std::vector<LineAddon>   &pline,
641    std::vector<FDRegion>          &region,
642    std::vector<RegionIndex>       &regionpile,
643    std::vector<bool>              &solidpart,
644    const int                      &actualfrontline
645    );
```

Elsőnek megnézzük, hogy a vizsgált frontvonalszakaszra hány másik szakasz illeszkedik. Ha csak egy, akkor a visszatérési értékünk annak a szakasznak az indexe, ha több, akkor tovább folytatjuk a vizsgálatot.

Következő lépésben egy ciklusműveleten keresztül megvizsgáljuk a csatlakozó élek régióit. Csak azokat a régiókat vizsgáljuk ezek közül, amik még nem alkotnak kontúrt. A vizsgálat itt abból áll, hogy összehasonlítjuk minden csatlakozó régió normálvektorának a frontvonal mögött lévő régió normálvektorával bezárt szögét. A legnagyobb szöghöz tartozó régió élének indexe lesz a visszatérő érték.

9.8. Régió belső pontjának kiválasztása

A régió belső pontjának kiválasztásához a RandomPointInRegion(…) nevű függvényt (*9.8–1. ábra*) használjuk.

9.8–1. ábra. Egy régió egy belső pontját visszaadó függvény prototípusa

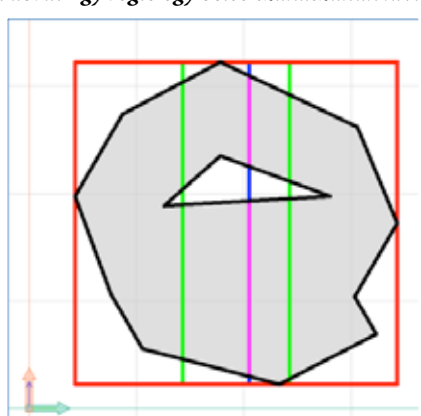
```
POINTW3 RandomPointInRegion(
  ProjectedNodes                &pnode,
  const std::vector<FDLine>     &line,
  const FDRegion                &region
  );
```

Első lépésben vetítjük a régiónkat, mivel síkban jóval kevesebb műveletet igényel a belsőpont-kiszámítás. Továbbiakban a vetületen X és Y koordinátákkal dolgozunk.

A kiválasztás koncepciója

Amikor a régiónk egy belső pontját kiválasztjuk, fontos figyelembe vennünk, hogy a kiválasztott pontunkat mire akarjuk használni. Mivel ez a belső pontunk egy végtelen félegyenesünk végpontja lesz, el akarjuk majd kerülni az olyan szituációkat, ahol a pontunk, vagy a pont által meghatározott félegyenes, egybeesik más testek és régiók síkjaival, éleivel és sarokpontjaival térben és síkban egyaránt. Ha véletlenszerűen választunk belső pontot és biztosítjuk, hogy a belső pontunk a régió éleitől lehetőleg távol essen, akkor nagy valószínűséggel meg fogunk felelni az előbbi elvárásainknak.

9.8–2. ábra. Egy régió egy belső szakaszának kiválasztása.



A programban a belső pont keresését egy vetületen végezzük, majd a talált kétdimenziós belső pontot visszavetítjük térbe a régió síkjának egyenlete alapján.
Vesszük a vetített régiónk befoglaló téglalapjának (*9.8–2. ábrán* a piros téglalap) X koordináták szerinti középső harmadát (*9.8–2. ábrán* a két világoszöld vonal közötti rész). Ezen a középső harmadon belül állítunk fel véletlenszerűen egy egyenest, ami párhuzamos az Y tengellyel (*9.8–2. ábrán* a kék-lila egyenes). Az egyenesünkkel elmetsszük a régiónk kontúrjait. A metszéspontok segítségével kontúrtól kontúrig tartó belső szakaszokat (*9.8–2. ábrán* a lila szakaszok) hozunk létre és ezek közül kiválasztjuk a leghosszabbat. A régiónk belső pontja ezen leghosszabb szakasz középső harmadának egy véletlenszerűen kiválasztott pontja.

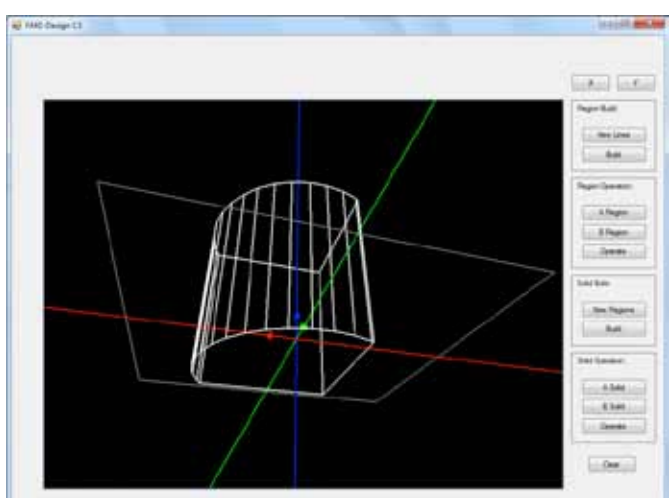
A függvény további műveleteit a kiválasztás koncepciója szerint valósítjuk meg.

10. A testműveletekért felelős függvények bemutatása

Ebben a fejezetben a 3.2-es alfejezetben ismertetett *(lásd: Dunakavics 2014/9 56. old.)* testműveletek programkódbeli leképezését mutatjuk be. Az ismertetés átfogó, de nem túlzottan részletes. A dolgozat terjedelmi korlátozásai miatt csak a fontosabb műveletek és utasításokat ismertetjük.

Az alábbi ábrán megtekinthető a testműveleteket végző függvények által létrehozott testmetszet.

10–1. ábra. Egy hengeren és egy kockán elvégzett közös rész művelet eredménye.



10.1. Testműveletek

A testműveleteket egy SolidOperation(...) nevű függvény (*10.1–1-es ábra*) segítségével végezhetjük el. Ez egy központi függvény, ami további belső függvényhívásokat végez, az egyes részműveletekhez.

10.1–1. ábra. A testműveleteket végző központi függvény prototípusa.

```
662 void SolidOperator(
663   FDSolid              &solid1,
664   FDSolid              &solid2,
665   SolidOperation       operation,
666   std::vector<FDSolid> &issue
667   );
```

A paramétermezőben az első két paraméter a két test tulajdonságait tartalmazó változóé, amiken a műveletet elvégezzük. A harmadik paraméter egy olyan enumerátor típusváltozója, amiben a lehetséges műveletek neveit definiáltuk. A negyedik paraméter egy eredménytesteket tároló vektor.

A függvénytörzsben első lépésben megvizsgáljuk, hogy a kapott testek befoglaló téglatestei átfedik-e egymást. Ezt az IsBoundingCuboidsOverlaps(...) nevű függvény mondja meg nekünk. Amennyiben a befoglalók között nincs átfedés, nem kell elvégeznünk a metszésvizsgálatokat és metszéseket. A műveletek eredményét az eredeti testekkel – új eredménytest létrehozása nélkül – is meg tudjuk adni

Második lépésben testeink tulajdonságait SolidRemains nevű struktúrák objektumaiban tároljuk el. Ezután meghívjuk a SignRegionIntersections(...) nevű függvényt, ami létrehozza a testek közti összemetsződések bemetszett szakaszait és eltárolja azokat az említett objektumokban.

Testmaradványok tárolása

A SolidRemains nevű struktúra (*10.1–2-es ábra*) azért lett megalkotva, hogy megkönnyítse az eredeti testek régióinak bejárását, kezelését és a metszésekkel kapcsolatos műveletek eredményeinek tárolását.

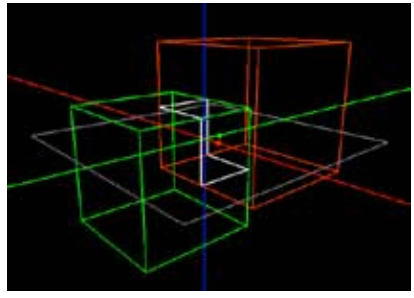
10.1–2. ábra. A testmaradványokat tároló struktúra definíciója.

```
292 struct SolidRemains
293 {
294   std::vector<POINTW3>          node;
295   std::vector<FDLine>           line;
296
297   std::vector<RegionRemains>    regionremains;
298
299   std::vector<FDRegion*>        regionshards;
300
301   //====================
302   SolidRemains();
303   SolidRemains(FDSolid &solid);
304 };
```

Mivel a testek összemetsződésének hatására új csomópontok és szakaszok jönnek létre, a testmaradványokat tartalmazó struktúra már saját pontokat és szakaszokat tároló vektorokkal rendelkezik. A regionremains nevű vektorban lévő objektumok az eredeti régiók mutatóit, a metszések szakaszait és az elmetsződés után létrejövő régiókat tárolják. A regionremains nevű vektor pedig az eredményrégiók mutatóit tárolja.

Most hogy már a metszéseket jelölő szakaszokat (*10.1–3-as ábra*) előállítottuk, elmetszhetjük velük a régiókat mindkét testben és az előállt szétdarabolt szakaszseregekből új régiókat építhetünk. Ezeket a műveleteket a BuildRegionShards(…) nevű függvény hajtja végre.

10.1–3. ábra. A metszéseket jelölő szakaszokat előállító függvény tesztelése. A bemetszéseket fehér színnel jelöltük.



A következő állomás az újonnan elkészült régiók közül kiválogatni azokat, amik az eredménytestet képezni fogják. A művelettípus szerinti kiválogatás kiválasztása a paraméterként kapott enumerátor alapján történik. Programban egyelőre csak a közös részműveletre vonatkozó kiválogatás lett megvalósítva, amit az FDSolidIntersection(…) nevű függvény hajt végre.

A kiválogatott régióinkból már csak fel kell építenünk az eredménytesteket. Itt az előző fejezetben már ismertetett SearchSolids(…) nevű függvény kerül meghívásra, majd a függvény befejezi futását.

10.2. Régiók összemetsződéseinek jelölése

A régiók valamennyi összemetsződésének kiszámításáért a SignRegionIntersections(…) (*10.2–1-es ábra*) valamint a SignIntersectedLinesInTwoRegions(…) (*10.2–2-es ábra*) nevű függvények felelnek.

10.2–1. ábra. A testek összemetsződéseit jelölő függvény prototípusa.

```
667  void SignRegionIntersections(
668    SolidRemains          &remains1,
669    SolidRemains          &remains2
670    );
```

Amikor meghívjuk a SignRegionIntersections(…) függvényt, az két egymásba ágyazott ciklus segítségével párba állítja a két eredeti test összes régióját, majd minden párosra megvizsgáltatja az összemetsződéseket és letároltatja a bemetszett szakaszokat a SignIntersectedLinesInTwoRegions(…) nevű függvény segítségével.

10.2–2. ábra. A két régió közötti összemetsződést jelölő függvény prototípusa.

```
672  void SignIntersectedLinesInTwoRegions(
673    SolidRemains          &remains1,
674    SolidRemains          &remains2,
675    RegionRemains         &rem1,
676    RegionRemains         &rem2
677    );
```

Az egyes régiók közötti metszésvizsgálat első lépése, hogy megvizsgáljuk a régióink párhuzamosak-e. Csak akkor tudjuk folytatni a műveleteket, ha a párhuzamosság nem áll fenn.

Következő lépésben kiszámítjuk a régiók normálvektorából a metszésvonaluk irányvektorát egy szimpla vektoriális szorzás művelettel, majd egy egyenletrendszeren keresztül meghatározzuk a régiók síkjának egy közös pontját. Innentől rendelkezünk a metszésvonal minden fontos tulajdonságával.

A metszésvonalunkkal elmetsszük a két régiónk minden szakaszát és a metszéspontokat egy halomban gyűjtjük. A metszéspontokat az IntersectedPoints(…) nevű függvénnyel számoltatjuk ki. Mivel a pontokhoz használtunk burkolóosztály, amihez megírtuk a < operátor függvény, a halomba behelyezett pontok automatikusan rendezve lesznek.

A PushBackIntersections(…) függvényünk meghívásával a rendezett metszéspontok alapján létrejönnek a régiók közös bemetszései és a régiók osztályában tárolásra kerülnek.

10.3. A bemetszések megalkotása

A régiók közötti metszéseket jelölő szakaszok tényleges megalkotásáért a PushBackIntersections(…) nevű függvény felel.

10.3–1. ábra. A metszéseket jelölő függvény prototípusa.

```
687  void PushBackIntersections(
688    SolidRemains          &remains1,
689    SolidRemains          &remains2,
690    RegionRemains         &rem1,
691    RegionRemains         &rem2,
692    std::set<SetPoint>    &linecuts
693    );
```

Mivel a metszéspontjainkat egy sorbarendezett halomban tároltuk le, könnyű dolgunk van. Egy ciklus segítségével sorban bejárjuk a pontjainkat és minden egymás után következő két pontot úgy értelmezünk, mintha egy szakasz két végpontját alkotnák. A szakaszaink felezőpontjára ellenőrzéseket futtatunk le.

Egymásba ágyazott ha-elágazásokon keresztül megvizsgáljuk, hogy a potenciális metszésszakaszunk felezőpontja hol található. Amennyiben a felező mindkét régió belsejében van, a szakaszunk mindkét régióhoz tartozó objektumban tárolásra kerül, mint bemetszést jelölő szakasz. Ha a felezőpont az egyik régió

belsejében, de a másik régió kontúrján található, akkor csak a befoglaló régió objektumában lesz letárolva. Ha se kontúron, se belső pontként nem szerepel a felezőnk, akkor nem foglalkozunk a szakasszal.

10.4. Régiószilánkok felépítése

Miután a bemetszéseket jelölő szakaszokat megalkottuk, nincs más hátra, mint ténylegesen is elmetszeni a régiókat és a szétmetszett szakasz halomból új régiókat építeni. Ez a BuildRegionShards(…) nevű függvényen belül, illetve a CreateShardsFromRegionRemain(…) függvényben (*10.4–1. ábra*) történik meg.

