



**BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM**  
**Gazdaság és Társadalomtudományi Kar**  
Műszaki Pedagógia Tanszék

# **A gyakorlati programozás és játékfejlesztés tanításának alternatív módszere grafikai felületek támogatásával**

Készítette: Nyirő Péter  
Mechatronikai mérnök  
BSc szakos hallgató

Konzulens:  
Dr. Molnár György  
egyetemi docens

**Tudományos Diákköri Konferencia**  
**Budapest, 2013**

# Tartalomjegyzék

|           |                                                                  |           |
|-----------|------------------------------------------------------------------|-----------|
| <b>1.</b> | <b>BEVEZETÉS .....</b>                                           | <b>1</b>  |
| <b>2.</b> | <b>A PROGRAMOZÁS TANÍTÁSÁNAK MÓDSZERTANA .....</b>               | <b>3</b>  |
| 2.1       | A HAGYOMÁNYOS ELMÉLETORIENTÁLT OKTATÁS .....                     | 6         |
| 2.2       | A GYAKORLATON ALAPULÓ OKTATÁS.....                               | 7         |
| <b>3.</b> | <b>A FEJLESZTŐI ESZKÖZ .....</b>                                 | <b>10</b> |
| 3.1       | GRAFIKUS FELÜLET ÉS A GML.....                                   | 11        |
| 3.2       | MULTIMÉDIA.....                                                  | 14        |
| <b>4.</b> | <b>AZ OKTATÁSI MÓDSZER ALKALMAZÁSA.....</b>                      | <b>17</b> |
| 4.1       | A KURZUSOK CÉLCSOPORTJA .....                                    | 17        |
| 4.2       | A KURZUSOK TANANYAGA.....                                        | 19        |
| 4.2.1     | <i>Kezdő kurzus.....</i>                                         | <i>19</i> |
| 4.2.2     | <i>Haladó kurzus.....</i>                                        | <i>21</i> |
| <b>5.</b> | <b>A PRÓBAKURZUSOK ELÉGEDETTSÉGVIZSGÁLATÁNAK BEMUTATÁSA.....</b> | <b>24</b> |
| 5.1       | AZ ELŐADÁSOK TAPASZTALATAI .....                                 | 25        |
| 5.2       | AZ ELŐADÁSOK ALAPJÁN KÉSZÜLT JEGYZETEK .....                     | 26        |
| 5.3       | AZ ELŐADÓ .....                                                  | 27        |
| 5.4       | SZÁMONKÉRÉS .....                                                | 28        |
| 5.5       | MOTIVÁCIÓ, JELENLÉT .....                                        | 31        |
| 5.6       | ÖSSZEFOGLALÓ ÉRTÉKELÉS .....                                     | 33        |
| <b>6.</b> | <b>TOVÁBBI MUNKÁK.....</b>                                       | <b>36</b> |
| 6.1       | ÚJABB FELMÉRÉS INDÍTÁSA .....                                    | 36        |
| 6.2       | TÁVOLABBI CÉLOK.....                                             | 37        |
| <b>7.</b> | <b>ÖSSZEFOGLALÁS .....</b>                                       | <b>38</b> |
|           | <b>IRODALOMJEGYZÉK .....</b>                                     | <b>40</b> |
|           | <b>MELLÉKLETEK, FÜGGELÉK.....</b>                                | <b>43</b> |

## Táblázat- és ábrajegyzék

|                                                                   |    |
|-------------------------------------------------------------------|----|
| 1. ábra: Játékprogramozás Game Maker 8.0 használatával .....      | 2  |
| 2. ábra: PHP programozási nyelv kódrészlete .....                 | 3  |
| 3. ábra: LabVIEW programozási nyelv részlete .....                | 4  |
| 4. ábra: Code Hero, programozást oktató játék .....               | 5  |
| 5. ábra: Game Maker 8.0 logója .....                              | 10 |
| 6. ábra: Game Maker grafikus felülete .....                       | 12 |
| 7. ábra: GML-ben írt kódrészlet .....                             | 14 |
| 8. ábra: Animáció készítés a Game Maker felületén belül .....     | 16 |
| 9. ábra: Ipari robotokat vezérlő program GML-ben elkészítve ..... | 18 |
| 10. ábra: Játékkészítés a kezdő kurzus alapján .....              | 21 |
| 11. ábra: Kész játék a haladó kurzus alapján .....                | 23 |
| 12. ábra: Előadáson bemutatott példafájl.....                     | 25 |
| 13. ábra: Egy elkészített házi feladat .....                      | 30 |
| 14. ábra: Egy jelentkező által készített játék.....               | 33 |
| 15. ábra: Dolgozat eredmények melléklete.....                     | 34 |
| 16. ábra: Dolgozat utáni kérdőív részlete .....                   | 35 |
| 17. ábra: Egy GML-ben írt játék.....                              | 38 |

