Tények, érdekességek az informatika világából

Közmondások programnyelven

- */* A hazug embert hamarabb utolérik, mint a sánta kutyát */*
  - `capturetime(human.type(LIAR)) < capturetime(dog.type(CRIPPLE))`
- */* Kerülgeti, mint macska a forró kását */*
  - `sideStep(cat.getWalkType(new Kása(HOT)));`
- */* Aki másnak vermet ás... */*
  - `Stack.push(someOneOther.getStack().madeBy());`
- */* A napra lehet nézni de rá nem */*
  - `SUN.CanView := true;`
  - `HE.CanView := false;`
- */* Amilyen az adjonisten, olyan a fogadjisten */*
  - `setAcceptGod(getGiveGod());`
- */* Madarat tolláról, embert barátjáról */*
  - `Bird.Type := Bird.feather;`
  - `Human.Type := Human.friend;`
- */* Éhes disznó makkal álmodik */*
  - `pig.setType(HUNGRY);`
  - `pig.setDream(MAKK);`
- */* A részvétel a fontos... */*
  - `Winnig.Priority := 0;`
  - `Attendance.Priority := CONST_HIGH;`
- */* A szomszéd kertje mindig zöldebb */*
  - `const bool compareGreenness(Grass* grass)`
  - `{`
  - `if(grass.getOwner() == NEIGHBOUR) return true;`
  - `}`
- */* Lassan járj... */*
  - `PassedDistance := PassedDistance + (1/WalkSpeed);`
- */* Okos enged, szamár szenved */*
  - `if Human.Type = CONST_SMART then Release;`
  - `if Human.Type = CONST_DONKEY then Suffer;`

Fotorealisztikus számítógépes grafika

A *generatív számítógépes grafika* a képi információ tartalmára vonatkozó adatok és algoritmusok alapján modelleket állít fel, képeket jelenít meg (*renderel*). Ide tartozik a speciális effektusok előállítása, vagy az animáció is, amely a generált grafikát az időtől teszi függővé. Általában két- (2D) vagy háromdimenziós (3D) grafikus objektumok számítógépes generálását, tárolását, felhasználását és megjelenítését fedi a fogalom.

Nyilvánvaló, hogy az ember által készített mesterséges objektumok könnyűszerrel modellezhetők fotorealisztikusan számítógépen, hisz nem egy már eleve számítógép segítségével volt megtervezve. A nagy kérdés a természet alkotta tájak, élőlények, kövek, sziklák stb. modellezése. Ebben nagy segítségünkre vannak a fraktálok.

A *fraktál*ok *önhasonló*, végtelenül komplex matematikai alakzatok, amelyek változatos formáiban legalább egy felismerhető (tehát matematikai eszközökkel leírható) ismétlődés tapasztalható. Az elnevezést 1975-ben Benoît Mandelbrot adta, a latin *fractus* (vagyis törött; törés) szó alapján, ami az ilyen alakzatok tört számú dimenziójára utal. „A természet geometriájának fraktál arculata van." – vallotta Mandelbrot.

1. Általános követelmények

Fotorealisztikus képek előállításának általános követelményei ([1.] alapján):

- *Térhatás (depth cueing)*: A 3D-s modelltér jelenete a 2D-s raszteres képen is térhatású legyen. Érvényesüljön a perspektivikus ábrázolási mód. Reálisan ábrázoljuk a tárgyak látható és nem látható éleit, felületeit. Érvényesüljön a mélység-élesség. A messzeségbe tűnő objektumok legyenek elmosódottabbak, kevésbé kidolgozottak. Használjuk a *mip-maping* technikát.
- *Felületek megvilágítása, tükröződés, árnyékok*: modellezzük és használjuk fel a természetben is lezajló jelenségeket. A képeken a fényhatások feleljenek meg a természet és a fizika törvényeinek. A természethűség érdekében használjunk természetes (természetutánzó) textúrákat. Érdes, göröngyös térhatású felületeket tudunk elkészíteni a *bump-maping* technikával, amikor a felületre merőlegesen véletlenszerűen módosítjuk a tárgy felszínét: kiemelünk, lesüllyesztünk. A testek egymásra vetett árnyékait meg kell jeleníteni.
- *Átlátszóság, áttetszőség, köd, füst modellezése*: figyelembe kell venni a fénytörést, a fény intenzitásának csökkenését. Használjuk az *alpha-blending* technikát.
- *Textúrák alkalmazása*: a valósághűség érdekében fényképeket, ábrákat tudunk ráhúzni az egyes grafikus objektumokra.

