Beágyazott programozás-oktatás – Grafika

Szalayné Tahy Zsuzsanna

sztzs@caesar.elte.hu
ELTE IK

Absztrakt: Sokan vitatják, hogy kell-e közismereti, alkalmazói informatikát oktatni. Egyre többen látják, hogy probléma van az informatikai műveltséggel: felhasználók tömegei nem értik és ezért rosszul használják a szoftvereket, amelyek így nem segítik, hanem lassítják a munkát. A megoldást egyre többen a programozás oktatásában látják, ami azonban sokaknak nehéz, megtanulhatatlan tudománynak tűnik. Cikkemben szeretnék egy lehetőséget bemutatni, hogy hogyan lehet közismert grafikai szoftver használatának oktatásával, indirekt, programozást előkészíteni, majd hogyan lehet a programozáshoz szükséges készségeket és előismereteket, a programozás alapjait és az informatikai műveltséget együtt tanítani.

Bevezetés

Amikor külföldiek megkérdezik, mi a foglalkozásom, zavarba jövök. Informatikatanár vagyok, de tudom, hogy ahány ember hallja, annyiféle tartalmat társít ehhez a szakmához. Iskolám legközelebbi és legtávolabbi kapcsolatait tekintve informatikaoktatás címén:

- a székelyudvarhelyi Tamási Áron Líceumban minden diákot felkészítenek az ECDL vizsgára szakosodás esetén programozást is tanulnak a diákok [1] [2];
- a révkomáromi Sellye János Gimnáziumban irodai alkalmazásokat és programozást is tanítanak [3];
- a maláj SM Sains Seri Puteri (bentlakásos egyetemi előkészítő lányiskola Kuala Lumpurban) órarendjében nincs informatikaóra [4];
- az indiai Saint Mark's Sr. Secondary Public School (hatévfolyamos középiskola Delhiben) a Syllabus [5] alapján alsóbb évfolyamokon alkalmazásokat tanít, az utolsó években pedig programozást. Egy tanáruktól azt hallottam, hogy választható a programozás helyett alkalmazások tanulása. Projektmunkáik alapján e-újság, videó és különböző grafikai, médiakészítő szoftvereket használnak.

Ahány ország, annyiféle választ kaphatunk. Görögországban Logo-t tanítanak, a németek LEGO robotot programoznak, ami észt ismerőseimnél szakkörként népszerű… Van, ahol nincs informatikaoktatás, van, ahol az egyes tantárgyakba beépítve jelenik meg a közismereti informatika, van, ahol külön tárgyként oktatják az irodai alkalmazások használatát. Van, ahol a programozás, a robotika, a technika és a természettudományok együtt jelennek meg. A haladás irányát jól jellemzi az angol nemzeti alaptanterv, ahol napjainkban vezetik be minden képzési forma minden osztályában a programozást [6].

Idén másodszor vettem részt az Információs Társadalom Parlamentjén. Az oktatási területtel kapcsolatban – a kormány képviselője kivételével – egyöntetű volt az igény a programozás közoktatásban történő tanítására [7], a kormány részéről az alkalmazások más tantárgy keretében gyakorlása és az általános képzés előtérbe helyezése szerepel célként [8]. Eközben a hazai iskolákban, a maradék informatikaórákon, jellemzően az irodai alkalmazások oktatása folyik, a középszintű informatika érettségi részeinek megfelelően.

Nem csak a magyar informatikai cégek és felsőoktatás képviselői gondolják úgy, hogy mindenkinek képesnek kell lennie programozni. Az angolok új nemzeti alaptanterve mellett ezt bizonyítják a világ számos országában, köztük az EU-ban is meghirdetett CodeWeek [9] prog-

ramok, illetve a legjelentősebb európai informatikai szakmai szervezetek jelentése a közoktatásban végrehajtandó fejlesztésekről [10]

Miért gondolják olyan sokan, hogy a népnek tudnia kell programozni? Mert, aki tud programozni, az érti, hogy hogyan működik az az eszköz, amit használ, ezért jobban ki tudja használni a tulajdonságait. Mert, aki tud programozni, az tud tervezni, modellezni… problémát megoldani; az nagyobb valószínűséggel lesz a társadalom hasznos, produktív tagja.

Egy program megírása összetett feladat, számos ismeret, képesség és készség birtokában kell ahhoz lenni, hogy sikeresen vegyük az akadályokat. A programozás oktatásának különböző módszerei [11] ezen ismeretek és készségek tanításának sorrendjében, hangsúlyában különböznek:

”

- methodical, algorithmic oriented
- data oriented
- specification oriented
- problem type-oriented
- language oriented
- instruction oriented
- mathematics oriented
- hardware oriented
- model oriented.

”

- eljárás-, algoritmus-orientált
- adatorientált
- specifikációorientált
- probléma típus-orientált
- nyelvorientált
- utasításorientált
- matematikaorientált
- hardverorientált
- modell-, mintaorientált.

Az egyes módszerek sikeres alkalmazása az oktatás tárgyának (gyereknek, diáknak, hallgatónak) az előzetes ismereteitől, képességeitől függ. Számos tankönyv [12, 13, 14, 15], egyetemi tematika, jegyzet [16, 17, 18, 19] átnézése, 25 év tanítási gyakorlat alapján úgy látom, hogy a programozás tanítását nem lehet nulláról kezdeni. A különböző ismereteket, képességeket évekkel a tényleges programozás előtt el kell kezdeni fejleszteni ahhoz, hogy az egyes részképességek a programozás során összehangoltan érvényesüljenek, az egyes ismeretek értelmet nyerjenek a felhasználás során.

Én az X-generáció előtt születtem. Az oktatók jelentős része hozzám hasonló, vagy X-, illetve Y-generációba tartozik. Mi még felnőttként tanultuk az informatikát. De a Z-generáció kisgyermekként, az Alfa-generáció már születése pillanatában informatikai eszközökkel körülvéve él. Számukra természetes az eszközök, illetve a programok használata. Ezért azt gondolom, idejét múlt a programozás oktatásához a programok mellőzésével példát keresni. Egy mai kisgyerek előbb rajzol, mint teát főz. 15 évvel ezelőtt, még tanítani kellett a gyerekeknek a kattintást, menühasználatot, plug&play-t. Ma már ezt a mindennapi gyakorlatban „felszedik”. Ezért sokan úgy gondolják, hogy ennek a generációnak már nincs is szüksége informatikaoktatásra. Valójában azonban, egyre nagyobb szükség van egy újfajta informatikaoktatásra, ahol nem csak használni tanulják meg az eszközöket, de meg is értik, hogyan működnek; aminek során megtanulják kulturáltan, célszerűen használni – és kihasználni – az eszközök, szoftverek adta lehetőségeket.