A BuildRegionShards(…) függvény egy iterációs művelet segítségével bejárja a testmaradványokat kezelő objektum régióit. A ciklusmagban amelyik régiónál bemetszést jelölő tárolt szakaszt találunk, ott meghívjuk a CreateShardsFromRegionRemain(…) függvényt, ami elvégzi a metszéseket és régiófelépítéseket, majd az újonnan keletkezett régiók mutatóit betölti egy vektorba. Ahol nincs metszés, ott az eredeti régió mutatóját töltjük be az említett vektorba. Így bár a régiók különböző helyeken vannak letárolva, mégis egy mutatóvektoron belül be tudjuk járni őket.

10.4–1. ábra. A régiók elmetszéséért és az új régiók felépítéséért felelős függvények prototípusai.

```
695  void BuildRegionShards(SolidRemains &remains);
696
697  void CreateShardsFromRegionRemain(
698    SolidRemains          &remains,
699    RegionRemains         &rrem
700    );
```

A CreateShardsFromRegionRemain(…) függvény első lépéseként meghívjuk a metszések utáni szakaszsereg előállításáért felelős függvényt.

Második lépésben a szakaszseregünkre meghívjuk a 7. fejezetben ismertetett régió felépítő függvényt.

Harmadik lépésben leellenőrizzük, hogy az eredményrégiók normálvektora nem inverze-e az eredeti régió normálvektorának. Ha igen, akkor az eredményrégiók inverzét képezzük.

Utolsó lépésben az eredményrégiók mutatóit egy vektorban tároljuk.

10.5. Régió szakaszainak elmetszése

A CreateLinePileForRegionRemains(...) függvény (*10.5–1. ábra*) a régiót metsző szakaszokkal elmetszi a régió szakaszait, majd az eredményszakaszokat és a metsző szakaszokat tárolja.

10.5–1. ábra. A szétmetszett régiókat előállító függvény prototípusa.

```
702  void CreateLinePileForRegionRemains(
703    SolidRemains          &remains,
704    RegionRemains         &rrem
705    );
```

A függvény első fontos állomása a régió kontúrszakaszainak elmetszése. A kontúrszakaszokat két egymásba ágyazott ciklus segítségével járjuk be. Minden szakasznál egy újabb ciklusművelet segítségével megvizsgáljuk, hogy melyik metszőszakaszok végpontjai esnek rá. A szakaszra eső végpontokat külön vektorban gyűjtjük. A gyűjtés után, ha a vektorunk üres maradt, a szakaszunk indexét letároljuk a rrem objektum linepile nevű vektorában. Ha a vektorunk mérete nagyobb, mint nulla, akkor meghívjuk a LineFragmentation(...) nevű függvényt, ami széttördeli a szakaszunkat a metszéspontok szerint; az eredményszakaszokat tárolja, majd menti azok indexeit.

Második fontos állomás a régiót metsző szakaszok tárolása. Egy ciklusműveleten keresztül végighaladunk ezeken a szakaszokon. Újabb ciklusok segítségével megnézzük, hogy az aktuális metsző szakaszunk végpontjai léteznek-e már a letárolt szakaszok közül valamelyik végpontjaként. Ha elágazások segítségével eltároljuk az új pontokat és mentjük a szakaszainkat. Amelyik végpont még nem létezett, azt külön mentenünk kell, az indexét csak ezután adhatjuk meg a tárolni kívánt szakaszoknak. Fontos megjegyezni, hogy ezeket a metszésszakaszokat duplán kell letárolnunk, hogy a régiófelépítő-függvényeink később működhessenek.

10.6. Szakasztördelés

A szakasztördelő függvény (*10.6–1. ábra*) egy szakasz és a rajta lévő metszéspontok alapján állít elő új szakaszokat.

10.6–1. ábra. A szakasztördelő függvény prototípusa.

```
707  void LineFragmentation(
708    SolidRemains          &remains,
709    RegionRemains         &rrem,
710    LineIndex             &lineindex,
711    std::vector<POINTW3> &newnodes
712    );
```

Elsőként egymásba ágyazott ciklusok segítségével sorba rendezzük a metszéspontokat. A sorrendet a metszeni kívánt szakasz a pontjától (első pont, vagy referencia pont) való növekvő távolság határozza meg.

Második lépésként egy újabb ciklusműveletben eltároljuk sorrendben az új pontokat, a pontokra illeszkedő szakaszokat és azok indexeit.

10.7. Régióválogatás műveletek szerint

Az FDSolidIntersection(…) nevű függvény (*10.7–1. ábra*) válogatja ki azokat az eredményrégiókat, amelyek a testek közötti unió, kivonás és közös részműveletek szerint szükségesek. Ezekből az eredményrégiókból épülnek fel majd az adott művelet eredménytestei. A kizáró-vagy műveletet az FDSolidExcludedOR(…) nevű függvény valósítja meg, ami az előbb említett függvény egyszerűsített változata, mivel itt nem kell vizsgálnunk a régiók bentlévőségeit.

10.7–1. ábra. A „ÉS” műveletet régióválogató függvényének prototípusa.

```
721  void FDSolidIntersection(
722    FDSolid               &solid1,
723    FDSolid               &solid2,
724    SolidRemains          &remains1,
725    SolidRemains          &remains2,
726    std::vector<POINTW3>  &node,
727    std::vector<FDLine>   &line,
728    std::vector<FDRegion> &region
729    );
```

Az eredményrégiók kiválogatásánál kritikus problémának minősül a régiók pontjainak letárolása pontismétlődés elkerülése szempontjából. A probléma kezeléséhez egy NodeAdministrator nevű struktúrát vezetünk be és a függvényben példányosítjuk.

„NodeAdministrator”

A NodeAdministrator nevű struktúra azért lett megalkotva, hogy megkönnyítse a csomópontok tárolásánál keletkező ismétlődések kiszűrését.

10.7–2. ábra. A csomópont ismétlődés szűrését segítő struktúra definíciója.

```
347 struct NodeAdministrator
348 {
349   std::set<SetPoint_p> snode;
350   std::vector<POINTW3> node;
351
352   NodeAdministrator();
353   NodeAdministrator(int s);
354
355   int Pushback(POINTW3 &n);
356 };
```

A struktúra a csomópontokat saját vektorában tárolja. Ezenkívül rendelkezik egy halom (std::set<t>) típusú tárolóval, ami egy olyan struktúra példányait tárolja, amik rendelkeznek az egyes hozzájuk tartozó csomópontok mutatóival és indexével, valamint rendelkeznek < operátorfüggvénnyel. A halomtárolóban ezek az objektumok automatikusan sorba rendeződnek, elem ismétlődések pedig eleve nem jöhetnek létre a sablonosztályban definiáltak szerint.

A tárolóink kezelését a NodeAdministrator::Pushback(…) függvény végzi, ami egyszerűen csak azt a csomópontot tárolja el a vektorban, amihez tárolni lehetett a hozzá tartozó objektumot a halomban, majd visszaadja a letárolt csomópont indexét. Ha már létezett a csomópont, akkor annak az indexét adja vissza, ha most került tárolásra, akkor az utolsó elem indexét adja vissza.

A következőkben ciklusműveleteken keresztül ellenőrzéseket folytatunk. Először leellenőrizzük, hogy az első testből metszések után keletkezett régiók a másik test belsejében vannak-e. Másodjára azt ellenőrizzük, hogy a második testből metszések után keletkezett régiók az első test belsejében vannak-e.

Ahol a bentlévőség – a kért művelet szerint – teljesül, ott az aktuális régiót az eredményrégiók vektorában tároljuk, és a hozzá tartozó szakaszokat és pontokat is tároljuk a RegionJoinToSolidComponents(…) nevű függvény segítségével (Ez a függvény használja a korábban említett NodeAdministrator típusú objektumot is).

A ciklusaink után már csak a node nevű vektorunknak kell átadnunk a NodeAdministrator típusú objektumban tárolt csomópontokat.

10.8. Eredményrégió mentése

A fejezet utolsó ismertetett függvénye a RegionJoinToSolidComponents(…) nevet viseli (*10.8–1. ábra*). Mivel a kiválogatott eredményrégiókat, valamint azok szakaszait és csomópontjait közös tárolókban gyűjtjük ez a függvény gondoskodik a régiók és hozzájuk tartozó adatok megfelelő tárolásáról.

10.8–1. ábra. A régiók, szakaszok és csomópontok megfelelő tárolásáról gondoskodó függvény.

```
715  void RegionJoinToSolidComponents(
716    SolidRemains          &remains,
717    FDRegion              &comer,
718    NodeAdministrator     &na,
719    std::vector<FDLine>   &line,
720    std::vector<FDRegion> &region
721    );
```

Első lépésként a paraméterként kapott régiót szimplán behelyezzük az eredményrégiók tárolójába.

Egymásba ágyazott ciklusok segítségével bejárjuk a régiómaradvány és az újonnan tárolt régió adattagjait és eltároljuk a szükséges szakaszokat és csomópontokat, majd azok indexeit.

11. Összehasonlítás és mérési eredmények

Ebben a fejezetben tárgyaljuk a dolgozat keretei között elvégzett munka eredményeit, illetve az új és a régi testműveleteket végző függvények közötti különbségeket.

11.1. A sebességkülönbség

A dolgozatban kidolgozott eljárások és függvények segítségével sikerült igen nagy sebességnövekedést elérni a FEM-Design-ban lévő programfüggvényekhez képest. A következőkben bemutatásra kerülnek a mérési eredmények.

A tesztelések a testek közötti „ÉS” művelet megvalósításával történtek. Sajnos a mérési eredmények nem kifejezetten objektívek, mivel a FEM–Design-ban nehéz egzakt eredményeket nyújtó teszteket futtatni. A tesztek során fontos szempont volt, hogy a két függvényt hasonló alakú és bonyolultságú testekkel tegyük próbára, a szélsőséges eseteket kizártuk.

A számítási idő mérése úgy történt, hogy az egyes függvényhívások előtt és a függvény lefutása után mért aktuális idők különbségét számítottuk. A FEM–Design Solid_And_Solid(...) műveletének futási idejét az alábbi kód segítségével mértük le.

11.1–1. ábra. A FEM–Design-ban megírt kódrészlet, ami az eddigi testek közötti „ÉS” műveletet megvalósító függvény futási idejét méri és kiíratja.

```
344        //=========================================
345        //Solid_And_Solid(...) time test:
346
347        st = GetTickCount();
348        Solid_And_Solid(hesRgn, hesSec, hftemp);
349        et = GetTickCount();
350
351        {
352          SZBUF sz;
353
354          sprintf(sz, "%.3f", (et - st) / 1000.0);
355          DskMsg_Debug(sz);
356        }
357        //=========================================
```

A dolgozatban kidolgozott testműveletet végző függvény tesztelése már több nehézségbe ütközött. Mivel az 5. fejezetben ismertetett grafikus felület a megjelenítéshez részben a .NET-hez tartozó System osztály függvényeit használja, ezért a fordító a programkódot nem gépi kódra fordította le, hanem a .NET által használt köztes nyelvre. Mivel a köztes nyelvű kódot a program futtatásakor a .NET virtuális gépének gépi kódra kell fordítania, ezért ez jelentősen növeli a program futási idejét. Ennek kivédésére a fontosabb füg-

gvényeket egy konzolos felületű program kódjába ágyaztuk be és gépi kódra fordítottuk, hogy így pontosabb méréseket végezhessünk. Az új testműveletet végző függvény futási idejét az alábbi kód segítségével mértük.

11.1–2. ábra. Az új testműveleteket végző függvény futási idejének méréséért felelős kódrészlet.

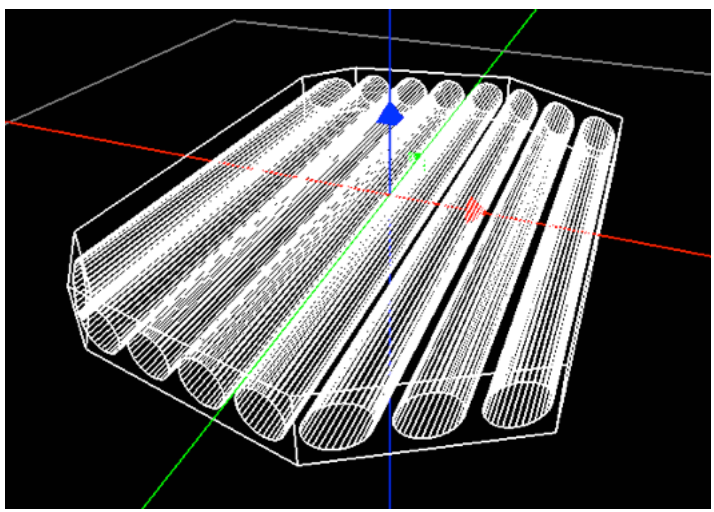
```
31      std::clock_t a = std::clock();
32
33      SolidOperator(solid1, solid2, Intersection, solids);
34
35      std::clock_t b = std::clock();
36
37      t+=(b-a);
38
39      std::cout << "time:\t" << b-a << std::endl;
```

A dolgozat elkészítése során rengeteg mérést végeztünk, az egyes munkafázisok befejezése után. Szélsőséges esetekben előfordult, hogy csak 30-szoros sebességkülönbség volt az új és a régi testműveletet végző függvények között, de előfordult a 170-szeres sebességkülönbség is. Az alábbi táblázatban olyan műveletek számítási idejét láthatjuk, ahol hasonló bonyolultságú panelelemeket elmetszettük kilencszög alapú hasábokkal (*11.1–3. ábra*).