Mindezeken az ábrázolási lehetőségeken, követelményeken túl, vizsgáljuk meg, milyen algoritmusok segítségével lehet előállítani a megfelelő természetes objektumokat, itt elsősorban felhőkre, domborzatra, vízre, fákra gondolunk. Megjegyezhető, hogy a nem természetes, mesterséges objektumok nagyon egyszerűen előállíthatók fotorealisztikusan, hisz az utóbbi években ezek megtervezése CAD eszközök segítségével történik (pl. épületek, bútorzat, lámpatestek, autók stb.), amelyek már eleve képesek arra, hogy fotorealisztikus látványtervet készítsenek a modellről.

2. Felhők generálása

Egy kép megalkotásakor elsődleges szempont a háttér létrehozása. A szabadban ez gyakran egy felhős égboltot (is) jelent.

A valóságmodellezéskor is nagy szerephez jutnak a véletlen fraktálok, hisz a természet alkotta valós objektumok nem teljesen szabályosak.

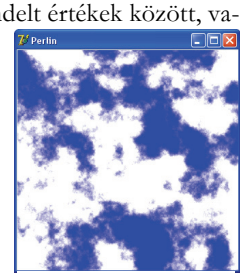
A véletlen fraktálok vagy véletlen halmazokból veszik fel értékeiket, vagy egy generált véletlen-számmal perturbáljuk a fraktál értékét, vagy valamilyen más szinten kötődnek a véletlenhez, pl. a Brown-féle mozgás pályájának a fraktál jellegű tulajdonságait használjuk fel.

A valóság modellezésében felületeket, felhőzetet, atmoszférikus effektusokat stb. nagyon jól elő tudunk állítani *Perlin-zaj* [2.] alkalmazásával.

Perlin zajfüggvénye R^n-en értelmezett ($f : R^n \to [-1,\ 1]$), az egész számokban csomópontokat képző rácshoz igazított pszeudo-véletlen spline függvény, amely a véletlenszerűség hatását kelti, de ugyanakkor rendelkezik azzal a tulajdonsággal, hogy azo-

nos bemeneti értékekre azonos függvényértéket térít vissza. A gyakrabban használt *n* értékei 1 – animáció esetén, 2 – egyszerű textúrák, 3 – bonyolultabb 3D textúrák, 4 – animált 3D textúrák (pl. mozgó felhők).

A következőképpen generálhatunk Perlin-zajt: adott egy bemeneti pont. Minden környező rács-csomópontra választunk egy pszeudo-véletlen értéket egy előre generált halmazból. Interpolálunk az így megkapott csomópontokhoz rendelt értékek között, valamilyen *S* görbét használva (pl. $3t^2 - 2t^3$).



1. ábra
Felhőzet Perlin-zajjal

Ha a Perlin-zajfüggvényt kifejezésben használjuk, különböző procedurális mintákat és textúrákat hozhatunk létre.

Ha ezeket a kifejezéseket fraktál-összegben használjuk, minden iterációban új adatot vihetünk be, amely valamilyen módon befolyásolja a teljes képet. Például domborzat generálás esetén, az iteráció során a fraktál dimenzióját akarjuk befolyásolni, azaz minden iterációban az amplitúdót osztani fogjuk egy bizonyos értékkel.

A gyakorlati kísérletek azt mutatják, hogy a Perlin-zajfüggvény a következő együttható-értékekre ad fotorealisztikus felhős égboltot:

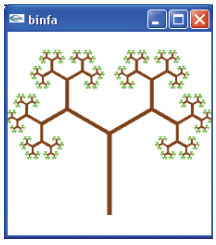
```
1.    r1 := 1000+Random(10000);
2.    r2 := 100000+Random(1000000);
3.    r3 := 1000000000+Random(2000000000);
```

3. Fák, bokrok generálása

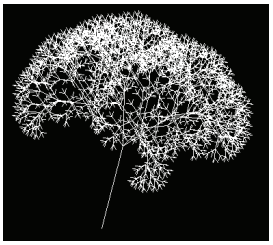
A távolban lévő fák, növényzet előállítható egyszerűen *bináris* vagy *kvadrális fák* segítségével, vagy *Barnsley-féle páfrányok* segítségével.

A barna törzsű fákat akár levél-szinten zöldre is színezhetjük, vagy egy perturbáló faktor segítségével szétrázhatjuk az ágaikat, mintha szél fújta volna meg őket. A páfrányokat IFS segítségével állíthatjuk elő.