Az ICT használatára lassacskán már egy majom is képes. Készíthetünk kutatást arról, hogy a majmok hány százaléka rendeli meg tableten a banánt. Az informatikai műveltség mértéke nem az ICT használatának ténye, hanem az eszközök működésének megértése, ismerete. Az informatikai kultúra – mire, hogyan használjuk az informatikai eszközöket – a műveltség függvénye, amelyet jellemez a viszonyunk az eszközhöz, a használat módja. A majom – bár meg tud érinteni három ikont – informatikai műveltsége, így kultúrája is közel nulla. Napjainkban tömegek

használják az informatikai eszközöket, azonban egyéni, gazdasági és társadalmi problémát jelent az informatikai kultúra alacsony szintje.

Az informatikai kultúra fejlesztésének szükségessége társadalmi kihívás. A kultúra alapja a műveltség, az eszközök működésének ismerete. Az eszközök működését programok határozzák meg, így a programozás tanítása szorosan összefügg a műveltség fejlesztésével. Ezért több országban tantervi célként szerepel, hogy mindenki tudjon valamilyen nyelven, alapszinten programozni, ezekben az országokban külön megjelenik a Computer Science vagy valami ehhez hasonló nevű/tartalmú tanóra.

A kérdés számomra az az út, ami magas szintű informatikai műveltséghez, informatikai kultúrához és ezzel összefüggésben a programozás megtanulásához vezet.

Évek során azt tapasztaltam, hogy van, aki „képtelen megtanulni programozni" és van, aki „magától, észrevétlenül megtanulja". Felmerült bennem a kérdés, hogy egyáltalán, tanítható-e a programozás. Számos cikk szól a programozás oktatásának módjáról. A különböző kísérleti módszerek egy-egy csoportban hatékonyabbak (valamivel), mint a kísérletet megelőző módszer. …abban a környezetben, a kísérletben részt vevő tanár-diák viszonylatban. Az eredményeket meghatározza – a módszer mellett – a diákok előképzettsége és a tanár személyisége [20, 21, 22]. Az elmúlt néhány évben olyan módszert, szemléletmódot választottam, amellyel – reményeim szerint –, az is megtanulja az alapokat, aki „képtelen megtanulni programozni". Hasonlóan Sysło „Computing thinking" elvéhez [23], a programozás oktatásának megkezdése előtt a gondolkodásmód fejlesztése áll a középpontban. A szemlélet lényege az indirekció: a tanár tudja, hogy a tananyag része a programozási ismereteknek, majd a programozás oktatása során erre fog építeni, de a diák számára ez csak a környezetének tanulmányozása.

Napjainkban az jellemző, hogy a programozás tanításával indirekt fejlesztjük az informatikai műveltséget, Ennek a fordítottját szeretném bemutatni, amikor az informatikai műveltség fejlesztésével készítjük elő a programozás elsajátítását. A lényeg, hogy milyen a viszonyunk az eszközzel, szoftverrel. Nézzünk bele, szedjük szét, derítsük ki mit, hogyan, miért… Jelen cikkben a „mindenki által ismert" Paint program, illetve grafikai alkalmazások segítségével közelítsünk a programozás felé!

Programozás oktatása – Paint-tel

Paint helyett tetszőleges egyrétegű pixelgrafikus program használható, amely képes bmp formátumban menteni. Valójában bármilyen pixelgrafikus szoftver megfelel, aminek egyik mentési formátuma tömörítetlen és könnyen érthető.

Nézzünk bele a képbe, szedjük szét, derítsük ki mit, hogyan, miért…

Alapok

Paintet, természetesen, minden legalább 12 éves tudja használni. Mégis, szánjunk több tanórát a mögöttes, rejtett fogalmak megismerésére. Kérdések mentén fel kell fedeztetni a diákokkal a kép, illetve a szoftver tulajdonságait, adattípusait, eljárásait és függvényeit.

Az 1. mellékletben megtalálható 1–25 kérdésekből álljon itt néhány **példa:**

2. Milyen objektumai vannak egy pixelgrafikus elven működő szoftvernek?
8. Hogyan adjuk meg egy pixel helyét?
13. Hányféle színt képes tárolni a szoftver? Milyen kapcsolatban van a választott színkód és a tárolt érték?
18. Milyen színű a #FABABA? (#FACEDE) Mit jelent színek megadásakor ez a 6 karakter? Miért jó a 16-os számrendszer? Mi a kapcsolat az így megadott és az eddigi színdefiniálási módszerek között?

20. Az egyik rendelkezésre álló tárhelyed szabad területe alapján, mekkora lehet az a 4:3 arányú, 24 bites színmélységű kép, ami még elfér a tárhelyen?

Még el sem kezdtünk rajzolni, de már van 25 kérdésünk egy pixelgrafikus program működésével kapcsolatban. A válaszokat részben megadhatja a művészet-, a matematika-, a természettudomány, de együttes értelmezésük, a képek készítésére vonatkozó általános szabályok megértése az informatikai műveltség része.

A kérdésekre válaszolva – a grafikai fogalmak (kép, pixel, méret, színmélység, formátum, szín, átszínezés, inch, cm, RGB, HSL, CMYK, színmélység) – mellett a programozás szempontjából fontos fogalmakkal is találkozik a diák:

- Létrehozás; törlés
- „objektum”; „tulajdonság”; „eljárás” (idézőjelben, mert definíció nélkül, hétköznapi értelmezéssel)
- táblázatos elrendezés; sor; oszlop; tömb-absztrakció; indexelés; indexelés bázisa (0: [0; 0], [n–1; m–1] 1: [1; 1], [n; m]); műveletek indexekkel;
- RGB; skála; kerekítés; digitális; kvantáltság; bináris kód; inch; bájt; bit; adattípus és ennek mérete; bináris, decimális hexadecimális számrendszer; adat (pozitív egész) ábrázolása;
- képméret; fájlméret, veszteséges és veszteségmentes tömörítés, optimális, képformátumok, szabvány, tömörítési eljárások (algoritmusok)