	Solid_And_Solid(...) (sec)	SolidOperator(...) (sec)
1. mérés:	1,544	0,015
2. mérés:	1,701	0,013
3. mérés:	1,56	0,014
4. mérés:	1,482	0,014
5. mérés:	1,466	0,014
6. mérés:	1,731	0,013
7. mérés:	1,872	0,013
8. mérés:	2,106	0,013
9. mérés:	1,669	0,013
10. mérés:	1,482	0,014
Átlag:	1,6613	0,0136

Ha a méréseket átlagoljuk és összehasonlítjuk (1,6613 sec / 0,0136 sec = 122,1544), elmondhatjuk, hogy a dolgozatban kidolgozott függvény futási ideje hasonló körülmények között nagyjából 120-szor rövidebb, mint a FEM–Design-ban lévő függvénynek. Következtetésképpen: a dolgozat célja – a legalább tízszeres sebességnövekedés produkálása – többszörösen túlteljesítettük. Fontos kihangsúlyozni, hogy teljesen objektív mérési eredmények, csak akkor lesznek elérhetőek, ha az új kódot beépítjük a FEM–Design-ba. Mindenesetre az eddigi méréseink is jól szemléltetik a végső fázis várható arányait.

11.1–3. ábra. A lefuttatott műveleti tesztek által generált eredmény test.



11.2. Hiányosságok

A dolgozatban kidolgozott függvények még több szempontból nem tökéletesek:
1. A testműveletek során esetlegesen létrejövő felületi érintkezések lekezelése még folyamatban van.
2. Mivel az előregyártott panelelemek létrehozásánál nem használunk olyan testeket, ahol fennállna az önérintkezés lehetősége, ezért a függvényeket az ilyen szituációkra nem készítettük fel. Nyilván ezt a hiányosságot előbb-utóbb pótolni kell, ha a testműveletet megvalósító függvényt szélesebb körben kívánjuk alkalmazni.
3. Az esetleges bemenő adatokban lévő hibák kezelése még nincs megoldva. A műveletek végzésekor eddig feltételeztük, hogy a paraméterben kapott adataink hibátlanok.

12. A projekt jövője és lehetőségei

A dolgozatban kidolgozott eljárások és függvények a jövőben még számos átalakításon és tökéletesítésen fognak átesni, mire végleges formájukban beépülnek a FEM–Design programrendszerbe.

12.1. A matematikai függvények optimalizálása

A dolgozatban megírt matematikai segédfüggvények többsége rögtönzött és általános matematikai és geometriai ismeretekre támaszkodik. A jövőben szükség lesz matematikus végzettségű kollégák szakértelmét igénybe venni az optimálisabb függvénymegoldások érdekében.

12.2. Párhuzamosítás

Az függvényeink jelenleg nem képesek kihasználni a több processzorral rendelkező számítógépek által nyújtott lehetőségeket. A jövőben a C++ nyelv által támogatott Open MP (Open Multiprocessing) alkalmazásprogramozási interfész lehetőségeit is megkíséreljük igénybe venni.

12.3. Hiányosságok kiküszöbölése

Az előző fejezetben említett hiányosságok még megoldásra várnak. Az elképzelések szerint ezek a problémák kezelhetőek úgy, hogy ne jelentsenek sebességcsökkenést a futási idő szempontjából.

Felhasznált irodalom

A dolgozat döntő többségében Dr. Kirchner István – Végeselemes eljárások hatékonyságának növelése című PhD értekezésére támaszkodik és az abban tárgyalt eljárásokat valósítja meg.

A további szakirodalmak nagy segítséget nyújtottak a dolgozatban kidolgozott programfüggvények létrehozásában:

– Bjarne Stroustrup – A C++ programozási nyelv

– Bancsik Zsolt, Lajos Sándor, Juhász Imre – Ábrázoló geometria kezdőknek

– http://msdn.microsoft.com/en-us/library/system.aspx

– http://patakino.web.elte.hu/Halado_C++/HaladoSTL.pdf

– http://www.codeproject.com/Articles/16051/Creating-an-OpenGL-view-on-a-Windows-Form

– http://www.songho.ca/opengl/gl_transform.html

– http://falloutsoftware.com/tutorials/gl/gl2p5.htm

– http://msdn.microsoft.com/en-us/library/ms171538.aspx#Y0

– http://en.wikipedia.org/wiki/Map_(C%2B%2B)

– http://courses.cms.caltech.edu/cs11/material/cpp/donnie/cpp-ops.html

– http://www.functionx.com/cpp/keywords/typedef.htm

KLUCSIK GÁBOR *

A Dunaújvárosi Főiskolán alkalmazott online oktatás vizsgálata

* *Dunaújvárosi Főiskola, Társadalomtudományi Intézet*
E-mail: klucsikg@mail.duf.hu

Összefoglalás: A Dunaújvárosi Főiskolán pilot projektként bevezetésre került 5 tárgyból a teljesen videóval támogatott oktatási forma, amely a Moodle -rendszeren alapul. A megjelenítés felülete és a hallgatók online irányítása teljesen egyéni fejlesztésben valósult meg, publikációmban először ezt ismertetem. A hallgatók online oktatáshoz való attitűdjét empirikus kutatással vizsgáltam 2012-ben és 2013-ban, ezen kutatásom eredményeit ismertetem.
Kulcsszavak: Online oktatás, hallgatók, attitűd.

Abstract: Online teaching has been introduced at the College Dunaujvaros as a pilot project. The unique feature of our online teaching program is that all the learning materials are available as a short video. In the beginning the students could select from 5 subjects to learn online. Our online learning environment is based on the Moodle system. In the first part of my publication I describe the user interface which is highly customized and branded. After that I describe the results of my empirical research about the students' attitudes to online education.
Keywords: Online teaching, students, attitudes.

A Dunaújvárosi Főiskolán alkalmazott videóval támogatott online oktatási környezet

Napjainkban az online környezetben a tartalom-közvetítés preferált módja a videó lett, a hírportálokon megnövekedett azon cikkek száma, amelyekhez videófilm tartozik, illetve amelyek csak videófilm formájában érhetők el. A közösségi oldalon megszaporodott azon megosztások száma, amelyek videófilmet osztanak meg, illetve a videófilmet megosztó portálok forgalma is folyamatosan növekszik.

Feltevésem szerint, mivel egyrészt rendelkezésre áll a kellő sávszélességet biztosító internet-hozzáférés megfizethető áron, valamint rendelkezésre áll a videófilm-megosztás jól kiforrott technológiája, ezért napjainkban az oktatás számára mindez egy új módszertani elemet ad – az online videófilmmel támogatott E-learning formájában történő oktatási módszert –, melynek még nincsen kiforrott didaktikája. Nem olyan E-learning oktatási anyagra gondolok, amely tartalmaz videófilmeket, hanem amely csak rövid, lényegre törő filmekből áll, amelyek bármikor könnyen megtekinthetők és könnyen megérthetők.

Az ilyen formában zajló E-learning-oktatás napjainkban megfigyelhető a külföldi egyetemek online felületén, pl.: MIT [1], University of Valencia. [2] Az Apple iTunes U-felületen is sok egyetem kínál oktatási anyagokat: hanganyag vagy videófilm formában. Egy tanulmányút keretében 2011-ben az University of Valencia egyetemen látogatást tettünk, hogy tanulmányozni tudjuk az ott folyó videófilmmel támogatott E-learning- oktatás technológiáját és módszertanát. Ennek eredményeként a Dunaújvárosi Főiskolán 2012-ben bevezettük 4 tárgy [3] esetén a videófilmmel támogatott E-learning-oktatást a két vizsgált egyetem (MIT, University of Valencia) módszerei alapján. A bevezetés ideje alatt az University of Valencia egyetem E-learning oktatásért felelős oktatója [4] a Dunaújvárosi Főiskolán tartózkodott, felügyelte és segítette a bevezetés folyamatát.

Ez a felügyelet a „Korszerű tanulási környezetek tervezésének fókuszpontjai" Komenczi (2009) [5] alfejezetben ismertetett fókuszpontok közül a tudásközpontúság pontban leírtakkal van összhangban, hiszen az online képzésben résztvevő oktatók a saját szakterületüket oktatják, ők választhatták ki azt a tárgyat, amelyet online formában kívánnak oktatni, a tananyagok kidolgozását egy, a Főiskolán dolgozó E-learning-szakértő és egy szakmai lektor folyamatosan felügyelte.

Ekkor a tanulóközpontúság is érvényesül, amely egyrészt ott kezdődik, hogy a hallgató maga döntheti el, hogy online vagy frontális módon szeretné a tárgyat a hallgató maga döntheti el, hogy online vagy frontális módon szeretné a tárgyat tanulni, másrészt az online-képzésben résztvevő oktatók a hallgatókat más, frontális tárgyakból is oktatják, így a tanulók ismereteivel az oktatók tisztában vannak.

Az elkészített E-learning-tananyagokban található videófilmekre jellemző, hogy az oktató a képernyő harmadában látszik, a többi területet pedig a szemléltető eszközök töltik ki. Az E-learning oktatási anyagok 15 hétre vannak bontva, hetente jellemzően 5 db 5 perc hosszú videófilmeket tartalmaznak, mindegyik videófilmet 5–10 önértékelő tesztkérdésből álló rövid önellenőrző számonkérés követ.

[1] http://ocw.mit.edu/courses/chemistry/5-95j-teaching-college-level-science-and-engineering-spring-2009/videó-discussions/ [2013. 05. 10]

[2] https://c2k.valenciacollege.edu/Heading.asp?heading_id=1185 [2013. 05. 10]

[3] Ember és társadalom II., Képfeldolgozás és számítógépes grafika, Matematika I., Multimédia I.

[4] Carlos Garcia

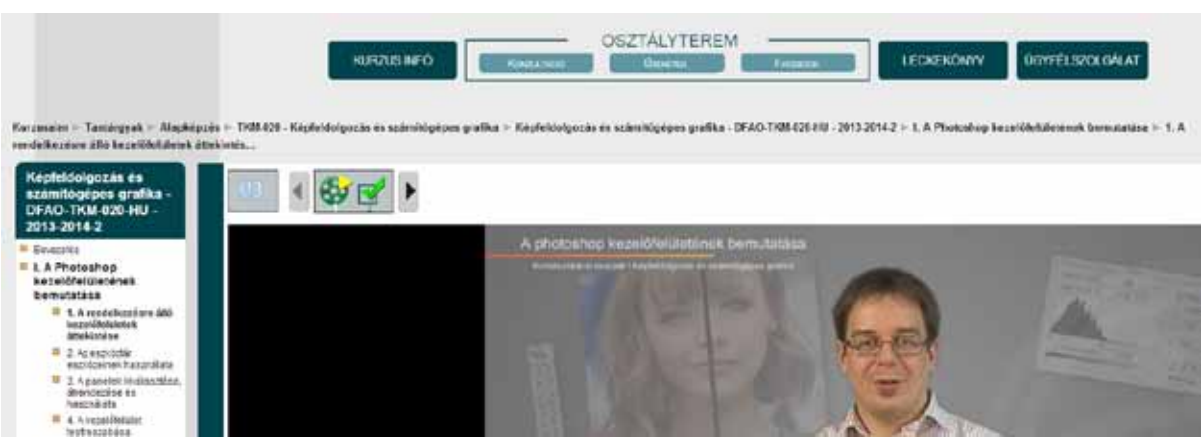
[5] Komenczi B. (2009): *Elektronikus tanulási környezetek*. Budapest: Gondolat.

Minden heti tananyagot szöveges összefoglaló zár, amely egy maximum 2 oldalas dokumentum, amely a heti tananyag legfontosabb ismereteinek a leírását tartalmazza. Ezen kívül a heti témát még két darab beadandó feladat és egy összefoglaló teszt is zárja, amely 25 kérdésből áll. A teszt és a két beadandó feladat (amelyet az oktató szövegesen is értékel) a kurzus teljesítésének értékelése során - az oktató által meghatározott mértékben - az érdemjegy meghatározásába beleszámít. A félév során kettő zárthelyi dolgozatot is teljesíteniük kell a hallgatóknak, amin személyesen kell megjelenniük. Ezzel a komplex számonkérési rendszerrel a hallgatók teljesítménye folyamatosan követhetővé válik, állandó készülésre ösztönöz, Komenczi [5] publikációjában ismertetett értékelés-középpontúság is teljesül.

A legelterjedtebb online környezetek vizsgálata és a saját fejlesztésű felület lehetőségén is elgondolkodva, rendszerként a Moodle 1.9-es változatának használata mellett döntött a Főiskola vezetése. Átalakításra, leegyszerűsítésre került a kezelőfelülete úgy a hallgatók, mint az oktatók oldalán. A felület megtekinthető a https://moodle.duf.hu címen, vendégként is lehetséges a bejelentkezés. A hallgatók, ill. az oktatók a Neptunban használt azonosítójukkal és jelszavukkal tudnak bejelentkezni.

Amikor valaki bejelentkezik a Moodle-felületre, akkor a saját kurzusainak a listáját látja, az adatok a Neptun tanulmányi rendszerből származnak, a kurzuslista 3 naponta kerül aktualizálásra. Mindenki csak a saját kurzusához fér hozzá.

1. ábra. A saját kezelőfelület.



A kurzus kiválasztását követően egy függőleges és vízszintes menüvel rendelkező felület jelenik meg. (*1. ábra*)

A függőleges menü segít a tananyagelemek közötti tájékozódásban: a fejezetek, alfejezetek itt jelennek meg. A felület sajátossága, hogy az alfejezetekhez tartozó lehetőségek a szürke sáv alatti területen jelenik meg ikonok formájában. Itt van lehetőség a szöveges jegyzet megtekintésére, a videók lejátszására, tesztek kitöltésére, fájlok feltöltésére.

2. ábra. Oktatóvideó.