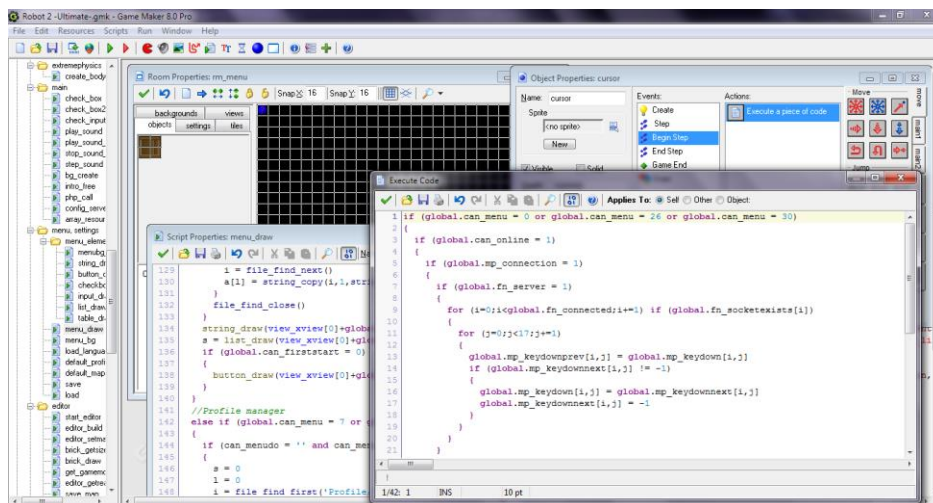
## 1. Bevezetés

A számítástechnika nagymértékű fejlődésének köszönhető, hogy társadalmunk információ központúvá vált. (Siemens, 2005) (Siemens, 2006) Ennek hatására, ahogy sok más tudomány, a pedagógia is átalakulóban van. (Benedek, 2007) (Miskolczi, 2010) Főleg a felsőoktatásban jelenik meg a probléma, miszerint a diákoknak egyre nagyobb információ mennyiséget kell elsajátítaniuk rövid idő alatt. Egyre több kutatás mutatja ki, hogy az ember gondolkodásának elsősorban vizuális alapja van, ezt használva lényegesen nagyobb hatékonysággal tud memorizálni és tanulni elméleti dolgokat. (Bleed, 2005) (Nyíri, 2003) (Visual Education, 2013) Ugyan sok helyen megjelenik már a vizuális oktatás gyakorlati alkalmazása, de jellemzően még ma is a szöveg és könyv alapú oktatás az elfogadott a pedagógiában. (Minden, ami Közgáz, 2010)

Az informatika és ezen belül a gyakorlati programozás oktatásának esetében ennek az ellentétnek egy szélsőséges változata lelhető fel. A számítógépek nagymértékű fejlődésének köszönhetően ma már nem okoz problémát képek, hangok, animációk és videók megjelenítése bármilyen elektronikai berendezésen. Ezzel szemben a programozás oktatása sok helyen még mindig a szöveges formát támogatja. Azért vált jelentős méretűvé ez a probléma napjainkra, mert az információ központúság miatt elengedhetetlen az élet bármely területén az informatikai tudás és sok helyen a programozás alkalmazása. Éppen ezért szükséges jelentős figyelmet fordítani a programozás oktatására és a lehetőségekhez mérten hatékonyabb módszert alkalmazni a jelenleginél.