Elvárható tudás, az informatikai kultúrához tartozik, hogy egy felvett vagy rajzolt kép csak a közlés szándékának megfelelő részt tartalmazza, a közlés szándékának megfelelő színmélységgel és értelmezhető méretben. Például: Ne legyen teljes képernyős a rajz, ha csak egy 100×100 pixeles ábrát készítünk. A céldokumentumba (pl. weblap) ágyazva a megfelelő képformátumot válasszuk, a megjelenítendő részletet, méretet külön képként készítsük el. Veszteséges tömörítést utolsó lépésként alkalmazzunk. Akárhány nyomozós filmben szerepel, akkor se higgyük el, hogy egy kép felnagyításával eredetileg nem tárolt adathoz – és ezzel információhoz – jutunk. Magától értetődő legyen, hogy egy kicsinyítéssel információt veszítünk, amit nem tudunk nagyítással visszanyerni.

Az informatika, a számítástudomány, a programozás oldaláról közelítve a témát – bár egyetlen kérdésben sem mondjuk ki –, a kérdésekre adott válaszok a programozás során szükséges ismeretet, képességet tartalmaznak. A kérdések megalapozzák számos programozással kapcsolatos fogalom megértését. A kérdések indukálják, hogy a diák felfedezze a rajzoló szoftverben a programot, az adatokat, a kódot, az eljárásokat.

Talán erőltetettnek tűnik, hogy kérdések formájában fogalmaztam meg a tanítandó ismereteket, de ennek is oka van: Az is egy ismeret, hogy hogyan fedezzük fel egy program működését, hogyan szerezzünk információkat arról, amit eddig csak feketedobozként kezeltünk. A kérdésekkel lehet előkészíteni a programkészítés igényének felvetődését; támogatjuk a „hogyan működik” természetes kíváncsiság érvényesülését.

Ebben a fejezetben az alapokkal foglalkoztunk. A rajzolás alapjaival? Konkrétan, tanórákon a Paint használata közben, a kérdésekre válaszolva semmit sem rajzolunk. A diák egy ismert szoftverrel ismerkedik – erre szokták azt mondani, hogy minek –, de a tanár programozást tanít, a programozáshoz szükséges néhány alapfogalmat, a probléma felvetésének módját mutatja be.

Azért ne álljunk meg a szoftver statikus vizsgálatánál! Rajzoljunk is…

Algoritmusok

Kezdjünk el rajzolni! Lehet, a diákok reménykednek abban, hogy most aztán jöhet a játék, de most is rejtett programozásról van szó. A szoftver számos eljárást tartalmaz, amelyekkel illik megismerkedni. Ez egyben alkalmat ad arra, hogy tapasztalataink alapján algoritmusokat fogal-

mazzunk meg: Mit csinál a szoftver (a számítógépben dolgozó segédmunkás) az egyes pixelekkel, amikor…

- színmintát veszünk, illetve kattintunk egy képpontra?
- kitöltünk egy adott színnel?
- radírozunk?
- vonalat rajzolunk (nehezítés: a vonal vastagsága, az ecsetválasztás hogyan módosítja az eredményt; haladóknak a segédbillentyűk hogyan befolyásolják az eredményt)?
- téglalapot rajzolunk (nehezítés: szegéllyel, kitöltve, kitöltetlenül; matematikából haladóknak: kört, illetve más alakzatot rajzolunk, segédbillentyűket is használunk a rajzoláshoz)?
- kijelölünk egy tetszőleges részletet?
- másolunk, áthelyezünk (a háttérszín szerepét is tudatosan megadva)? Ekkor érdemes tisztázni a transzparens szín fogalmát is, egy kicsit visszautalva az alapokra.
- forgatunk, tükrözünk (haladó szinten: sor- és oszlopazonosítók transzformációjának megadásával)?
- átméretezünk?

A tanított korosztálytól, hozott ismeretektől, egyéni képességektől függ, hogy a fenti műveletekre milyen pontossággal adunk meg algoritmusokat. Az eszközök funkcióinak megismertetése során fejleszthető az algoritmikus gondolkodás. Kimondva, kimondatlanul felírunk – gyakran egymásba ágyazott – ciklusokat, az elágazást, értéklekérdezését és értékadást. Az így megadott algoritmusok az adott szoftver működését tapasztalati alapon írják le, ezzel az algoritmizálás készségét fejlesztjük. Az algoritmusaink hipotézisek.

A programozó által írt programkód algoritmusához úgy kerülhetünk közelebb, ha teszteljük az eszközt: meg kell figyelni, hogy különböző méretek, beállítások esetén mit tapasztalunk. Tesztelési esetek kidolgozását, állapotok megadását és állapotátmentetek meghatározását gyakoroltatjuk.

Egyrészt, rejtett célunk, hogy programozni tanítsunk, aminek része a programunk tesztelése. A tesztesetek megtervezése a helyes (körültekintő) megoldás alapvető feltétele. Másrészt informatikai műveletlenség egy programról, ami nem úgy működik, ahogy elsőre gondoljuk, azt mondani, hogy bugos, rossz… Minősíti a felhasználót, ha nem tudja, hogy egy adott szoftvertípus mire alkalmas, mik a korlátai, illetve, ha nincs tisztában azzal, hogy miért szükségesek az alapértelmezett beállítások, miért nem lehetnek ezek a beállítások éppen a legmegfelelőbbek. Azaz itt is a programozás rejtett oktatása egyben az informatikai műveltség fejlesztését is jelenti.

Miután pontosan ismerjük az eszközöket, talán el lehet kezdeni rajzolni? Esetleg egy rajzórán igen, informatikaórán azonban nem. A célszerű, pontos és gyors munka, tervezést igényel. A legegyszerűbb KRESZ-tábla megrajzolása is igényli a megvalósítás sorrendjének a tervezését. Egy összetett kép – például a tanterem alaprajza – elkészítése során már a feladat részekre bontása, a többször előforduló alapegységek megfelelő méretű prototípusának elkészítése majd másolása a munka idejét ötödére csökkenti. Az igazítások, fedések, árnyékok, tükröződések, szimmetriák, feliratok mind igénylik a megvalósítás tervezését olyannyira, hogy a matematikai szerkesztésekhez hasonlóan, egy feladat megoldásának értéke – informatikai szempontból – sokkal inkább az algoritmus minőségén múlik, mint a kézügyességen.