Az IFS az *Iterated Function System* (iterált függvényrendszer) kifejezés rövidítése. Egy IFS nem más, mint kontraktív, $R^2 \rightarrow R^2$ alakú transzformációk kollekciója, mely szintén egy leképezés. Az ilyen típusú leképezéseknek mindig van egy egyedi fixpontja, digitális képekre alkalmazva ez a fixpont általában egy fraktálkép.



2. ábra. *Bináris fa*



3. ábra. *Véletlen perturbáció alkalmazása kvadrális fánál*

A Barnsley-páfrányt [3.] úgy állíthatjuk elő IFS-ként, hogy kiindulunk az origóból ($x_0 = 0, y_0 = 0$), kirajzoljuk a pontot, majd véletlenszerűen alkalmazunk egy transzformációt a következő négyből (pl. 300 000-szer), a kapott új pontokat kirajzoljuk:

1. $\begin{cases} x_{n+1} = 0 \\ y_{n+1} = 0,16 \cdot y_n \end{cases}$, ezt a transzformációt 1%-os valószínűséggel alkalmazzuk.
2. $\begin{cases} x_{n+1} = 0,2 \cdot x_n - 0,26 \cdot y_n \\ y_{n+1} = 0,23 \cdot x_n + 0,22 \cdot y_n + 1,6 \end{cases}$, 7%-os valószínűséggel.
3. $\begin{cases} x_{n+1} = -0,15 \cdot x_n + 0,28 \cdot y_n \\ y_{n+1} = 0,26 \cdot x_n + 0,24 \cdot y_n + 0,44 \end{cases}$, 7%-os valószínűséggel.
4. $\begin{cases} x_{n+1} = 0,85 \cdot x_n + 0,04 \cdot y_n \\ y_{n+1} = -0,04 \cdot x_n + 0,85 \cdot y_n + 1,6 \end{cases}$, 85%-os valószínűséggel.

Ha a fák vagy bokrok az előtérben – tehát közel helyezkednek el, jóval bonyolultabb algoritmusokkal tudjuk ezeket fotorealisztikussá tenni.

Ezek az algoritmusok a fa természetes növekedését követik, véletlen perturbálófaktorok alkalmazásával, a törzs textúrázásával, az ágak levelekkel való ellátásával együtt. Minden egyes levél hű mintázata a természetes leveleknek.

Az egyik módszer a *graftál*ok alkalmazása. A graftálok egyszerű szabályokból iteratív eljárással létrehozott alakzatok, amik a növényeket modelleznek.

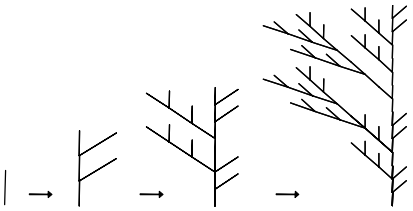
4. ábra. *Barnsley-páfrány*

Példa graftálra:

```
1. Legyen egy négy jelből álló nyelv: 0, 1, [, ].
2. A [-t mindig követi egy ], a ] előtt mindig áll egy
   [.
3. A [ ] páros között egy vagy több jel is állhat.
4. A 0 és 1 jelentése: lépj előre egy egységnyit.
5. A [ jelentése: jegyezd meg az aktuális pozíciót és
   irányt, majd fordulj el meghatározott szöggel.
6. A ] jelentése: menj vissza és fordulj a legutóbb
   megjegyzett pozícióba és irányba.
```

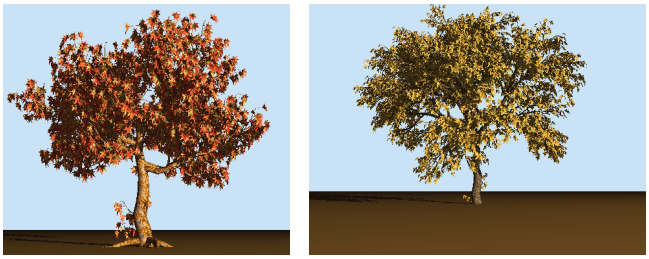
„Életet” egy graftálba kicserélési szabályok alkalmazásával lehelhetünk. Például:

```
1. Cseréljünk ki minden 0-át 1[0]1[0]0-ra.
2. Cseréljünk ki minden 1-et 11-re.
```



5. ábra. *Graftál „növekedése”*

Fotorealisztikus fa előállítási algoritmusokat ír le Gilles Tran [5.]. Ezeket próbáltuk meg továbbfejleszteni és úgy paraméterezni, textúrázni, hogy általános fákat lehessen velük előállítani.