A dolgozat egy alternatív módszert mutat be a programozás oktatására, amely teljes mértékben kihasználja a számítógépek adta vizuális lehetőségeket. Az elméletközpontú oktatás helyett a gyakorlatra és a megszerzett tudás alkalmazhatóságára koncentrál elsősorban. (Katona, Kurdi, 2008) Ennek egyik feltétele, hogy a diákok már az első órán megismerjék a programozásban alkalmazott, logikán alapuló gondolkodásmódot. A szükséges, nagy mennyiségű elméleti tudásra csak a programozás alkalmazásának valamilyen szinten történő begyakorlása után tér ki a módszer. A dolgozat bemutat egy programot, amely segítségével ez az oktatási módszer könnyen megvalósítható.

A képek és a hangok a számítástechnikában betöltött jelentős szerepe miatt a diákok érdeklődését szintén egyre nehezebb fenntartani a hagyományos pedagógiai módszerekkel. A motiváció növelésének érdekében az itt kidolgozott oktatási módszer a számítógépes játékfejlesztésre koncentrál elsősorban. (Haden, 2006) (Kirkland, O’Riordan, 2008) Ennek további előnye, hogy a programozói képzéshez szorosan kapcsolódó multimédiás és általános számítógépes ismeretek mind megjelennek a játékfejlesztés során. Komplexitás szempontjából lényegesen több figyelmet kell fordítani a játékfejlesztésre, mint funkcionális programok tervezésére. (Chen, Cheng, 2007) (Rajaravivarma, 2005) Egy mai számítógépes játék tervezése és programozása során nagyon sok részletre oda kell figyelni, ezek begyakorlása után a tanuló egy sokkal átfogóbb és biztosabb tudás birtokában folytathatja programozói tevékenységét.



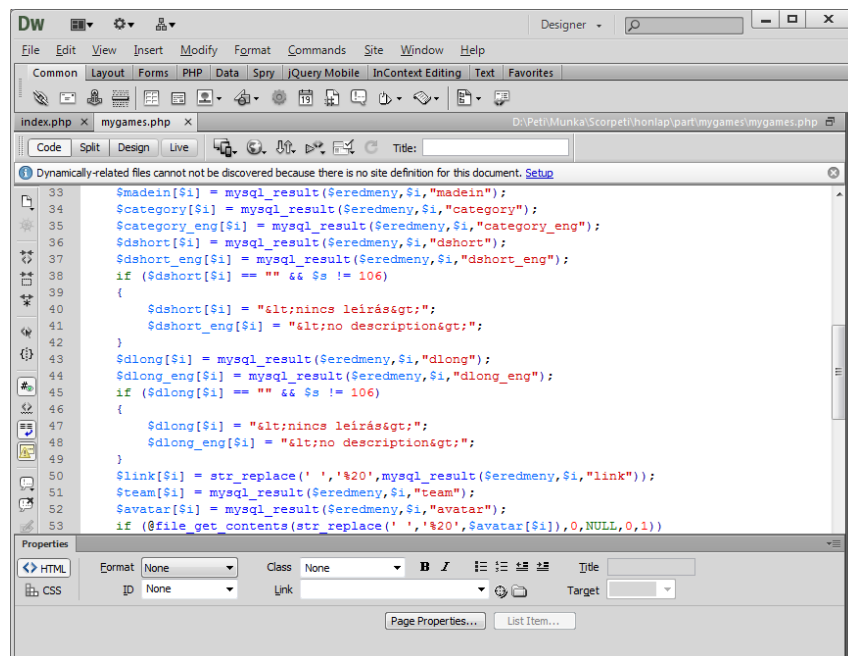
**1. ábra: Játékprogramozás Game Maker 8.0 használatával**

*Forrás: saját készítés*

A munka célja, hogy ezt az elméleti módszert a gyakorlatba is át lehessen ültetni. Készült egy felmérés, amely keretében néhány hallgató egy rövid kurzuson vett részt. Ezen a kurzuson a hallgatók megtapasztalhatták a kidolgozott módszer előnyeit és hátrányait, majd véleményt nyilváníthattak kérdőívek formájában. A dolgozat kitér a visszajelzések alapján készült statisztikára is, majd levonja a következtetéseket a további teendőkkel kapcsolatban. (Moore, 1997) (Terrón, 2003)