Eddig a programozáshoz szükséges ismeretekből előkerült az adattípus, adatstruktúra, az algoritmus, a tesztelés, az állapotok és állapotátmenetek fogalma. Most a programírásának egy újabb részfeladata: a tervezés került terítékre. Megfelelő feladatok választásával – tapasztalati szinten – készséggé válhat az ismétlődő elemek egyszeri megvalósítás többszörös felhasználása elve, a moduláris feladatmegoldás.

Érdemes arra is felhívni a figyelmet, hogy a művészi kivitelezés igényelheti, hogy pixelenként adjuk meg a színt, amit egész más típusú tervezés (vázlatkészítés) előz meg. Ez a terület a

művészeti képzéshez tartozik. Informatikai ismeretként nem értékelhető a művészileg szép rajz, nem a rajz szépségét értékeljük, hanem a kivitelezés technikáját.

Cél

Egy rajz vagy fénykép önmagában nem sokat ér. Annak megfelelően kell elkészíteni, hogy a dokumentumban, a prezentációban, a honlapon, a kiadványban vagy egy program kellékeként, hogyan szeretnénk felhasználni. Grafikai szempontból ez a cél (is) meghatározó a kivitelezés során, az informatikai műveltséget mutatja a célhoz vezető út megválasztása. Programozásoktatói szemüvegünkön keresztül viszont újra alapozunk. Eddig a kép és a képpont viszonyát vizsgáltuk, most a befogadó alkalmazás, (a gazdaobjektum) és a kép viszonyát elemezzük, értelmezzük. Egyes esetekben a rajzot csatoljuk, máskor beágyazzuk. Az 1. melléklet 26–31. kérdései ismét informatikai műveltséghez tartoznak, de alapjaiban a programozáshoz kapcsolódnak. Kód szintjén kell érteni, hogy a fájl kiterjesztésének átírásától nem lesz egy bmp-ből jpg. Érteni kell, hogy – csatolt kép esetén – egy hivatkozásban a kis és nagybetűk számíthatnak, az elérési útvonal is karakter, a karaktereket is bájtokban adjuk meg, ahogy a színeknek, a karaktereknek is van kódjuk…

Programozói szemmel a gazdaalkalmazáshoz hozzáadott kép pixeleit mint objektumokat, már csak közvetve érjük el (plugin fogalma). A gazdaalkalmazás objektumaihoz képest a képnek többféle viszonya lehet:

- Az objektumok sorozatának egyik tagja, amely a megadott sorrend szerinti helyen jelenik meg.
- Kódként az objektumok sorozatának egyik tagja, megjelenésében, az adott helyhez képest eltolva, a többi objektumhoz képest előtte, mögötte vagy az objektumok elrendezését módosítva jelenik meg.
- Egy objektum tulajdonságaként (például háttérkép) jelenik meg.

Informatikai műveltség kérdése, hogy nem keverjük a kitöltő színt a mögé helyezett képpel. Programozási ismeret, hogy az objektumnak lehet egy másik objektumtípus a tulajdonsága. Praktikusan az objektumok egymáshoz való viszonya határozza meg, hogy hogyan tudom módosítani a megjelenést.

A rajz felhasználási területei között nem véletlen a „program kelléke” említése. Ha már egy képet elhelyezünk más alkalmazásokban, akkor miért ne próbálnánk ki egy – grafikus felületet adó – fejlesztői környezetben is? Lehetőleg azt az eszközt választva, amelyben később programozást is szeretnénk tanítani, illesszük be az elkészített képet! Például: rajzoljunk sok virágot háttérként és külön 1-2 pillangót, majd állítsuk be egy programablak háttérképeként a virágos mezőt; néhány gomb háttérképeként pedig a pillangókat. Itt az alkalom, hogy egy pillantást vessünk a programozásra, megismerjük a fejlesztői felületet, miközben felhasználjuk a rajzunkat.

Játék

Amikor bejelentem, hogy rajzolást, Paintet fogunk tanulni, sokan úgy gondolják, hogy kikapcsolódás lesz a tanóra. Hamar rájönnek, hogy van még mit tanulniuk. A megoldandó (dolgozat) feladat egy, a fentihez hasonló virágos rét és néhány pillangó elkészítése. A cél lehet ezek montázsban egyesítése, dokumentumon belül egymásra helyezése, prezentációban animációval meg is lehet mozgatni… de lehet a feladat egy program hátterének és szereplőinek elkészítése, aminek a végén egy fejlesztői környezetbe helyezzük a rajzokat.

…és ekkor jöhet a játék, azaz a programozás. Aki akarja, továbbfejlesztheti a rajzait, de meg is mozdíthatja a pillangót. Kattintásra néhány pixeles elmozdulás… Néhány sor kód a táblán, ügyeskedve, egymásnak is segítve, pár perccel később már „repkednek” a pillangók: kattintásra, 3 pixelt felfelé. Majd jönnek az ötletek, kérdések: „Hogyan kell…?” Van, aki inkább szépítgeti a

rajzot… de a szomszédjának sikere felbátorítja, látja hogyan „kelnek életre” a pillangók, és ő is kipróbálja. Ha ezt elérjük, akkor már saját akaratból írt egy sor kódot az is, aki „képtelen megtanulni programozni”. A diákok egy része szeretne jobb hátteret, szebb pillangót. Kézzel rajzol, keres interneten… Azt is fel tudja használni… Talán észre sem veszi, de megváltozik az érdeklődésének fókusza: „Paintezni jöttem, de végül programoztam.”

Vektorgrafikus kiterjesztés

Adatok és algoritmusok

Célszerű a grafikai eszközök vizsgálatát pixelgrafikus programmal kezdeni, mert abban egyszerűbbek az objektumok. Azonban mindenképpen meg kell vizsgálni egy vektorgrafikus programot is. Külön figyelmet érdemes fordítani az „alakváltás” lehetőségére. Az általános sokszög, görbe, mosolygós fej… vizsgálata lehetőséget ad a csomópont és vonal objektumok jellemzésére. Egy vektorgrafikus szoftver megismerése során alaposan áttekinthetjük az „osztály”, a „öröklődés”, a „példány”, a „konstruktor”, a „függvény”, a „paraméter” fogalmakat, a pixelgrafikus szoftver vizsgálatához hasonlóan itt kérdések, problémafelvetések formájában:

- Melyek az alakzatok közös, illetve egyedi tulajdonságai?
- Egy alakzat (objektum) milyen objektumokat tartalmaz?
- Mit tárol egy vektorgrafikus képfájl?
- Milyen függvény írja le egy vonal megjelenését? (Milyen paraméterei, belső változói vannak? Vajon milyen matematikai összefüggéseket használ?)