A felső, vízszintes menüsor letisztult; megtekinthető egy kattintásra a kurzus tematikája, a konzultációs formák (online chat, videó chat, fórum). Az online-konzultációk időpontja a hallgatók órarendjében is látszik, azokon a részvétel kötelező. Az üzenetek menüpontban van lehetőség az oktatóval történő kapcsolatfelvételre. Minden kurzusnak létezik egy zárt Facebook-csoportja, amely a felső menürendszerből elérhető, az oktatók a csoportba jellemzően három naponta írnak üzenetet, pl. emlékeztetik a hallgatót egy határidő közeledtéről. Így a negyedik szempont, a közösség-középpontúság is érvényesül, ezen túlmenően pedig az online tanulásvezetés is.

Ennek a csoporton belüli kommunikációnak a fontosságára Fejes (2007) [6] is rámutatott, a legkritikusabb faktorként nevezve meg.

A leckekönyvre kattintva a hallgató látja a kurzusból elért eredményeit egy áttekinthető táblázatos formában. Az ügyfélszolgálat menüpontban van lehetősége a hallgatónak segítséget kérnie a Moodle-rendszergazdától.

Az oktatóvideók legelterjedtebb megjelenítési formája az a nézet, amikor az oktató teljes alakban látszik (2. ábra). Számítógépes alkalmazások oktatásánál elterjedt még a megosztott nézet, amikor az oktató csak a jobb alsó sarokban látszik, a többi helyen a számítógép képernyője foglal helyet. Elméleti tárgyak esetében a prezentáció diája és az oktató fele-fele arányban látszik.

A rendszer bevezetésének fő koncepciója volt, hogy az oktatók egyszerűen hozhassanak létre videós oktatóanyagot, ezért kezdetben egy, napjainkban kettő stúdió került kialakításra, és az oktató az előre lefoglalt időpontban bemegy, ahol az előre leadott igényei alapján berendezett környezet fogadja. Az előadása, használt nyersanyagok felmásolása után a felvétel elindul, jó esetben – kellő gyakorlat és megfelelő felkészülés után – nem tart tovább a felvétel, mint egy óra megtartása. Az utómunka körülbelül két hetet vesz igénybe.

A stúdió minden külső fényforrástól mentes, és emellett hangszigetelt is. Kizárólag a stúdió saját fényforrásait lehet felhasználni. Fontos a megvilágítás során fényáteresztő szűrőpapír használata. A szűrő funkciója az, hogy tompítsa a direkt fényt, ezáltal irányítottan szórt fénnyé alakítsa át azt. A szórt fény legfőbb haszna, hogy a fénytörés miatt tompább, mattabb lesz a fény, ezáltal sokkal természetesebb közeget lehet teremteni vele, és a felvétel szempontjából is előnyösebb.

A rögzítés közvetlenül a számítógépre történik. Ennek nagy előnye az, hogy az utómunkák során nem kell feleslegesen várni a nyersanyag beolvasásra. A felvételek minden esetben két kamerával készülnek. Az egyik kamera az oktatót szemből veszi, a másik pedig felülről, így lehetséges olyan vágóképek beillesztése, amikor az oktató papírra ír vagy rajzol. Ezek a kamerák három CCD-vel rendelkező, félprofi Sony kamerák. Az E-learning-videók felbontása 1920x1080, azaz full HD minőségben készülnek. A HD felbontás nagyszerűen használható a kisebb kapacitású platformokon is. A nagy méretnek köszönhetően kitűnően látható a kis képernyőkön is. Ez fontos, hiszen sok hallgató mobil eszközön tekinti meg a tartalmat, a Moodle-rendszerbe kétféle minőségben kerülnek a videók feltöltésre.

Az oktatók minden esetben egy zöld, kifeszített, megvilágított (lámpánként 500 W) háttér előtt helyezkednek el, ami az utómunka során lyukasztásra kerül.

[6] Fejes J. B. (2007): *Online tanulóközösségek*. Iskolakultúra, 4. Pp32–37.

Az empirikus kutatásom elméleti háttere

A kutatásom során az előző fejezetben ismertetett online oktatási módszer elfogadottságát tárom fel a Főiskola hallgatói körében, vizsgálom a hatékonyságát a kontaktórás kurzusokkal összevetve a félévvégi érdemjegyek alapján.

Az E-learning témájával foglalkozó hazai szakirodalmakat áttekintve azt tapasztaltam, hogy az online oktatás területén a fejlesztések az egyes intézményekben önállóan folynak, alig születnek az elért eredményekről publikációk, ha mégis, akkor a publikációk kizárólag az elért saját sikerek bemutatására szorítkoznak, ezért tartom fontosnak a kutatás eredményei mellett a Főiskolán kidolgozott koncepciót és munkamódszereket ismertetni. A kutatásom és a koncepció ismertetése előtt áttekintettem több projektről történő beszámolót: Komenczi (2009) [5] publikációjában bemutatott amerikai, ausztráliai és németországi projektek ismertetését, melyek jó alapot nyújtottak a kérdőív kidolgozásához és a cikk megírásához egyaránt. Az ismertetett projektek csak részben mutatnak hasonlóságot a Főiskolán alkalmazott online oktatási módszerrel, melynek egyedisége leginkább a teljesen videó alapúságban és az online konzultációk órarendbe illesztésében jelenik meg. Természetesenaz online előadás, mint a hálózati-tanítás és -tanulás alapformája megjelenik a publikációban, a hivatkozott egyetemek (University of California Television Online, MIT World Videó Archive, Princeton University WebMedia Lectures, UC Berkeley: Conversations with History) weboldalai egyetemi videóelőadás-adatbázisok, nem alkotnak olyan komplex online rendszert, mint ami a Főiskolán kidolgozásra került.

Kutatásom alaphipotézise, hogy az online oktatás sikeressége függ attól, hogy a hallgató milyen képzési területen tanul és attól is, hogy az online tanítani kívánt tárgy milyen képzési területhez tartozik, valamint hogy a hallgató milyen informatikai kompetenciákkal rendelkezik. Hipotézisem vizsgálata során kereszttáblás elemzést fogok végezni, a szignifikancia-vizsgálatot Khi-négyzet-próbával hajtom végre.

A kérdőívem kidolgozásánál és a kutatási kérdés megfogalmazásánál felhasználtam Ollé (2013) [7] publikációját, amely fontos jelentőséget tulajdonít az online képzések során az attitűdök vizsgálatának, melyet a kérdőív célzott kérdéseivel vizsgálom. Az attitűdök vizsgálatát különböző tárgyak mentén tagoltam (informatika, matematika, politológia), amely alapját két publikáció adta, melyben a jog (Verebics, 2013) [8] és a nyelvtanítás (Szabó, 2013) [9] E-learning-módszerrel való oktatásának vizsgálata olvasható.

[5] Komenczi B. (2009): *Elektronikus tanulási környezetek*. Budapest: Gondolat.

[7] Ollé J. (2013): *Virtuális környezet, virtuális oktatás.* Budapest: ELTE Eötvös.

[8] Verebics J. (2013): „Élmnypedagógia" – elektronikus környezetbe ágyazottan. In: Benedek A. (Szerk.): *Digitális Pedagógia. 2.0*. Budapest:Typotex.

[9] Szabó, E. M. (2013): A Laterna Magicától az okostelefonig. Az online nyelvtanulás és nyelvtanítás egy lehetséges modellje. In: Benedek A. (Szerk.): *Digitális Pedagógia 2.0*. Budapest:Typotex.

A 2012. és 2013. évi empirikus kutatás mintavételi eljárása és a minták összetétele

Az empirikus kutatásomat két egymást követő évben online kérdőíves lekérdezéssel végeztem a Dunaújvárosi Főiskola aktív státuszú, nem idegen nyelvű képzésben résztvevő hallgatói körében. Az első lekérdezés 2012. november 9–18-ig, a második lekérdezés 2013. november 8–17-ig tartott.

A hallgatók a kérdőív kitöltéséhez szükséges linket a Neptun-rendszer részét képező Unipoll kérdőíves lekérdezést biztosító rendszeren keresztül kapták meg, így biztosítva volt a zárt hallgatói csoport. A kérdőív kitöltésére a hallgatók kétszer kaptak emlékeztető e-mail üzenetet. A kérdőív kitöltése során a válaszokhoz az Unipoll-rendszer lehetővé tette, hogy a Neptunból a válaszadók alábbi adatai rendelkezésre álljanak: a válaszadó képzésének a neve, tagozata (nappali vagy levelező), a képzés szintje (BA, BSc, MA, MSc, felsőfokú szakképesítés, felsőoktatási szakképesítés), a válaszadó születési dátuma, neme, lakhelye (csak a település neve). Így a kérdőívben demográfiai adatok lekérdezése nem vált szükségessé és biztosítható azok pontossága.

Mindkét lekérdezés esetében a képzések neve mellett a képzéseket gondozó tanszékeket is feltüntettem (Anyagtudományi, Gépészeti, Informatika, Kommunikáció- és Médiatudományi, Közgazdaságtudományi, Tanárképző, Vezetés- és Vállalkozástudományi), amelyek így biztosítják a képzések tudományterületenként történő csoportosítását. A kutatás során a képzések neve helyett is ezzel a változóval dolgoztam, a minta elemszámtartó súlyozással reprezentatív a képzést gondozó tanszékek szerint [10], az elemzések során a súlyozást végig használtam. A válaszok közül csak a teljesen kitöltött kérdőíveket vettem figyelembe.

Látható az *1. táblázat*ból, hogy a súlyozás csak kis mértékben változtat a minta összetételén, legnagyobb szerepe abban van, hogy 2012-ben az informatikus, 2013-ban a tanárképzésben résztvevő hallgatók ne legyenek felülreprezentálva.

A *2. és a 3. táblázat* a minták nemek szerinti megoszlását mutatja be, a minta összetétele már a súlyozást követően került feltüntetésre, egyik évben sem tapasztalható nagy eltérés a minta és az alapsokaság között a nemek tekintetében.

Kutatásom célja, eredményeim általánosítása a Dunaújvárosi Főiskola hallgatóira vonatkozóan a tudományterületek szerinti reprezentativitással és a nemek szerinti összetétel alapján.

[10] Az összes szak és képzési terület lekérdezésre került, mindegyik szakról érkezett válasz, azonban egyes szakok nagyon kis létszámúak, ezért tartottam célszerűnek a képzést gondozó tanszékek alapján történő reprezentativitást, amely szükséges annak érdekében, hogy az eredmények általánosíthatók legyenek a Főiskolára.

1. táblázat. A minták összetétele.

	Minta		Alapsokaság		Súly
2012					
Anyagtudományi	50	3,28%	103	3,34%	1,016979
Gépészeti	270	17,73%	537	17,41%	0,981873
Informatika	507	33,29%	691	22,40%	0,672845
Kommunikáció- és Médiatudományi	127	8,34%	275	8,91%	1,06899
Közgazdaságtudományi	211	13,85%	622	20,16%	1,4553
Tanárképző	164	10,77%	391	12,67%	1,177003
Vezetés- és Vállalkozástudományi	194	12,74%	466	15,11%	1,185848
Összesen	**1523**		**3085**		
2013					
Anyagtudományi	23	2,82%	79	3,08%	1,092277
Gépészeti	170	20,83%	506	19,72%	0,946532
Informatika	202	24,75%	618	24,08%	0,972905
Kommunikáció- és Médiatudományi	40	4,90%	154	6,00%	1,224318
Közgazdaságtudományi	166	20,34%	527	20,54%	1,009569
Tanárképző	116	14,22%	307	11,96%	0,841616
Vezetés- és Vállalkozástudományi	99	12,13%	375	14,61%	1,204563
Összesen	**816**		**2566**		

2. táblázat. A 2012. évi minta nemek szerinti megoszlása.

	2012			
	Alapsokaság	Alapsokaság %	Minta	Minta %
Férfi	2113	68,49%	921	60,47%
Nő	972	31,51%	602	39,53%

3. táblázat. A 2013. évi minta nemek szerinti megoszlása.

	2013			
	Alapsokaság	Alapsokaság %	Minta	Minta %
Férfi	1797	70,03%	579	70,96%
Nő	769	29,97%	237	29,04%

A hallgatók informatikai kompetenciának a vizsgálata

A kérdéscsoport kérdéseiből megállapítható, hogy a Főiskola hallgatói – ahogyan az várható is volt – használják az internetes szolgáltatásokat (2012-ben a válaszadók 98 %-a, 2013-ban 97,6 %-a), a megkérdezettek rendelkeznek saját email címmel (2012-ben a válaszadók 99,6 %-a, 2013-ban 99,0 %-a).

4. táblázat. Emailolvasás gyakorisága (2012).

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
1	2,4%	16,1%	24,5%	7,7%	17,6%	15,0%	16,6%
2	4,1%	19,8%	18,8%	8,8%	24,4%	10,7%	13,4%
3	3,8%	18,0%	18,0%	14,3%	18,8%	9,0%	18,0%
4	8,1%	18,9%	21,6%	12,2%	31,1%	2,7%	5,4%
E	4,8%	19,0%	33,3%	14,3%	14,3%	9,5%	4,8%

A *4. táblázat* mutatja az emailolvasás gyakoriságát a 2012. évi felmérésben, a válaszok értékei az alábbiakat jelentik: 1 - naponta többször, 2 - naponta egyszer, 3 - hetente többször,

4 - hetente egyszer, E - egyéb. A képzési területek szerint nem szignifikáns, de jelentős eltérések tapasztalhatók a válaszokban.

Az *5. táblázat* a 2013. évi kutatás eredményeit mutatja, amelyet összevetve az előző évi kutatás eredményeivel az tapasztalható, hogy a legtöbb esetben az emailolvasás gyakorisága nőtt. Az emailolvasás gyakorisága és a képzési terület jelen esetben sem mutat szignifikáns különbséget.

5. táblázat. Emailolvasás gyakorisága (2013).