6. ábra. *Fotorealisztikus fák*

4. Vízfelület, hegyes táj, domborzat generálása

A domborzat modellezése a virtuális valóság és a fotorealisztikus grafika egyik fontos alkotóeleme.

Az egyik legsikeresebb domborzat-modell a fraktál domborzat-modell, amelynek az alapja szintén a Perlin-zaj [6.].

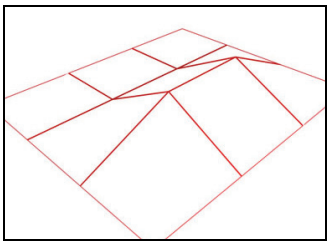
A fraktál domborzat-modell létrehozásához négy elem szükséges:

- egy alapfüggvény, amely megadja a domborzat alakját (Perlin-alap),
- a fraktál dimenziója (az amplitúdó módosulása minden iterációban),
- az oktávok (iterációk) száma,
- a frekvencia módosulási tényezője.

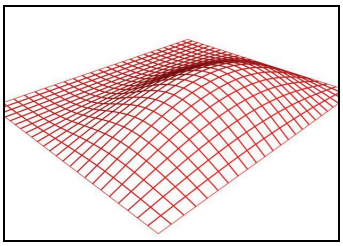
Az algoritmusban a Perlin-zaj első iterációja dönti el, hogy az adott pont magasság szerint milyen tájegységhez tartozik, majd az iterációs lépésekben, a tájegységnek megfelelő amplitúdó és frekvencia változás paramétereit alkalmazzuk. Például hegyek esetén az amplitúdó kis változást kell, hogy eredményezzen a fraktálösszegben, míg egy fennsík esetén az amplitúdónak egyből redukálnia kell a részletét, hogy ezt egy sima felszínné alakítsa.

Domborzatot kétféleképpen állíthatunk elő: *szimulálás* és *szintetizálás* segítségével.

A szimulálás azt jelenti, hogy létező adatok alapján készül a modell (véges adatmennyiség); a szintetizálás pedig azt, hogy a természetben előforduló szabályosságok alapján állítunk elő virtuális modelleket.



7. ábra. *Szimulálás: maximális közelítés véges adathalmazból szimulált domborzaton (GPS)*

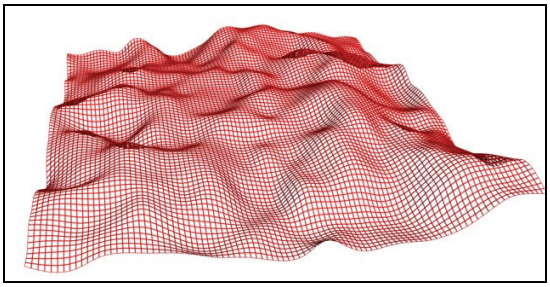


8. ábra. *Szintetizálás esetén, nincs „maximális” közelítés, ugyanis a domborzatot leíró eljárások, mindig generálnak új adatot számunkra*

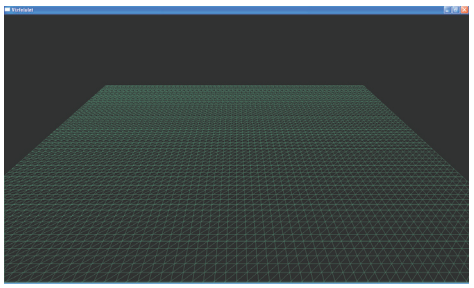
Algoritmus felületgenerálásra:

1. Adott egy bemeneti pont.
2. Minden környező rács-csomópontra választani kell egy pszeudo-random értéket egy előre generált halmazból (mivel a csomópontok koordinátái egész számok, ezeket használjuk az eredmény kiválasztására).
3. Majd interpolálni kell az így megkapott csomópontokhoz rendelt értékek között, valamilyen S görbét használva. (pl. 3t2-2t3).
4. Ha ezeket a kifejezéseket fraktál összegben használjuk, minden iterációban új adatot vihetünk be a képbe, amik valamilyen módon befolyásolják ezt.
5. Domborzat generálás esetén, az iteráció során a fraktál dimenzióját akarjuk befolyásolni, azaz minden iterációban az amplitúdót osztani fogjuk egy bizonyos értékkel.