Miközben a diák egy alkalmazás lehetőségeit vizsgálja, értelmezi, a tanár tudja, hogy aki közülük majd komolyabban foglalkozik programozással, ezen (vagy nagyon hasonló) példa alapján fogja megtanulni az objektumorientált-programozás elveit. Most csak a grafikai szoftver működését kell megértenie a diáknak, de ehhez már olyan fogalmakat társít, ami a programozáshoz szükséges.

Az eszköz működése (algoritmusai) is érdekesek. Hogyan gondolkodik, számol, működik a szoftver, amikor:

- beszúrok egy alakzatot?
- törlök egy alakzatot?
- az alakzatokat előre, hátra… viszem?
- egymáshoz képest középre igazítok?
- jobb (bal, alsó, felső) szélsőhöz igazítok?
- csoportba foglalok elemeket?
- alakzatot egy csoport mögé viszek?
- módosítom egy alakzat egy tulajdonságát?
- módosítom az alakzat csomópontjainak számát (beszúrok, törlök…)?
- módosítom az alakzat egy vonalának a tulajdonságát?

Egyes válaszok nagyon messzire elvezethetnek, de nem szükséges mindennek a végére járni, elegendő azt megmutatni, merrefelé kell kutatni. Ismét fontos megjegyezni, hogy az algoritmusok és függvények a működés alapján kitalált megoldási lehetőségek, nem a ténylegesen megírt kód leírása. . Így a „Vajon hogyan működik a…?” kérdésekre válaszolva kitalálhatjuk a csere, az átlagszámítás, a szélsőérték meghatározásának algoritmusait. A csoportba foglalással, a csomópontokkal, az élekkel kapcsolatban előkerülhet a lista, sőt, a hivatkozás és a pointer fogalma is.

Ahogy a pixelgrafikus szoftvereknél is érdemes ismerni a különböző mentési formátumokat (mert ez a kép adatainak kódolását jelenti) a vektorgrafikus képek mentési formátumainak specifikációit is érdemes tanulmányozni. Talán a legjellemzőbbek: EPS, WMF, SVG, amiből az SVG „meglepően” egyszerű.

Cél

A vektorgrafikus szoftverek használata sok esetben inkább a matematikai, mint a művészi készségeket fejleszti. A függvények, koordinátageometriai ismeretek felhasználása fontos, ezért a pixelgrafikus szoftverekhez képest idősebb korban (középiskola) érdemes ezen szoftverek működését vizsgálni. A vizsgálatot érdemes kiegészíteni a két szoftvertípus összehasonlításával.

Az informatikai műveltség része, hogy a felhasználó el tudja dönteni, egy kép vagy ábra elkészítéséhez melyik szoftvertípus a megfelelőbb, hogyan tudja praktikusan együtt használni a különböző grafikai rendszereket. A kép elkészítése előtt a tervezés még hangsúlyosabbá válik, a feladatmegoldás igényli a részfeladatok előzetes megtervezését, lényegében a megoldás algoritmusának elkészítését.

Játék

Az SVG specifikációja olyan egyszerű, hogy akár ki is próbálható: Rajzoljunk autót jegyzettömbben!

Ismét majdnem eljutottunk addig, hogy nyíltan kimondjuk: programot írunk. Itt egy példa arra, hogy kódot írunk és kép lesz belőle. Vizsgáljuk meg, mit tárol a számítógép (karakterek kódjait), illetve amit látok, az ezekből hogyan keletkezik.

- Mi a szoftver szerepe a kép megjelenítésében?
- A böngészőben is karakterekkel van kódolva, hogy mit kell megjelenítenie egy kódsor olvasásakor?
- Mi a program és mi az adat? Hogyan lesz a karakteres programból kép, futtatható program?
- Mit jelent, hogy egy programkódot egy interpreter értelmez, illetve egy compiler lefordít?

A „játék” során felmerülnek ezek a kérdések és ezzel tapasztalatot szerez a diák a kód mint a gép vezérlésének eszköze hasznosságáról. Eközben gyakorolja a több alkalmazást igénylő kódolás → mentés → futtatás → tesztelés → kódolás… programozási tevékenységet.

Művészi eszközök

A legegyszerűbb szoftverek működését elemeztük. Vajon hogyan működnek az „igazi nagyágyúk”? Miben más, miben hasonló a GIMP, PhotoShop, Google SketchUp… a kedvenc grafikai szoftverem?

Természetesen a válasz nagyon sokféle lehet. Mind a programozás bevezetéséhez, mind az informatikai műveltség fejlesztéséhez, nem a válasz a fontos, hanem a kérdés felvetése és képesség arra, hogy felfedezzük az azonos objektumokat, tulajdonságokat, eljárásokat. Felfedezzük az egyszerű megoldás kiterjeszthetőségét, a modulok újrahasznosíthatóságában rejlő lehetőséget, az eszközöket megvalósító matematikát, algoritmusokat.

Az oktatás és a számonkérés módszerei

A bevezetés a programozásba kurzusok jelentős része gyakorlatokkal kísért előadássorozat. Mivel az itt vázolt témafeldolgozásban – néhány alkalmazói szoftveren keresztül – rejtve tanítjuk a programozást, előadásra csak egy-egy komolyabb elméletet igénylő résznél van szükség. A kérdések jelentős részére a diákoknak kell megkeresni a válaszokat. Fontos a felfedezés, a rácsodálkozás élménye. Ahhoz, hogy egyszer képes legyen a diák programot írni, emberközelivé kell tenni a programokat. Bár azt mindenki tudja, hogy a szoftvert nem Isten teremtette, de sokan úgy viszonyulnak hozzá, mintha a programozók csodalények lennének. A szoftver értelmes felhasználásához – informatikai kultúrához – és az első programok megírásához is az szükséges, hogy megértsük a programozó gondolkodásmódját. Mint egy kisgyerek, mindenre rákérdezünk: Mi ez? Miért így?