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
1	2,8%	19,7%	26,8%	5,3%	17,2%	12,8%	15,4%
2	4,6%	20,3%	21,5%	6,3%	25,7%	10,1%	11,4%
3	0,0%	18,1%	20,8%	13,9%	19,4%	8,3%	19,4%
4	1,9%	17,0%	22,6%	0,0%	28,3%	15,1%	15,1%
E	5,9%	23,5%	17,6%	5,9%	17,6%	17,6%	11,8%

Az online oktatás egyik legnagyobb kihívása a konzultációs formák megválasztása, ezért fontosnak tartottam felmérni, hogy a hallgatók mennyire használják a csevegő szolgáltatásokat, az eredmények a *6. és a 7. táblázat*ban láthatók. A válaszok értékei az alábbiakat jelenti: 1 - még nincs ilyen tapasztalatom, 2 - egyáltalán nem, 3 - alkalmaztam néhányszor, 4 - igen, sokszor használom. Habár a kereszttáblák szignifikáns összefüggést nem mutatnak, megállapítható, hogy a 2013. évi vizsgálatban már többen, gyakrabban csevegtek és mindkét vizsgálat esetében a legnagyobb számban az informatikus, gazdász és a gépész hallgatók használták ezt a szolgáltatást.

6. táblázat. Csevegő szolgáltatások használata (2012).

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
1	5,9%	23,5%	23,5%	5,9%	17,6%	23,5%	0,0%
2	4,5%	21,8%	11,8%	3,6%	10,9%	28,2%	19,1%
3	4,6%	21,6%	20,7%	7,8%	15,5%	13,6%	16,3%
4	2,3%	14,3%	24,7%	10,3%	24,2%	10,1%	14,1%

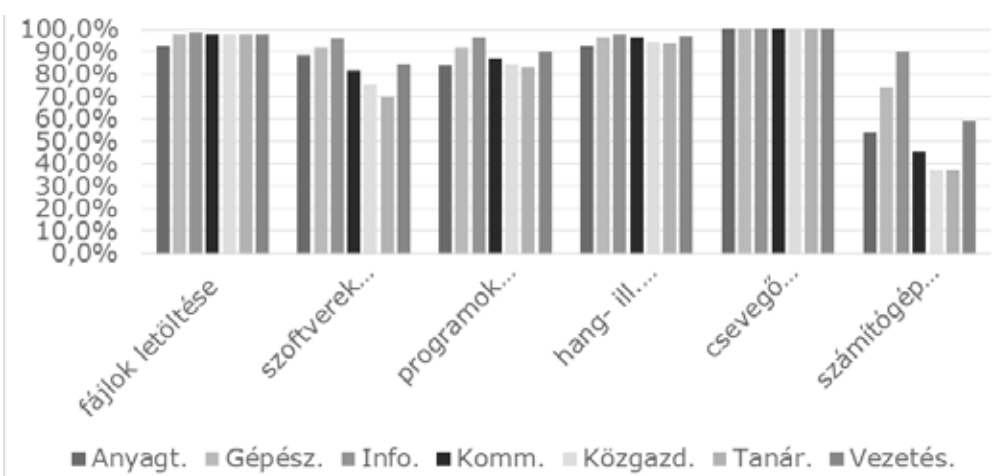
7. táblázat. Csevegő szolgáltatások használata (2013).

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
1	0,0%	30,4%	17,4%	4,3%	17,4%	13,0%	17,4%
2	3,5%	15,8%	15,8%	10,5%	22,8%	12,3%	19,3%
3	1,7%	19,5%	27,0%	5,0%	21,6%	13,3%	12,0%
4	3,8%	19,8%	24,0%	5,9%	19,8%	11,3%	15,4%

Fontosnak tartottam felmérni, hogy a hallgatók hogyan ítélik meg saját informatikai képességüket, mennyire képesek önállóan elvégezni az elektronikus tanulás során felmerülő műveleteket, az eredmények a 3. ábrán láthatók.

A vizsgált tevékenységek listája az alábbi: fájlok letöltése, szoftverek letöltése, programok installálása/eltávolítása, hang- ill. képállomány küldése, csevegő programokon keresztül csatolt állomány továbbítása és számítógép konfigurálása. A két kutatás eredményei között nincs jelentős különbség, az ábrán a 2012. évi kutatás eredményei láthatók. Az értékek azt mutatják, hogy a hallgatók hány %-a képes a tevékenység elvégzésére.

3. ábra. Számítógép-használati képességek felmérése (2012)



A *8. és a 9. táblázat*ban látható, hogy a hallgatók milyen mértékben használják tanulás során az internetet. Az egyes válaszok értékei az alábbiakat jelentik: 1 - egyáltalán nem, 2 - kismértékben, 3 - többé-kevésbé, 4 - teljes mértékben. Egyik összefüggés sem szignifikáns, azonban látható, hogy a legnagyobb mértékben az informatika és a gazdálkodástudomány területeken tanuló hallgatók használják tanulás során a leggyakrabban az internetet és az egyes képzési területek között nagy különbözőség tapasztalható.

A *4. ábrán* látható, hogy a hallgatók a nyomtatott vagy az elektronikus segédanyagot használják-e szívesebben. A szakok között a 2012. évi felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,000 szinten) különbség van, a kapcsolat Cramer-erőssége 0,130. A tudományterületenként eltérő válaszok az ábrán jól láthatók.

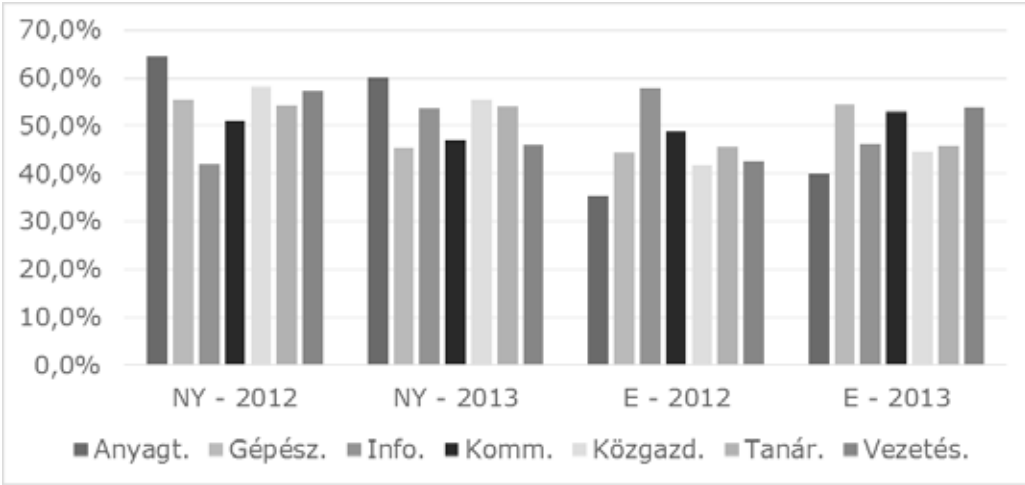
8. táblázat. Internethasználat a tanulás során (2012).

	Tanszék						
	Anyagt.	**Gépész.**	**Info.**	**Komm.**	**Közgazd.**	**Tanár.**	**Vezetés.**
1	12,5%	25,0%	0,0%	0,0%	50,0%	12,5%	0,0%
2	6,7%	21,3%	21,3%	4,0%	16,0%	10,7%	20,0%
3	3,9%	20,2%	19,6%	10,1%	18,9%	11,3%	16,0%
4	2,3%	14,7%	25,2%	8,4%	21,4%	13,9%	14,1%

9. táblázat. Internethasználat a tanulás során (2013).

	Tanszék						
	Anyagt.	**Gépész.**	**Info.**	**Komm.**	**Közgazd.**	**Tanár.**	**Vezetés.**
1	0,0%	25,0%	0,0%	0,0%	25,0%	50,0%	0,0%
2	0,0%	23,4%	25,5%	4,3%	23,4%	12,8%	10,6%
3	1,5%	18,4%	27,2%	6,6%	20,8%	12,4%	13,0%
4	4,4%	20,5%	21,9%	5,6%	20,0%	11,4%	16,3%

4. ábra. Nyomtatott / elektronikus segédanyag használata



A hallgatók korábbi E-learning tapasztalatainak vizsgálata

A *10. táblázat*ban látható, hogy a hallgatók hány százaléka vett már részt korábban E-learning-oktatáson. Meglepő módon a 2013. évben a tanárképzésben résztvevőkön kívül mindenki más nagyobb arányban vett részt E-learning-oktatáson. 2012-ben a tanárképzéses hallgatók vettek részt legnagyobb arányban E-learning-képzésben. A 2012. évi felmérés eredménye Khi-négyzet-próba szerint szignifikáns (0,000 szinten), a kapcsolat Cremer-erőssége 0,140.

10. táblázat. E-learning képzésben való részvétel

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
I - 2012	9,8%	10,9%	13,5%	15,6%	10,7%	26,4%	14,3%
I - 2013	32,0%	18,0%	16,8%	24,5%	17,4%	19,4%	16,0%
N - 2012	90,2%	89,1%	86,5%	84,4%	89,3%	73,6%	85,7%
N - 2013	68,0%	82,0%	83,2%	75,5%	82,6%	80,6%	84,0%

10. táblázat. E-learning képzésben való részvétel

	Tanszék						
	Anyagt.	Gépész.	Info.	Komm.	Közgazd.	Tanár.	Vezetés.
I - 2012	9,8%	10,9%	13,5%	15,6%	10,7%	26,4%	14,3%
I - 2013	32,0%	18,0%	16,8%	24,5%	17,4%	19,4%	16,0%
N - 2012	90,2%	89,1%	86,5%	84,4%	89,3%	73,6%	85,7%
N - 2013	68,0%	82,0%	83,2%	75,5%	82,6%	80,6%	84,0%

A hallgatók kifejtős kérdésben megadhatták, hogy hol vettek részt korábban E-learning-képzésben, a válaszok nagyon vegyesek, azonban jól körülhatárolható területeket jelöltek meg a kérdezettek. 53 fő tanult már más oktatási intézményben E-learning formában, a válaszok között a hazai neves felsőoktatási intézményeken túl, OKJ-s képzések, KRESZ-tanfolyam, számos külföldi oldal is megtalálható, de többen válaszolták a homeopátiás-, sportedzői- és tűzsdetanfolyamokat is. 50 fő tanult már a Főiskolán E-learning formában valamilyen tárgyat. 38 fő a munkahelyén tanult már, 6 darab meghatározó cég van jelen a válaszok között: a biztosítás, bútorértékesítés, erőmű, olajfinomító és telekommunikáció területekről. 8 fő válaszolt úgy, hogy idegen nyelven ilyen formában tanul.

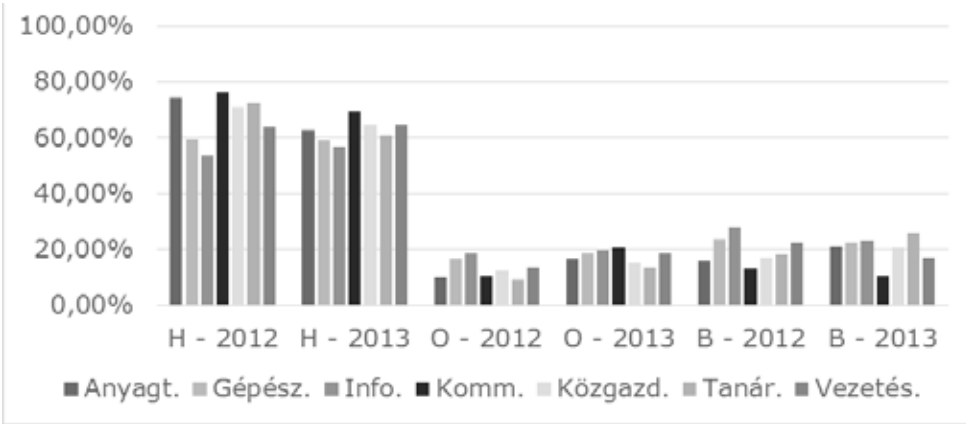
A következő kérdésben a hallgatóknak az elvárásukat kellett megfogalmazniuk az E-learning-képzésekkel szemben. A legtöbben az alábbiakat válaszolták: gyors, hatékony, egyszerű, érthető, naprakész, megbízható, minőségi, alapos, gyakorlati szemléltetés, gyors reakció a kapcsolattartásban, folyamatos frissítés, hanggal támogatott online konzultáció, szabad időbeosztás.

Az online és a hagyományos képzési forma összehasonlító elemzése

Hipotézisem szerint a hallgatók online oktatásra való nyitottsága függ a hallgató képzési területétől valamint attól is, hogy a tárgy milyen képzési területhez tartozik. Ezért megvizsgáltam azon tárgyak esetében az online oktatásra való nyitottságot, ahol rendelkezésre áll a publikációm elején ismertetett videóval támogatott online képzési forma. Ezek az alábbi tárgyak: Matematika felkészítő, Matematika I., Multimédia I., Képfeldolgozás és számítógépes grafika, Ember és társadalom II. (politológia és szociológia) és Mérnöki fizika (csak a 2013. évi lekérdezésben).

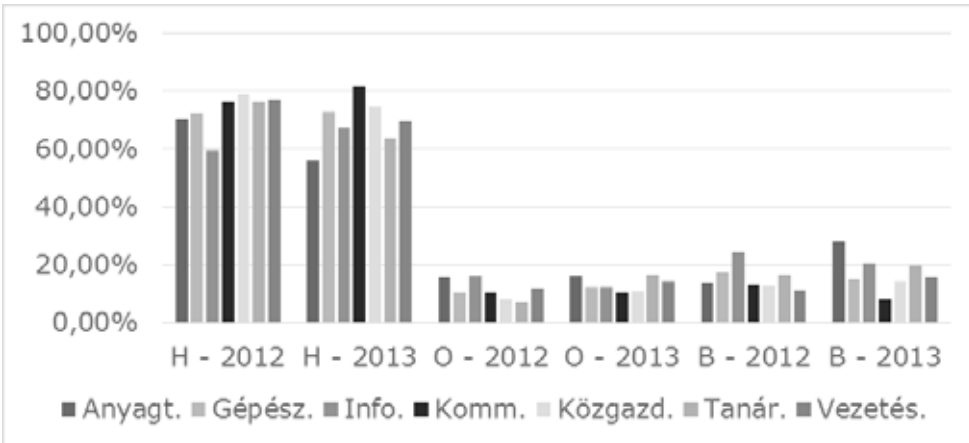
A Matematika felkészítő kurzus esetében a válaszok az 5. ábrán láthatók. Az összefüggés mindkét felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,000 szinten), a kapcsolat Cramer szerinti erőssége 2012. évi felmérés esetén 0,119, a 2013. évi felmérés esetén pedig 0,078.