Vízfelszín modellezésére is kiválóan alkalmas a Perlin-zaj, itt azonban szem előtt kell tartanunk a különböző fizikai törvényeket is, például a hullámzás megvalósítására. Vízfelszín létrehozására elkerülhetetlen az animáció használata, éppen ezért a gyorsaság és hatékonyság növelése érdekében jobb ezeket az algoritmusokat valamilyen hardver által támogatott árnyaló nyelvben megírni.



9. ábra. *Alap domborzatmodell*

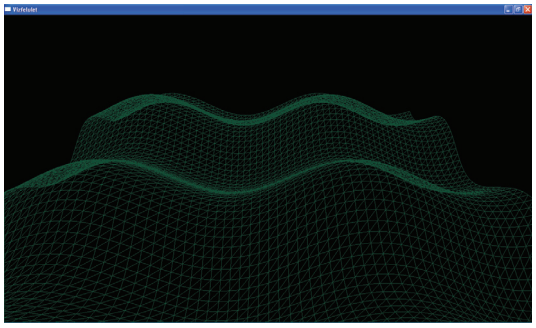


10. ábra. *Alap vízmodell*

Az animáláshoz használt CG program:

```
1.  struct appdata
2.  {
3.      float4 position: POSITION;
4.      float4 color: COLOR0;
5.      float3 wave: COLOR1;
6.  };
7.
8.  struct vfconn
9.  {
10.     float4 HPos: POSITION;
11.     float4 Col0: COLOR0;
12. };
13.
14. vfconn main(appdata IN, uniform float4x4
        ModelViewProj)
15.
16.     vfconn OUT;
17.     // szinusz hullámok
18.     IN.position.y = (sin(IN.wave.x +
        (IN.position.x / 5.0) ) + sin(IN.wave.x +
                                IN.position.z / 4.0) )
        ) * 2.5f;
19.     OUT.HPos = mul(ModelViewProj, IN.position);
20.     OUT.Col0.xyz = IN.color.xyz;
21.     return OUT;
22. }
```

Foster és Fedkiw [7.] olyan szimulációs módszert dolgozott ki, amelyben egy folyadék térfogatát egy implicit φ függvény körvonala határozza meg A víz felülete: $\varphi = 0$, a $\varphi \leq 0$ a vizet, a $\varphi > 0$ a levegőt jelenti. Az implicit függvény ábrázolása egy ideiglenesen koherens, finom, egyenletes vízfelszínt eredményez.



11. ábra. *Animált vízfelület*

Ez az implicit felület időben és térben dinamikusan alakul, a folyadék $\vec{u}$ sebességének függvényében. Osher és Sethian szerint [8.] az egyenlet: $\varphi_t + \vec{u} \cdot \nabla\varphi = 0$, ahol φ_t a φ függvény idő szerinti deriváltja, és ∇ a gradiens operátor: $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$.

12. ábra
Fotorealisztikus táj – fraktálok segítségével

5. Szoftverek

A fentieket szem előtt tartva, számos olyan grafikus motor létezik, amelyek segítségével fotorealisztikus számítógépes grafikákat tudunk előállítani mind statikus, mind pedig animált változatban. Most kettőt emelnék ki ezek közül, az egyik a POV-Ray, a másik a Unity.

A POV-Ray (Persistence of Vision Raytracer) egy szabadon terjeszthető (freeware) programcsomag, mely segítségével egy formális nyelven a modelltérben (3D lebegőpontos világ-koordinátarendszer) definiált 3D objektumokról fotorealisztikus képeket tudunk készíteni. A POV-Ray David Buck eredeti raytracerelőjére (sugárkövető algoritmus) épül, melyet állandóan tovább fejlesztenek. Létezik Windows, Linux, Mac stb. POV-Ray verzió, a modellek elkészítéséhez pedig több OpenSource modellező is létezik.

POV-Ray példa: `A Sphere {<0, 0, 0>}, 1` egy gömböt határoz meg.

A bonyolultabb testeket primitivekből tudjuk összerakni. A primitivek beépített építőelemek: gömb, henger, kúp, téglatest, torusz, sík.