Értékelésnél a programozási ismeretek meglétét ne kérjük számon. Ezt a területet hagyjuk meg a stresszmentes felfedezéseknek. A rajzolási feladat pontos, helyes és gyors megoldását

értékeljük, azaz alapvetően az informatikai műveltséget, kultúrát minősítjük. Ez csak közvetve jelenti a programozási ismeretek értékelését. Másik oldalról, ne értékeljük (túl) a kép szépségét, csak a megvalósítás technikai helyességét (pl: rikító színek használata, rózsaszín előtérbe helyezése… ne legyen értékelési szempont), mert ez a művészeti kompetencia.

Tovább a programozásra

Menjünk tovább a kutatásban, próbáljunk meg „meghekkelni" egy képet! A pixelgrafikus képek mentési formátumai közül a tömörítetlen BMP tűnik a legegyszerűbbnek, vektorgrafikus formátumok közül az SVG… Tetszés szerint választható más is… Az SVG szöveges, így a képbe belepiszkálás nem okoz gondot, de a bmp kódjait láthatóvá kell tenni.

A feladat megoldásához célszerű egy alapprogramot adni a diákoknak, ami beolvas egy képet, lehetővé teszi a módosítást és új néven el is menti. A legjobb, ha a keretprogramhoz azt a programozási nyelvet használjuk, amit később is szeretnénk tanítani. A keretprogram lényegében elérhetővé teszi az egyes pixelek színkomponenseit, adatstruktúra szempontjából a pixels[i,j].Red kifejezést kell értenie a diákoknak. Néhány lehetőség bevezető feladatként:

1. Készítsük el a beolvasott kép átszínezett változatát! Azaz a kép betöltése után minden (R; G; B) színű képpontot helyettesítsünk (R; 0; 0)-val
2. Kódoljunk a képbe egy fekete téglalapot!
3. Tükrözzük a képet vízszintesen, függőlegesen, forgassuk el 180 fokban!
4. Négyzet alakú képet, vagy a kép egy négyzet alakú részét forgassuk el jobbra, illetve balra 90°-kal!

Az 1. feladat egyszerű értékmódosítás, látványos eredményekkel. Élmény… Programozás szempontjából mindjárt két egymásba ágyazott ciklust kell alkalmazni, de ez a keretprogram része is lehet, elegendő felhasználni. A feladatban lehet ismerkedni a fejlesztőkörnyezettel, a programfuttatás módjával, érdemes kipróbálni a konzolra kiírást, megtapasztalni, hogy az egyes pixelek egy-egy színösszetevője tényleg az a szám, amit a Paint is mutat. Emellett elkezdhetünk alaposabban körülnézni, mit is csinál a programunk. Hol, hogyan van megírva a kép, illetve a pixel? Mit jelent az osztály, objektum az adott esetben, azaz hogyan kódoljuk, amit a Paintben láttunk. Amit a bitmapről olvashattunk, az hol jelenik meg a kódban. A kép beolvasásában, kiírásában hol a fájlnév, ez hogyan módosítható?

A 2. feladat a tömb indexelésének tanulmányozását indukálja, ami szükséges a kép–adat absztrakciójának a megértéséhez. A feladatban a ciklusokat tudatosabban kell paraméterezni. Kiindulásként egyetlen képpont (pl. bal felső sarok) átszínezése ad megfelelő tapasztalatot, majd a ciklusszervezés lényegét és a paramétereinek pontos szerepét kell tisztázni.

A 3. és 4. feladat koordináta-geometriai ismeretet követel (vagy nagyon jó matematikai készségeket). A feladatok megoldása segít megérteni az ciklusváltozó és indexek kapcsolatát.

Ezen a ponton tudatosítani kell a diákokban, hogy a programozás elsajátításához egy nyelvet kell megtanulniuk, nem csak olvasni, hanem helyesen írni is. Meg kell tanulni a nyelv szintaktikáját és a „helyesírás-ellenőrző" használatát.

Az oktatás és a számonkérés módszerei

A fejezet címe: Tovább a programozásra. Ez még nem a programozó szakmai képzésről szól, hanem tevékenységi formáról (programírás), illetve gondolkodási készségről. Nem célja a nyelv részleteinek strukturált megismerése, sem a matematikai megalapozás. Inkább kedvcsináló, amely megmutatja az utat egy tudomány megismerésére felé.

A programozás aktív cselekvés. Önálló kreatív munka, gondolkodás. Ennek élményét kell megtapasztalnia a diáknak. Nem a program minősége, szépsége a lényeg. A megoldás akkor teljes értékű, ha teljesen önállóan sikerült megvalósítani egy elképzelést.

Néhány feladat megoldása újabb igényeket szül… tetszés szerint, illetve a képzési terv alapján innen tovább lehet menni:

- a programozási tételek vizsgálata felé (egy adott szín keresése, legsötétebb szín keresése, kép kicsinyítése, torzítása, kitöltés - rekurzió…),
- matematikai alkalmazások felé (titkosítás, tömörítés),
- szövegfeldolgozás irányába (színcsere, vörösszem keresése, „kép színének” meghatározása),
- objektumorientált programozás felé, illetve az adott programozási nyelv alapos megismerése felé.
- …

De ez már egy másik történet, a programozás felfedezése után, a programozás alapjainak strukturált megismerése.

Összefoglalás

A lányom mondta alig 2 éves ikreinek:
"Ott egy locsolócső, olyan, mint a kígyó."
Majd rájött, hogy ezzel semmit sem mondott.

Szerencsés esetben, az itt leírt gondolatok nem egy tanfolyam anyagát jelentik, hanem a 6–8 éves közoktatási folyamat egy vetületét. 9–10 éves gyerekek már képesek arra, hogy választ keressenek az első kérdésekre. Évek során – a dokumentumkészítést, táblázatkezelést, adatbáziskezelést, eszközök vezérlését is hasonlóan tanulva – megértik, megtanulják a fogalmakat, megértik a „játékszabályokat”, amiket a programozók a szoftverekben alkalmaztak. 15–16 éves korukra eljutnak arra a szintre, hogy megpróbálják reprodukálni a szoftver egyes elemeit. Ekkor lesz a feltételezésből megbizonyosodás. A tapasztaltak és olvasottak alapján elképzelt modellt ekkor lehet kipróbálni.