5. ábra. Matematika felkészítő tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



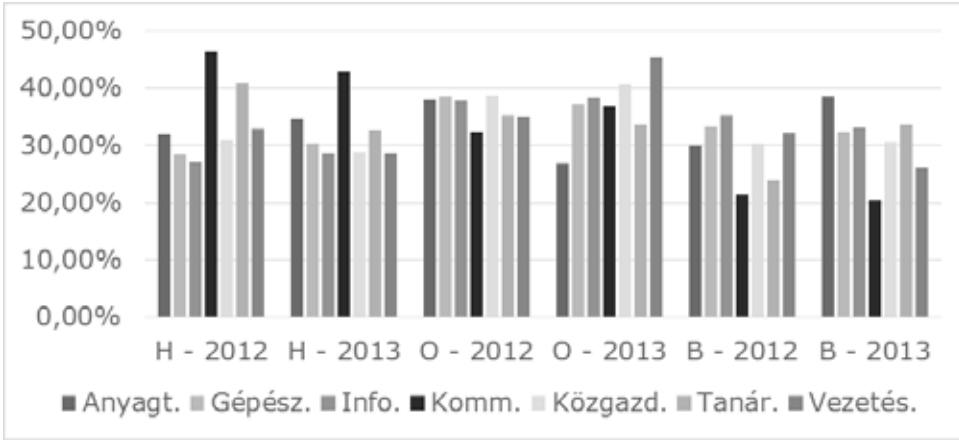
A *6. ábrán* a *Matematika I.* kurzus esetében adott válaszok láthatóak, érdekes, hogy az előző ábrával összevetve többen támogatják ezen kurzus az online oktatását. Az összefüggés a 2012. évi felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,000 szinten), a kapcsolat Cramer szerinti erőssége 0,122.

6. ábra. Matematika I. tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



A *Multimédia I.* tárgy esetén az eredmények a *7. ábrán* láthatóak. Fontos kiemelni, hogy ez a tárgy nagyon régóta elérhető online formában, mivel sok különböző szakon tanuló hallgató számára kötelező a kurzus, valószínű, hogy az eredményeket ez is nagymértékben befolyásolta. Az összefüggés a 2012. évi felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,005 szinten), a kapcsolat Cramer szerinti erőssége 0,097.

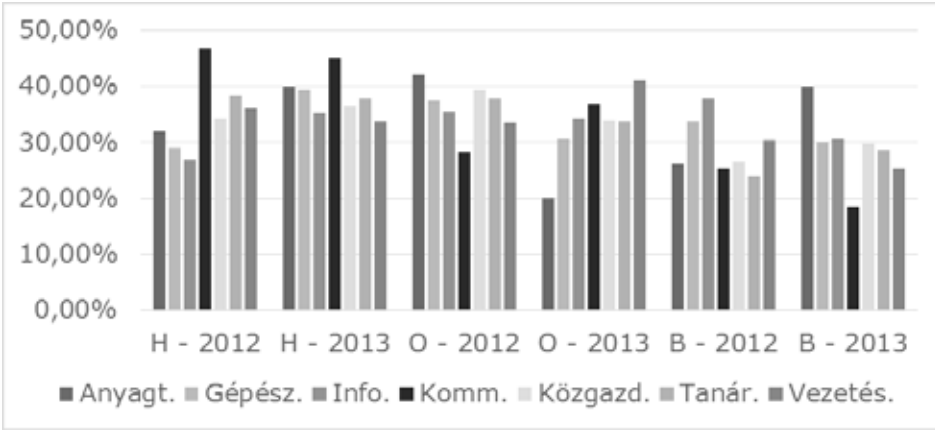
7. ábra. Multimédia I. tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



A *Képfeldolgozás és számítógépes grafika* tárgy esetén az eredmények a 8. ábrán láthatóak. Nagymértékben megegyeznek az eredmények a *Multimédia I.* kurzus eredményeivel.

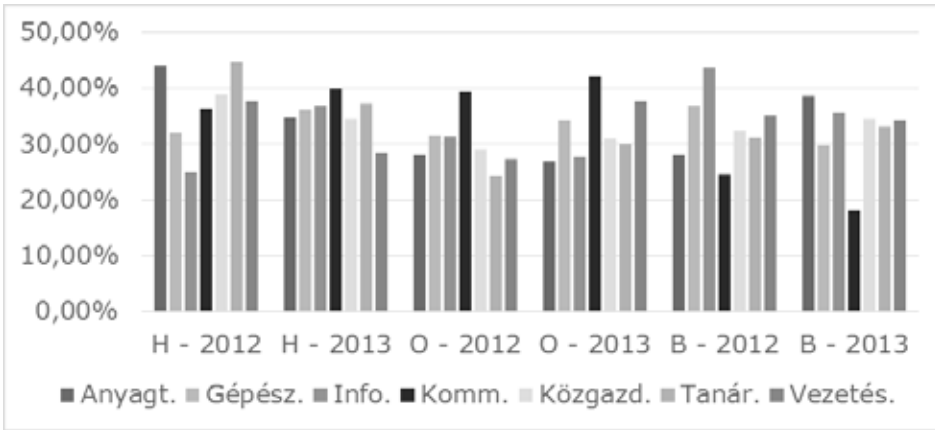
Az összefüggés a 2012. évi felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,001 szinten), a kapcsolat Cramer szerinti erőssége 0,103.

8. ábra. Képfeldolgozás és számítógépes grafika tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



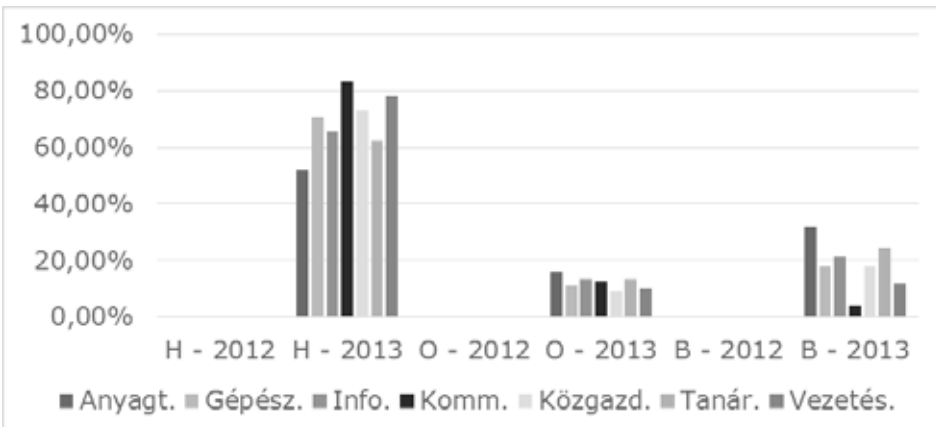
A *9. ábrán* az *Ember és társadalom II.* kurzus eredményei láthatók, a vizsgálatban az egyetlen, amelyben csak elméleti ismeretek kerülnek átadásra. Lehet látni, hogy egyes csoportok között a tárgy online oktatása kimagaslóan támogatott. Az összefüggés a 2012. évi felmérés esetében Khi-négyzet-próba szerint szignifikáns (0,000 szinten), a kapcsolat Cramer szerinti erőssége 0,114.

9. ábra. Ember és társadalom II. tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



A *10. ábrán* a *Mérnöki fizika kurzus* 2013. évi eredményei láthatók (2012-ben a tárgy még nem volt elérhető online formában). Az eredmény nagyon érdekes, ezt a tárgyat sokkal többen csak hagyományos formában tudják elképzelni, pedig az online tananyagot ismerve nagyon érdekes, szemléletes az elméleti ismeretek átadása – nyilván ezt sokan nem tekintették meg a válaszadók közül. Az összefüggés nem szignifikáns.

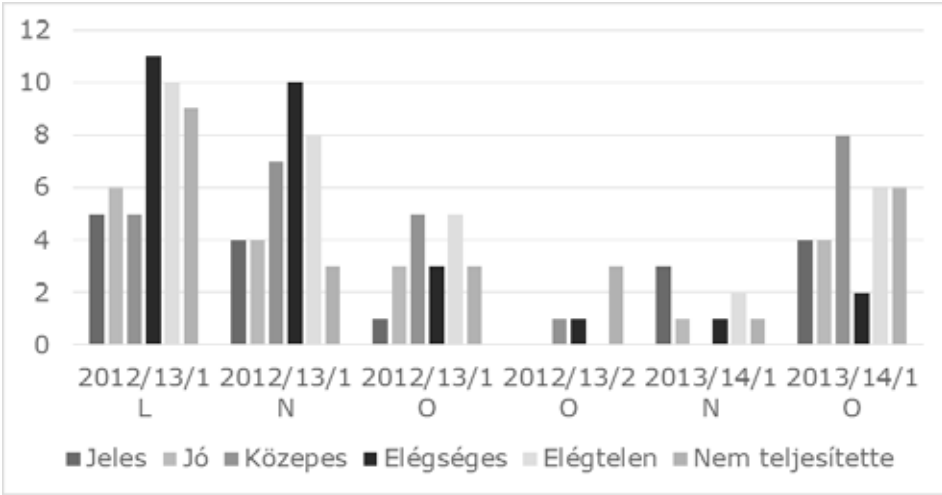
10. ábra. Mérnöki fizika tárgy hagyományos, online, bármelyik formában történő oktatásának támogatása.



A kurzusok külön-külön történő elemzését azért tartottam fontosnak, mert számomra úgy tűnik, hogy az online oktatás sikerességét több tényező is befolyásolja. Szerepe van annak, hogy a tárgy oktatása mióta folyik már online módon (a két szélsőséges példa a *Multimédia I.* és a *Mérnöki fizika* tárgyak), a számítógépes programok használatához köthető és tisztán elméleti tárgyak (*Ember és társadalom II.*, *Képfeldolgozás és számítógépes grafika, Multimédia I.* tárgyak) esetén nyitottak a hallgatók az online tanulásra. A *matematika* -tanulás a hallgatók számára mindig is nehézségekkel járt, ez az online oktatás esetében az eredményekben is megjelenik, pedig álláspontom szerintem kifejezetten jól tanítható a matematika online módon, nagy előny, hogy egy-egy példa megoldása vagy egy elméleti magyarázat bármikor visszakereshető, újra megnézhető, mégis számomra a vártnál többen elutasítóak az online tanulással kapcsolatban.

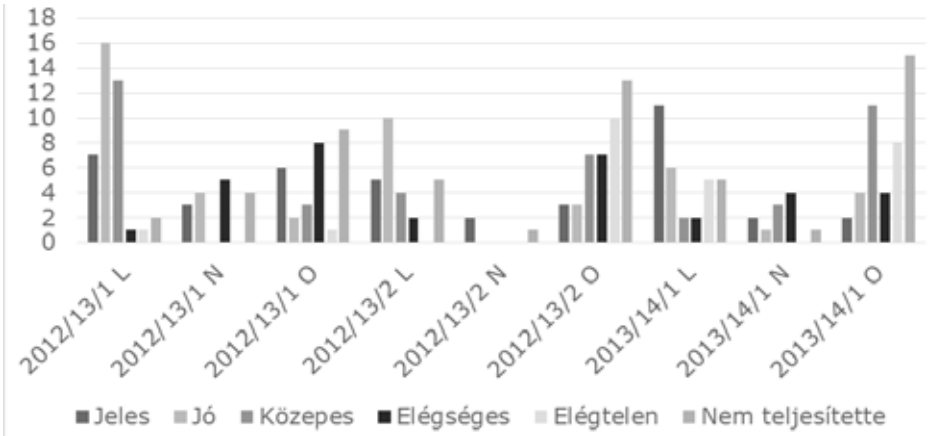
A tanulmányi eredmények összehasonlítása az *Ember és Társadalom II.* tárgy esetében a *11. ábrán* látható. A képzési formát rövidítések jelölik: L – levelező, N – nappali és O – online kurzust jelöl. Az oktató a levelező tagozatos hallgatóknak ajánlotta az online kurzus anyagai a használatát, az ő eredményük lett a legjobb. A második online kurzus esetében a hallgatók jobban teljesítettek, mint az első kurzus esetében, ez álláspontom szerint a hallgatók online környezethez történő alkalmazkodásuknak köszönhető.

11. ábra. Ember és Társadalom II. tárgy eredménye.



A *12. ábra* a *Multimédia I.* tárgy eredményeit mutatja a félévek tükrében, a rövidítések az előző ábráéval egyeznek meg. Az eredményekben nagy a szórás, az látható, hogy mivel a hallgatók döntik el, hogy online vagy hagyományos kurzust választanak, sokan választják a tárgy online teljesítését, pedig az eredmények alapján látható, hogy ott sem születnek jobb eredmények, mint a többi kurzuson.

12. ábra. Multimédia I. tárgy eredményei.



Az eredményeim összefoglalása

Jelen vizsgálatom alapján azt látom, hogy a két év alatt a hallgatók és az oktatók részéről egyértelmű nyitottság látható az online képzések iránt, másként tekintenek az online képzésre, mint a 2000-es évek elején, amikor egy E-learning tananyag tulajdonképpen fájlok gyűjteményéből állt. Úgy látom, hogy segíti az online oktatás elfogadását a színvonalas, videóalapú anyagok elkészítése, amelyekben a Dunaújvárosi Főiskola élen jár. Segíti továbbá az elfogadást az, hogy a hallgatók a Főiskolán kívül is találkoznak E-learning-anyagokkal: munkahelyükön, KRESZ / OKJ / nyelvtanfolyam vagy éppen önképzésük során. Természetesen vizsgálatomat 2014. novemberében is el fogom végezni, a kérdőív aktualizálásával együtt.