A Unity egy videojáték-motor, amelyet a Unity Technologies fejleszt. A Unity segítségével háromdimenziós videojátékokat, valamint egyéb interaktív jellegű tartalmakat lehet létrehozni: építészeti látványterveket, valós idejű háromdimenziós animációkat, geometriai eszközcsomagokat stb. A szoftver nagyméretű adatbázisokat képes kezelni, kihasználni a kölcsönhatások és animációk képességeit, előre kiszámított vagy valós idejű világítást tud biztosítani. Az objektumokhoz viselkedési elemeket tudunk hozzáadni. A játékmotor folyamatosan megőrzi a végleges változat megjelenítését. Segítségével fotorealisztikus videojátékokat tudunk készíteni Windowsra, Linuxra, Mac OS X-re, Xbox 360-ra, PlayStation 3-ra, Wii-re, iPad-re, iPhone-ra, vagy akár Android alá.

A Unitynek két fő alkotó része van: az egyik játékok fejlesztésére és tervezésére használható szerkesztő, a másik pedig maga a videojáték-motor, amely a végleges változat kivitelezésében nyújt segítséget.

Könyvészet

[1.] BUDAI Attila: *A számítógépes grafika*, LSI Oktatóközpont, Budapest, 1999.
[2.] PERLIN, Ken: *An Image Synthesizer*, In: Computer Graphics (SIGGRAPH 85 Proceedings) 19(3) July, 1985.
[3.] BARNSLEY, Michael: *Fractals Everywhere*, Academic Press, Inc., 1988.

[4.] SZIRMAY-KALOS László, ANTAL György, CSONKA Ferenc: *Háromdimenziós grafika, animáció és játékfejlesztés*, Computerbooks, Budapest, 2006.
[5.] TRAN, Gilles*: 3D art and graphic experiments*, http://www.oyonale.com
[6.] EBERT, David S.; MUSGRAVE, F. Kenton; PEACHEY, Darwyn; PERLIN, Ken; WORLEY, Steven: *Texturing & Modeling, A Procedural Approach*, AP Professional, 1994.
[7.] FOSTER, N.; FEDKIW, R.: *Practical animation of liquids*, In Proceedings of SIGGRAPH 2001, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 23–30.
[8.] OSHER, S.; SETHIAN, J. *Fronts propagating with curvature dependent speed: Algorithms based on hamiliton-jacobi formulations*. J. Comp. Phys. 79, 1988., 12–49.

Kovács Lehel István

Kémiatörténeti évfordulók

II. rész

305 éve született

Scherffer, Henrik Theophilus 1710. december 29-én Stockholmban. Az Uppsalai egyetemen és a királyi pénzverde vezetőjeként dolgozott. 1748-tól a Svéd Tudományos Akadémia tagja volt. Az 1750-es évek elején a Pinto spanyol folyó homokjából sikerült elkülönítenie a platinát, amit fehér aranynak, vagy Pintoi ezüstnek nevezett, tanulmányozta tulajdonságait. Kémiai előadásait T. Bergman adta ki, több nyelvre is lefordították. 1759. augusztus 10-én halt meg.

235 éve született

Döbereiner, Johann Wolfgang 1780. december 15-én Hofban (Németország). Münchbergben gyógyszerészeti, később bölcsészeti, ásvány- és vegytani tanulmányokat folytatott. 1803-ban szülővárosában vegyigyárat alapított. 1810-ben a jénai egyetem tanára lett, ahol haláláig dolgozott. 1823-ban egy gyújtót szerkesztett (egy hengerben cink kénsavval érintkezve hidrogént fejlesztett, mely vékony nyíláson áramlott a platinataplóra, amitől az izzásig felhevült és meggyújtotta a hidrogént). A gyufa felfedezése előtt elterjedten használták készülékét. Az elemek tulajdonságait vizsgálva megállapította, hogy azok triadokra oszthatók: a triád három elemből álló csoportjában az atomsúlyok különbsége állandó: ilyen triadok: Li,Na,K, Ca,Sr,Ba, S,Se,Te, vagy Cl,Br,I. Előállította a hangyasavat (1822). Több tankönyvet írt: Elemente der pharmacentischen Chemie (1819); Anfangsgründe der Chemie und Stöehiometrie (1826); Grundriss der allgemeinen Chemie (1828). Fiával, Ferenccel: Deutsches Apothekerbuch (Stuttgart 1840-55). Goethe barátja volt, akivel hosszan levelezett. 1849 március 24-én halt meg Jénában.

210 éve született

Graham, Thomas 1805. december 20-án Glasgowban (Skócia). Szülővárosában, Edinburgban és Oxfordban tanult. A Glasgowi Egyetem (1830), majd a Londoni egyetem (1837) tanára volt. A Kémiai Társaság első elnöke (1840). Főleg fizikai ké-