Aki kívülről érkezik a digitális világba, annak valószínűleg szüksége van a teafőzés algoritmusára, a „hello world”-re, az összeadó programra ahhoz, hogy megértse, mi az algoritmus, a kód és a változó. Akik ebben nőttek fel, azok – egy kis iránymutatással – annyi példát látnak, hogy észrevétlenül megtanulják ezeket a fogalmakat, ellenben nehezen értik meg, mi köze van a teakészítésnek a maximális érték meghatározásához. A tanár feladata, hogy tisztában legyen azzal, hogy nem az eszköz használatát kell megtanítania, hanem a működési elvet, azaz a programozó gondolkodásmódját, a program tulajdonságait kell megismertetni. A diák pillanatnyi tudásához, szókincséhez (valójában fogalomrendszeréhez) apránként hozzá kell illeszteni azokat az ismereteket, amik a programozáshoz szükségesek. Természetesen, nem a grafika az egyetlen alkalmazástípus, ami ezzel a szemlélettel tanítható, illetve amin keresztül a programozás tanítható. Az informatikai kultúra magas színvonalát az adja, ha minden alkalmazást programozói szemlélettel vizsgálunk, a programozói szemléletet pedig a programokból a programozók által készített alkalmazásokból ismerhetjük meg.

Azt tapasztaltam, hogy a Paint alapjai mindenki számára elsajátítható, érthető. Az algoritmusok leírása sem okoz gondot, a formalizálásra kell jobban odafigyelni. A „játék”-ot – a rajzolás témakörbe csempészett programozást – csak az utóbbi években vetettem be, de akkor mindenki megírta a kódot, többen továbbfejlesztették, sőt, volt olyan lány, aki másnapra a programot lufikkal egészítette ki… azóta programozónak készül.

Mivel középiskolában tanítok, a pixelgrafika tárgyalásával egy időben kell a vektorgrafika alapjait is tanítanom. Az ideális az lenne, ha 14 évesen a pixelgrafikával kapcsolatos fogalmak már ismertek lennének, amit a vektorgrafika tárgyalása során csak ismételni kell. Jellemzően

nem jut idő a képformátumokra. Csak speciális osztályban tudtam beiktatni a komolyabb grafikai programokat.

Programozásra általában nincs idő a tanórákon. A programozásra rávezető feladatokat szakkörön, programozó-táborokban szoktam kiadni. Mindig nagy sikere volt. Volt, aki néhány óra alatt minden részfeladatot megoldott, szteganográfiára is adott egy megoldást, majd komolyabb kihívás után nézett a versenyfeladatok között, ahol például a tömörítéssel kapcsolatos feladat is szerepelt. A másik véglet, aki elmerült a feladat „szépségében"; órákon át saját átszínezéseket talált ki. Ez is nagyon hasznos, mert közben felfedezte a programozásban a kreativitást.

Diákjaim szorgalmasak, tehetségesek… Nem tudom, más környezetbe mennyire adaptálható a módszer, a tematika. Remélem, megtetszik néhány kollégának és kipróbálják. 20 éve tanítok informatikát, én sem így kezdtem. Azt gondolom, hogy a felnövekvő Y-, Z- és Alfa-generációnak természetesebb a használt szoftverek vizsgálata során megérteni a szoftverek mögött levő programozót, majd megtanulni a programozás alapjait.

Azt gondolom, hogy ma már bármelyik évfolyamon a programozás tanítását be lehet ágyazni diákok által ismert szoftver használatának oktatásába. Informatikaórákon együtt, egymással összhangban kell az informatikai műveltséget és a programozást oktatni.

Itt is szeretném megköszönni jelenlegi és volt diákjaimnak az ötleteket és a visszajelzéseket. Ők, a „maguktól tanuló, felfedező" diákok és persze a „másmilyen" diákok azok, akik arra ösztönöznek, hogy mindig újítsak.

Hivatkozások

1. Tamási Áron Liceum, Székelyudvarhely, „Informatika,” [Online]. Available: http://www.gimi.ro/informatika. [Hozzáférés dátuma: 29 október 2014].
2. Tamási Áron Líceum, Székelyudvarhely, „Tantervek,” [Online]. Available: http://www.gimi.ro/tantervek. [Hozzáférés dátuma: 29. október 2014].
3. Sellye János Gimnázium, Komárom, „Informatika,” [Online]. Available: http://www.sjg.stranka.info/index.php?vid=GAH. [Hozzáférés dátuma: 29. október 2014].
4. SM Sain Seri Puteri, Kuala Lumpur, „Subjek Ditawarkan,” [Online]. Available: http://seseri.edu.my/v2/?page_id=180. [Hozzáférés dátuma: 29. október 2014].
5. St. Mark's Sr. Secondary Public School, Meera Bagh, New Delhi, „Syllabus,” [Online]. Available: http://www.saintmarksschool.com/meerabagh/academics/syllabus.html. [Hozzáférés dátuma: 29. október 2014].
6. Department for Education, GOV.UK, „National curriculum,” 16. július 2014.. [Online]. Available: https://www.gov.uk/government/collections/national-curriculum. [Hozzáférés dátuma: 1. novewmber 2014.].
7. Információs Tásadalom Parlamentje, „IT Parlament 2014 - kerekasztal-beszélgetés a köznevelés fejlesztéséről,” 27 június 2014. [Online]. Available: http://infoter.eu/video/it_parlament_2014_kerekasztal-beszelgetes_a_kozneveles_fejleszteserol. [Hozzáférés dátuma: 29. október 2014].
8. I. Sipos, „http://infoparlament.infoter.eu/_down/2-01.sipos_imre.pdf,” 27 június 2014. [Online]. Available: http://infoparlament.infoter.eu/_down/2-01.sipos_imre.pdf. [Hozzáférés dátuma: 29. október 2014].
9. Young Advisors Group at the European Commission, „Code Week,” 17 október 2014. [Online]. Available: http://codeweek.eu/. [Hozzáférés dátuma: 29. október 2014].
10. Informatics Europe & ACM Europe Working Group on Informatics Education, „Informatics education: Europe cannot afford to miss the boat,” április 2013.. [Online]. Available: http://europe.acm.org/iereport/ACMandIEreport.pdf. [Hozzáférés dátuma: 1. november 2014.].
11. P. Szlávi és L. Zsakó, „Methods of teaching programming 1(2):,” Teaching mathematics and Computer Science, %1. kötet 01, pp. 247-258, 2003.
12. Z. Illés, Programozás C# nyelven, Budapest: Jedlik Oktatási Stúdió, 2005.
13. K. Jónás, C# nyelv és a programozás alapjai, Budapest: Jedlik Oktatási Stúdió, 2014.
14. I. Reiter, C# programozás lépésről lépésre, Budapest: Jedlik Oktatási Stúdió, 2012.
15. Á. Dusza, Algoritmusok Pascal nyelven, 2 szerk., Miskolc: Dusza Bt, 2006.
16. Z. Dr. Czirkos, „INFOC - a BME EET Programozás alapjai I. tantárgyának tanulmányi portálja,” BME EET, 2009-2014. [Online]. Available: https://infoc.eet.bme.hu/. [Hozzáférés dátuma: 29. október 2014].
17. ELTE IK, „Programozási alapismeretek,” [Online]. Available: http://progalap.elte.hu. [Hozzáférés dátuma: 29. október 2014].
18. D. J. Malan, „EDX HarvardX: CS50x Introduction to Computer Science,” Harvard, [Online]. Available: HarvardX: CS50x Introduction to Computer Science. [Hozzáférés dátuma: 29. október 2014.].
19. IIT Bombay, „EDX - IITBombayX: CS101.1x Introduction to Computer Programming, Part 1,” IIT Bombay, [Online]. Available: https://courses.edx.org/courses/IITBombayX/CS101.1x/2T2014/info. [Hozzáférés dátuma: 29. október 2014.].