Az empirikus kutatások mellett fókuszcsoportos vizsgálatot is végeztem, amelynek eredménye elérhető az alábbi kiadványban online a http://mmo2014.nyme.hu címen: A Dunaújvárosi Főiskolán alkalmazott videóval támogatott online oktatás bemutatása és kutatási eredményeinek ismertetése In: Dr. Berke József (Szerk.) XX. Multimédia az oktatásban konferencia előadások. Konferencia helye, ideje: Sopron, Magyarország, 2014. 06. 05 - 2014. 06. 06. Budapest: Neumann János Számítógép-tudományi Társaság, 2014. Pp. 50–58.

LEVENTE RÁDAI * –VID SEBESTYÉN HONFI ** –ZOLTÁN KIRÁLY ***
–ANITA MIHALOVICS KOLLÁR *****

Teaching of ERP systems at the College of Dunaújváros

Abstract: The aim of this article is to present the development of the teaching activity of ERP systems at the College of Dunaújváros in the last one and a half decade through using SAP products for small, medium and large enterprises till started using Microsoft Dynamics NAV in September of 2013. The article compares the features of these ERP systems from the viewpoint of teaching and summarises the practical methods and its development used in the education. Moreover it try to measure the efficiency of teaching through the student results related to their academic average.
Keywords: Teaching ERP systems, SAP R/3, SAP Business One, SAP ECC 6.0, IDES, MS Dynamics NAV, teaching efficiency.

Összefoglalás: A cikk célja, hogy bemutassuk a Dunaújvárosi Főiskola integrált vállalatirányítási rendszerekkel kapcsolatos oktatási tevékenységét és fejlődését az elmúlt másfél évtizedben. A cikk áttekinti az ezen időszak alatt alkalmazott SAP különböző nagy- és kisvállalati rendszereivel, valamint a 2013. őszén oktatásba bevezetett Microsoft Dynamics NAV-rendszerrel kapcsolatos oktatási tapasztalatokat és az oktatásban alkalmazott gyakorlati módszereket, feladatokat. Továbbá elemzi a vállalatirányítási rendszerekkel foglalkozó tantárgyak esetében mérhető oktatási eredményességet a tantárgyi eredmények hallgatói kumulált tanulmányi átlagokkal történő összevetésével.
Kulcsszavak: Integrált vállalatirányítási rendszerek oktatása, ERP-rendszerek, SAP R/3, SAP Business One, SAP ECC 6.0, IDES, MS Dynamics NAV, oktatási hatékonyság.

* *Dunaújvárosi Főiskola Társadalomtudományi Intézet*
E-mail: radai@mail.duf.hu

** *Dunaújvárosi Főiskola Informatikai Intézet*
E-mail: honfivid@mail.duf.hu

*** *Dunaújvárosi Főiskola Informatikai Intézet*
E-mail: kiru@mail.duf.hu

**** *Dunaújvárosi Főiskola Informatikai Intézet*
E-mail: mkollar@mail.duf.hu

[1] Mészáros K. (2005): *SAP R/3 ismeretek*. Dunaújváros: Főiskolai Kiadó. .

[2] Mészáros K. (2005): *Logisztikai folyamatok az SAP R/3-ban - MM*. Dunaújváros: Főiskolai Kiadó.

[3] Mészáros K. (2006): *Logisztikai folyamatok az SAP R/3-ban - SD*. Dunaújváros: Főiskolai Kiadó.

[4] Mészáros K. (2005): *Operatív vállalatirányítási rendszerek bevezetése*. Dunaújváros: Főiskolai Kiadó.

1. ERP systems in teaching

On the IKT2012 Conference of Modern Techniques of Informatics we presented a short overview about Teaching of ERP systems at the College of Dunaújváros in the last half and other decade. Now we extend it with some additional analyises and the data of last two semesters.

The teaching of ERP systems is running now at the Computer Engineering, Business Informatics, Business Administration and Engineering Business Management of bachelor courses of College of Dunaújváros, furthermore at Logistic Technical Manager Assistant higher vocational training. The first higher educational training program was the Economics where the ERP teaching was launched. From the spring of 2001 the system of SAP R/3 4.6C DFI system was applied on its Specialisation of Business Informatics within the students had been learning about basis of the systems, about the modules of Controlling and Financials and the basics of Materials Management [2] and Sales and Distributions. [3] Furthermore they had been learning the ASAP methodology which was frequently asked at the final exams as well. Four lecture notes were written in series by Károly Mészáros at all. in order to learn the basics [1] and theory of SAP. These notes give excellent theoretical basics but the practical chapters e.g. the direct handling of the system are very poor and the practices and case studies are missing. At the fourth part of the lecture note series – the Start-up of operative ERP systems which introduce the students with the ASAP methodology [4] – are missing the case studies of real enterprises as well very much which could help the students to prepare themselves for the final exams questions in this topic.

The next course of the applying courses of ERP teaching was the Logistic Technical Manager Assistant Higher Vocational Training where in the subject of Logistic Informatics the students had been introduced to the basics and materials management of SAP based on Logistic informatics book written by Edit Vértes in the spring semester 2007 and 2008. In this time the Microsoft Dynamics Axapta was introduced at the College of Dunaújváros as well but only for demonstration, without any practice opportunity.

The first class of the engineering business management BSc program accredited in 2006 started the learning of ERP systems within the frame of specialisation in the academic year of 2008-2009 and had introduced to the SAP R/3 Materials Management on the Logistics Spec. and to the SAP R/3 Production Planning on the Production Management Spec. Beside the aging R/3, they had introduced to the SAP Business Suite and to the SAP ERP by Tamas Herger senior consultant of SAP Hungary Ltd. but there was no opportunity to get practical experience. Parallels they had learned the SAP Business One 2007A basic system of small and medium sized enterprises, which had been applied in the foundation subject for the BSc programs of Informatics and Engineering Business Management and had replaced of SAP R/3 in the Higher Vocational Program of Logistic Technical Manager Assistant Logistic Technical Manager Assistant. It has to be mentioned the scope and complexity for SME's for the benefits of the SAP B/1 which allows the self-practice of full logistic process of customer service through the development of own case-studies. This leads the students through from the item master data and bill of materials, the business partner master data and business opportunities and forecast, the MRP aided procurement and production containing the stock management and order picking to the delivery, billing, cash flow accounting till the special financial analysis of process elements and the general ledger.

This practice method is applied in the other ERP systems of the College in DFI SAP R/3 and in IDES provided by ELTE SAP Competence Centre but only on smaller scope and higher level of detail because of the higher complexity of these systems primary on materials and sales management without financials and controlling.

In the academic year of 2007–2008, so abbreviated VIR A and VIR B subjects had been launched in BSc programs Computer Engineering and Business Informatics with the topics of operation and enterprise modelling. Furthermore in the academic year of 2011–2012 and 2012–13 have been launched the subjects of Basics of SAP, Operating of SAP and programming of SAP for one semester. The application of the above mentioned IDES system of ELTE had been started in the academic year of 2011–2012 – after several attempting to contract – for 3 semesters replaced the DFI SAP R/3 4.6C which is still operating on its original 2001 vintage hardware. Unfortunately the SAP AG has withdrawn the license of service rights of SAP training systems worldwide and as well from ELTE, and only the SAP University Alliance training license is open today which is much more expensive and the college cannot afford it. So in 2013 the College started to examine the applying of Microsoft Dynamics NAV which may be a notable and growing competitor of the SAP Business One and SAP Business by Design on the ERP market of SME's.

[5] Mosaic Business System Kft. (2010): *SAP Business One Oktatási anyag*. Budapest.

2. Comparison of the applicability of the applied ERP systems in the education

2.1. SAP R/3 4.6C

Subjects relating to the SAP R/3 system is always surrounded by great expectations among the students, as it is well known among them the very high SAP experts' fees. However, the use of the system requires serious knowledge, the user interface is significantly different from that of standard office applications. In addition, the high complexity of a Company ERP system also requires that students learn detailed the business process and the organization's structure. The complexity of the DFI system could be said to excessive because it is the copy of a previous test environment of Dunaferr Co. which makes difficult to overview the system and to understand the essence of ERP systems and thus to give a reusable knowledge to the students. In the practical exercise it has to get a strong focus on the filtering, sorting and searching tasks and the applying of the various department codes.

2.2. SAP Business One 8.751–8.8

The easiest job – for both instructors and students – is the work with SAP Business One system. Installation, maintenance, preparation of individual reports and the programming is very easy, especially learning applying this software. For the installation of the educational system it is suitable to use an MS SQLServer Express which has a very compact size to save disk capacity and installation time on a small average class server as well. Then it can be installed the service manager and the server in less than an hour, and the "fat" client do not resist to cloning on the clients-side installation. The only problem could be with Office integration and the basic Add-Ons, but their lack does not cause problems in the education. Mosaic Business System Ltd. made available the Plasticware Ltd. training system [5], but the for practice there are available a Computer Sales small business with American and Hungarian localization preconfigured and uploaded with some historical data by the software manufacturer SAP. The interface is very similar to the SAP R/3 GUI page, but it is much cleaner and easier to learn the functions of the GUI controls.

The user interface is very friendly and almost all of them look the same transaction. The ease handling supported by "one-push" function to switch between the query (search) and

add (modify) mode. It is also a useful function at creating a particular record or process elements to use data of the source (base) records/documents. In addition, the "Base documents" and "Target documents" buttons visualize the existence of the base record/document or the record of the following process element of the actual record.

Another useful function has got the Navigator yellow arrow which related to the ID of the displayed record and helps to open it in the proper form. These features are implemented in most of the ERP systems, but in SAP Business One are most consistently positioned and applied allowing fast navigation between tables and fast retrieval of related records, and thus understanding the students the essence of ERP systems. The sorting, filtering, search methods and selections are little bit specific, but they are easy to learn. The help.sap.com online help provides for excellent help in self-study for both Hungarian and foreign students, because this help is available in many languages, for example Chinese, Russian, Turkish, Dutch and Finnish used by the foreign students of the college of Dunaújváros.

2.3. SAP ERP ECC6.00

Due to the withdrawal of the service right just only in this semester can we use one of the latest SAP education system modeling IDES Company in teaching of the fields of application, operating, customization and development. The learning materials and the exercises made by Institute of Informatics of College of Dunaújváros based on original English teaching materials of mySAP ERP Business Processes in Financial Accounting and Management Accounting, of Business Intelligence and of ERP Financials. In addition there were available the All-in-One Package Best Practises in English. The system help was poor English help, which was completed at previously mentioned help.sap.com online help but available only in English and German.

2.4. MS Dynamics NAV

At getting started with the Microsoft system has been brought to our attention that, due to the essential characteristics of this system the difference from other systems is that it does not post the transactions automatically, because the client software send a field modification immediately when the user leaves a field. This results a frequent network communication but the possibility of data loss is almost zero at inserting or modifying a record. But the posting should be done manually. This feature is extremely useful for understanding the accounting and controlling. Therefore the first lessons in Dynamics NAV teaching are the topics of dimensional analysis and accounting issues.

The interface shows the image of the Microsoft applications, which is slightly confusing if somebody has been used other type of ERP systems and/or Microsoft Office applications at the same time. But the same features can be found here as in company's and SME's ERP systems, so you can easily drill down or create target documents based on previously created documents. But it is very hard to find a document of a previous process element, however it is is easy to navigate between the tables. In addition, a complete production management functionality is available, which was missing from the SAP Business One and is available as an "AddOn". So it can completely replace the applying SAP R/3 in Production Planning. [6] The software development is very simple, but although it is easy to use, the development process is long because only basic techniques are available for database access. But this is an opportunity for both of It Engineering and Economics Informatics students to get zest to the NAV programming by some simple programming exercises which are close the students.

[6] Navision Support Online Kft. (20013): *Microsoft Dynamics NAV oktatási anyag.* Dunaújváros: Főiskolai Kiadó.

[7] Louis Columbus: 2013 ERP Market Share Update: SAP Solidifies Market Leadership, A Passion for Research – Focusing on the intersection of technology and trust – Blog at WordPress.com, on May 12, 2013, URL: http://softwarestrategiesblog.

2.5. Final question

We can take the final question that is it a step back in the education to replace a corporate ERP system with to MS Dynamics NAV. We can say that the answer is yes from the point of view of the complexity of system architecture, organizational structure and process modelling point. But if we take the financial situation of the college the cost- efficiency, the educational purpose and applicability into the consideration then it appears to be a good choice. Although Microsoft has a much smaller reputation in the market of ERPs than the market leader SAP's, and in 2012, its market share was quint of the SAP's share 25% [7], today has a rapidly, expanding market in Hungary especially among small and medium businesses. So it would be a good choice for our students. And for the curious students there is available the good old DFI SAP R / 3 4.6C. Our experiences show that if a student learn the use of one type of ERP systems, and understand the purpose and structure, can easily use any other type or size of an ERP system. The jobs at the large companies mostly require much narrower range of functional areas than the learned functional areas and only at small and medium-sized enterprises can occur that more functional areas are used by same member of the staff.

3 Analysis of the effectiveness of ERP education

For over a more than a decade, a total amount of about 1,638 foreign and Hungarian students got to know with ERP systems at the College of Dunaújváros in three (nowadays two) responsible Institutes in the frame of ten subjects (Table 1, which includes the failed exams and unfulfilled courses as well. The former Institute of Economics and Management and Enterprise Sciences designed the topics of the ERP subjects primarily for the purpose of business, logistics and production management applications and additionally from 2008 take into consideration the requirements of small and medium-sized enterprises. The Institute of Information Technology is aimed at the operating and development areas.

Table 1. Attendants of ERP subjects (2000–2013).