20. F. A. Dorça, L. V. Lima, M. A. Fernandez és C. R. Lopes, „A Stochastic Approach for Automatic and Dynamic Modeling of Students' Learning Styles in Adaptive Educational Systems," Informatics in Education, %1. kötet11, %1. szám2, pp. 191-212, 2011.
21. Y. Hofoku, S. Cho, T. Nishida és S. Kanemune, „Why is programming difficult? -Proposal for learning programming in "small steps" and a prototype tool for detecting "gaps"," in Universitätsverlag, Potsdam, 2013.
22. U. Nikula, O. Gotel és J. Kasurinen, „A Motivation Guided Holistic Rehabilitation of the First Programming Course" ACM Transactions on Computing Education. kötet 11,. szám4, 2011.
23. M. M. Sysło, Informatics for all students - A Computational Thinking Approach, 2014..

1. Melléklet

1. A rajzoló eszköz rajzolási elve lehet pixelgrafikus, illetve vektorgrafikus. Mit jelentenek ezek a fogalmak? Miben hasonló, miben tér el a szoftverhasználat?
2. Milyen objektumai vannak egy pixelgrafikus elven működő szoftvernek? (kép, pixel)
3. A képnek milyen tulajdonságai, eljárásai vannak? (méret, színmélység, formátum…)
4. A pixelnek milyen tulajdonságai, eljárásai vannak? (szín, átszínezés, létrehozás, törlés)
5. Mi az, hogy „objektum”, „tulajdonság”, „eljárás”?
6. Milyen mértékegységgel adható meg a kép mérete? Mekkorák ezek a mértékegységek? Mi közöttük a váltószám? (pixel, inch, cm)
7. Mi a kapcsolata a képnek és a pixeleknek? (táblázatos elrendezés, tömb absztrakció)
8. Hogyan adjuk meg egy pixel helyét? (sor, oszlop; koordináta-rendszerrel hasonlóság, eltérés)
9. Adjuk meg egy adott kép „legelső” illetve „legutolsó” pixelének sor- és oszlopazonosítóját! (pl.: [0, 0], [n–1, m–1])
10. Számítsuk ki, hol van a kép közepe! ... (Koordináta-geometriai előismeretektől függően még számos feladat adható.)
11. Hogyan adhatjuk meg egy pixel színét? A különböző módszereknek mi az elve? (RGB, HSL, CMYK)
12. Hányféle színt ajánl fel a szoftver a HSL és RGB… modellek alapján? Hogyan lehet, hogy a modelltől függ az érték? (skála, kerekítés, digitális, kvantált)
13. Hányféle színt képes tárolni a szoftver? Milyen kapcsolatban van a választott színkód és a tárolt érték?
14. Mi a színmélység?
15. Miért pont 0–255 egy RGB színkomponens értéke? (Miért éppen 256 féle értéket vehet fel egy komponens?)
16. Mi a Bájt? Mi a bit?
17. Mentéskor választhatsz fekete-fehér, 16 színű, 256 színű és 24 bites formátumot. Az egyes formátumok mennyi adatot tárolnak az általad megadott színekből? Mekkora tárhely kell egy pixel adatainak mentéséhez?
18. Milyen színű a #FABABA? (#FACEDE) Mit jelent színek megadásakor ez a 6 karakter? Miért jó a 16-os számrendszer? Mi a kapcsolat az így megadott és az eddigi színdefiniálási módszerek között?
19. Hogyan bővül a 24 bites színskála, ha 32 bites a tárolási méret?
20. Az egyik rendelkezésre álló tárhelyed szabad területe alapján, mekkora lehet az a 4:3 arányú, 24 bites színmélységű kép, ami még elfér a tárhelyen?
21. Mit jelent, hogy egy fényképezőgép x megapixeles? A te fényképező eszközöd hány MP-s?
22. Körülbelül hányszorosára nő a képméret, ha 32×32 pixel helyett véletlenül 32×32 inches képméretet állítasz be?
23. A kép mentésekor nem csak bittérkép formátum lehetséges. Miért van többféle formátum? Miben térnek el a formátumok? Keress az egyes formátumokról ismertetőt az interneten!
24. Mit jelent a veszteséges tömörítés? Mikor veszteségmentes a tárolás?
25. Milyen elvek betartása javasolt a képek méretére, formátumára és színmélységére egy kép készítése, módosítása illetve felhasználása során?
26. Mi a különbség a csatolás és a beágyazás között?
27. Hogyan jelenik meg kódban csatolás és beágyazás közötti különbség?
28. Hogyan fog változni a gazdafájl mérete, mire kell figyelni a másoláskor?
29. Mi köze van a kép formátumának a beágyazhatósághoz, a csatoláshoz?
30. Az egyes képformátumok hol, hogyan használhatók jól?

31. Hol jelenik meg a kép, hogyan tudjuk a megjelenést módosítani?