Responsible Institute	*Department for Economy Sciences*		*Department for Enterprise Management Sciences*				*Institue for Infomatics*				
Subject	*SAP applications*	*SAP base system*	*Logistic information systems*	*SAP Business ONE*	*SAP logistic application project*	*SAP production planning application project*	*SAP basics*	*SAP operating*	*ERP systems A*	*ERP systems B*	*TOTAL*
Economy BA	285	285		1							**571**
Economy and Business administration BA				7							**7**
Business Informatics BSc							113	31	382		**526**
Logistic Technical Manager Assistant vocational program			151	128							**279**
Computer Engineering BSc							90		549	317	**956**
Engineering Teacher – Computer Engineer MSc									3		**3**
Computer Engineering Assistant vocational program							26				**26**
Engineering Business Management BSc				98	237	77			7		**419**
TOTAL	**285**	**285**	**151**	**234**	**237**	**77**	**229**	**31**	**941**	**317**	**2787**

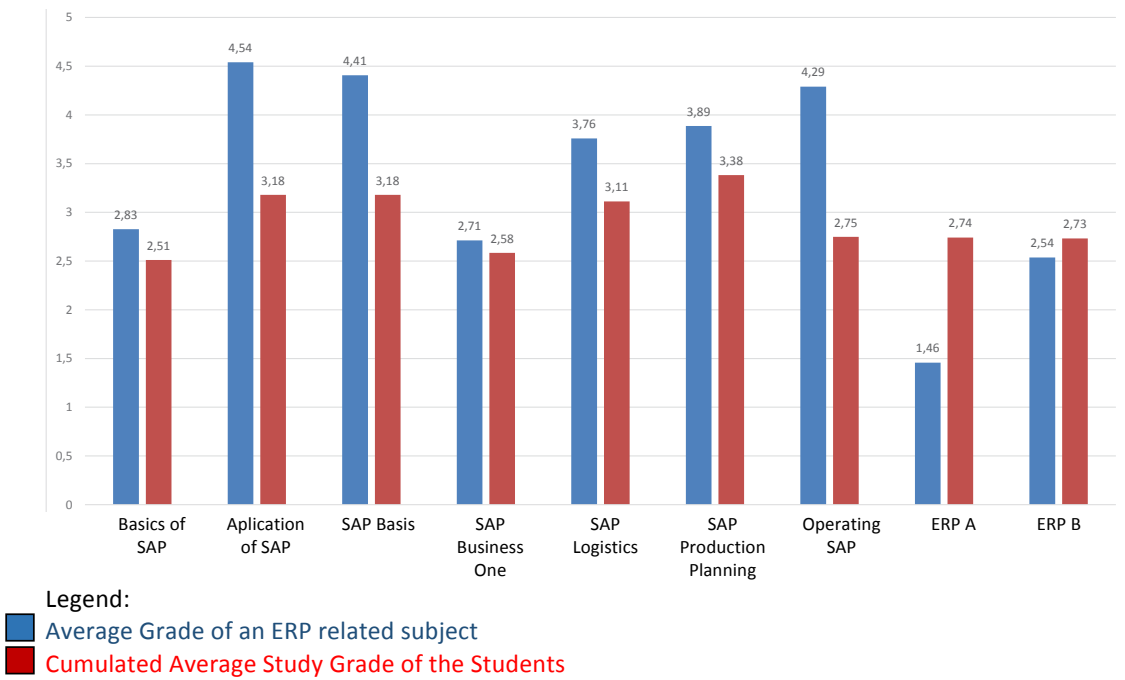
Source: own edition based on the data of Neptun Education Administration System of College of Dunaújváros

For the analysis, we collected the student grades of the ERP courses from the Neptun Education Management System and the grade averages of these students. If we compare the whole grade average of a student with the grade of ERP related subject, we can evaluate the effectiveness of the courses using a system type without the influencing the result by the diligence and the ability of the student.

Chart 1 presents the grades of the ERP related subjects, which show that the students have performed the ERP courses with higher grades that their Cumulated Average Study Grade. This is a good result although the ERP subjects are much more complicated because of the complex enterprise management and logistics aspect.

* The courses of Logistic Information Systems are not involved, because only a part of its topics were about SAP R/3.

Chart 1. Average Grade of each ERP related subjects in comparison with the Cumulated Average Study Grade of the Students in Period 2000–2014.



Source: own edition based on the data of Neptun Education Administration System of College of Dunaújváros

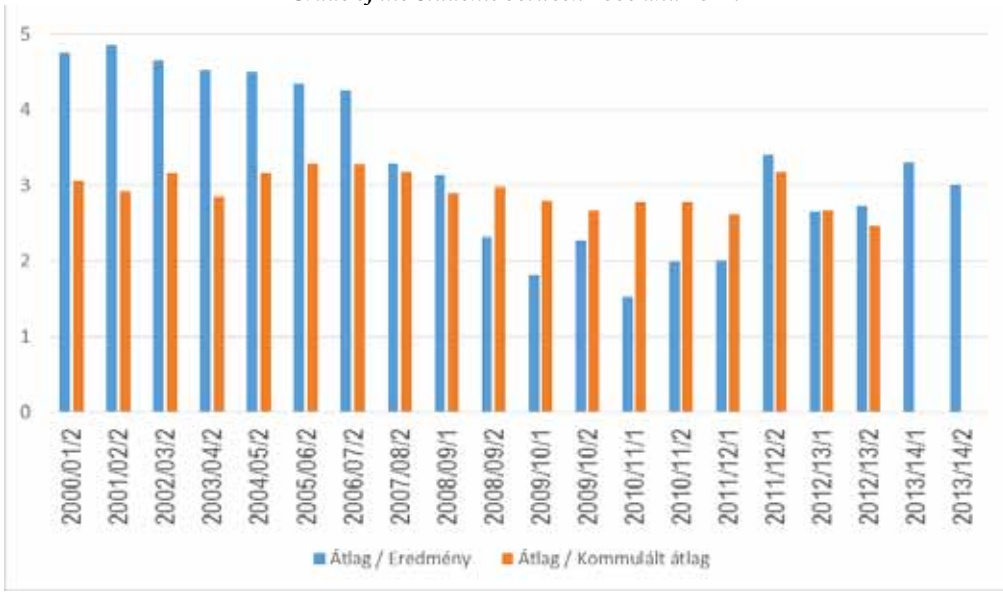
But the Chart 2 presents that the average results in a semester from 2001 to 2014 decreases. Why? Maybe in changing of the ERP system applied for teaching. But there is a stronger correlation between changing grades and starting new education programs:

– In Semester II of 2007/2008 the first ERP courses for IT engineer Students and Business Informatics Students was started – this is the first big fall.

– In Semester II of 2008/2009 the first ERP courses for Technical Manager Students – this is the second big fall.

– At last, in Semester I of 2010/2011 was moved the SAP Business One for Logistic Technical Manager Assistant, which was matched by releasing new teaching materials. (The original teaching materials of Mosaic Business System Ltd were created for enterprise application, where the involved employees know the business processes in practise.)

Chart 2. Average Grade of all ERP related subjects in a Semester in comparison with the Cumulated Average Study Grade of the Students between 2000 and 2014.



Legend:

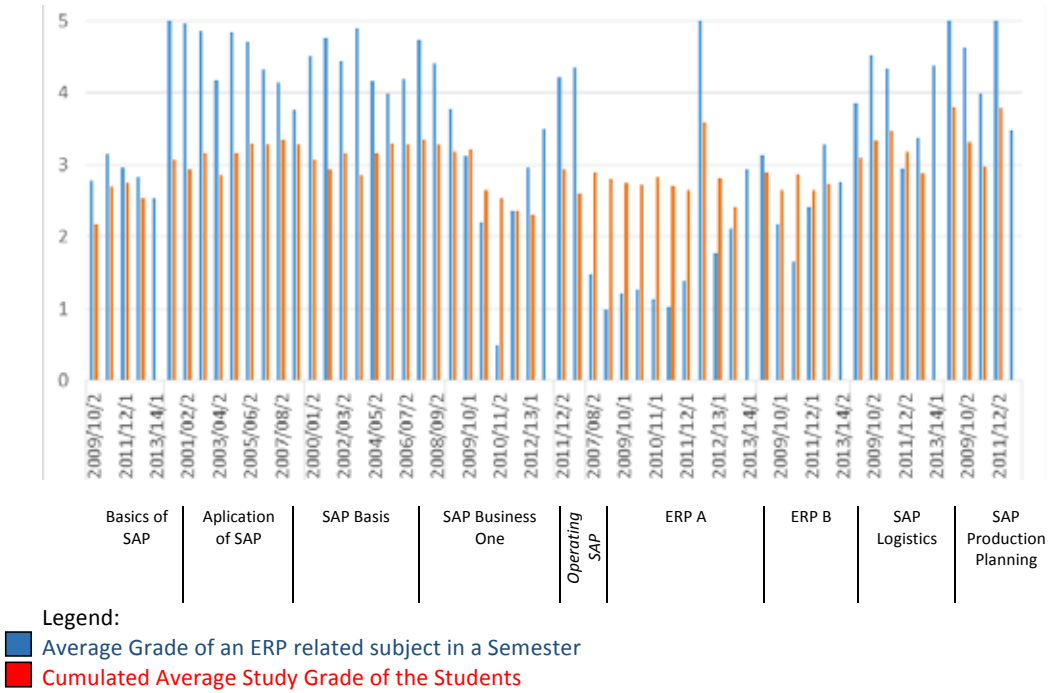
Average Grade of all ERP related subject in a Semester

Cumulated Average Study Grade of the Students except last 2 Semesters

Source: own edition based on the data of Neptun Education Administration System of College of Dunaújváros

The Cumulated Average Study Grade of the Students except last 2 Semesters we could not received so these the changes of grades in these semesters are not comparable with it. Furthermore, if we analyse the Average Grade of all ERP related subject in a Semester on Chart 3, then we can spectate that the Operating SAP and ERP A, ERP B courses in the bachelor program of IT Engineering decrease the average value of the grades. These subjects are very complex as the SAP Business Suite is. Therefore one of the development was to change the structure of these subject. Furthermore these subjects could be sliced for more courses and it could be also important to raise the rate or number of the laboratory practice. The demand by enterprises for ERP IT knowledge is growing on and on and there are more and more ERP systems operating at both of small- and medium sized enterprises and large companies.

Chart 3. Average Grade of each ERP related subject in a Semester in comparison with the Cumulated Average Study Grade of the Students between 2000–01 and 2013–14.



Source: own edition based on the data of Neptun Education Administration System of College of Dunaújváros

4. The experiences of teaching ERP systems with Microsoft Dynamics NAV in Academic Year 2013/14

In the first Semester of Academic Year 2013-14., we introduced the Microsoft Dynamics NAV at the practical courses in the bachelor program of IT-engineering and in Hungarian and English Technical Management. We spectated that the students are disappointed because of changing the SAP to NAV, so in the lessons, we have to discuss the comparison of the systems in beneficial and application aspect first.
The other experience is that the students have to reinforce their knowledge about the general ledger and financial and managerial accounting.

The students are working in the same database and company as in previous semesters, but because the NAV does not commit a business transaction at the entry, only after posting, some students have overwritten the entries of eachother making some disruption in processing the case studies. But on other hand they could experience the responsibility of the work with ERP systems.

Conclusion

As a conclusion we can determine, that the College of Dunaújváros was cutting-edge in teaching ERP systems, but today the financials limit the licensing both of large company ERP systems and ERP systems of small- and medium-size enterprises, so the range of the ERP systems in teaching is not enough wide. In the near future the Institute of Information Technology is going to renew the teaching with cooperation of industrial and IT partners in the field of mobile communication, business intelligence and memory-based databases.

About the development of education in the near future, we can say, that because of the wide functionality of the ERP systems, we could teach these systems not only in ERP courses but the in related finance, taxation, controlling and HR management courses. It could be advisable because the ERP courses could be very short to practise and understand the wide functionality deeper.

Galéria

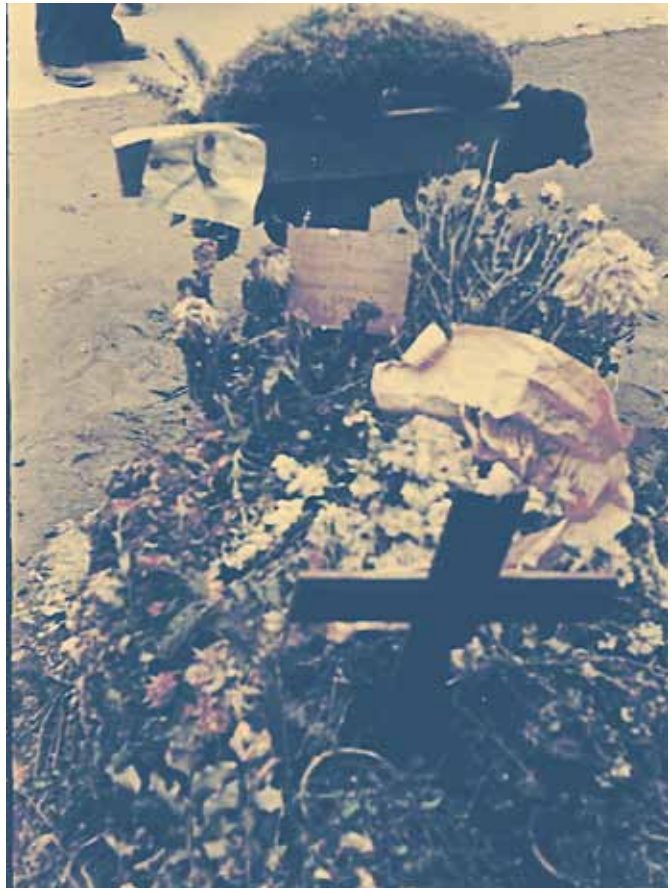
Ismeretlen szerző fotói


VIRÁG KÖZEP


10


NIS FAMILIA
USOK


085




409
443