## 2. A programozás tanításának módszertana

Az informatika és a számítógépek használata sok esetben szorosan kapcsolódik a programozói ismeretekhez. Ugyan a legtöbb program fejlesztésekor annak széles körű használhatósága az egyik legfőbb szempont, sokszor elkerülhetetlen, hogy a programozás során használt szakszavakkal, esetleg valamilyen programozói tudást igénylő feladattal találkozzon a felhasználó. Ezért szükséges lehet egy általános programozói ismeret megszerzése azok számára, akik később – akár teljesen más szakterületen – munkájukhoz számítógépet használnak.

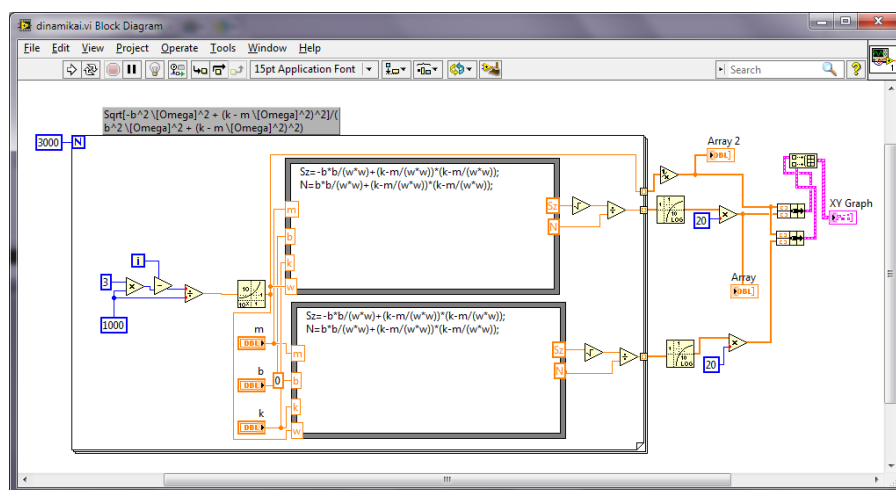


2. ábra: PHP programozási nyelv kódrészlete

*Forrás: saját készítés*

A mai népszerű programnyelvek (például: Java, C, C++, C#, Objective-C, PHP, Python, Ruby) mindegyike szöveges formát használ. (Mikoluk, 2013) Ennek előnye, hogy gyorsabban lehet velük dolgozni, mint a grafikus felületeket használó programnyelvekkel. Hátránya viszont, hogy egy programnyelv összes helyesírási formáját és elméleti felépítését meg kell tanulni, mielőtt azt széles körökben alkalmazni lehetne. Egy kezdő programozó számára nagy nehézséget okoz a szöveges programozás tanulása, mert az összes elméleti szabályt figyelembe kell vennie. Emellett a program felépítésének tervezésére és a gyakorlati módszerek alkalmazására már sokkal kevesebb figyelmet képes fordítani. Ezt a komplex feladatot csak azok tudják elvégezni, akiknek

egyáltalán nem okoz problémát a programozásban alkalmazott logikai matematika alkalmazása. Ugyanakkor egy ilyen programnyelv elsajátításához szükség van magas fokú elszántságra is, mivel a sok elmélet és a gyakorlat tanulásakor fellépő kezdeti nehézségek miatt csak sokára lesz sikerélménye a tanulónak.



**3. ábra: LabVIEW programozási nyelv részlete**

*Forrás: saját készítés*

A grafikus programnyelvek (például: App Inventor, Scratch, LabVIEW) a szövegesekkel ellentétben egy kezdő programozó számára sokkal átláthatóbbak és könnyebben tanulhatóak lehetnek. Ennek az oka, hogy amíg egy gyakorlott programozó a programsorok megírásakor sokszor fejben tartja a kódok kapcsolódását, a kezdőknek szükségük van a vizuális megjelenítésre. A grafikus nyelvben írt program logikája ránézésre könnyebben átlátható, továbbá szintaktikája is jóval egyszerűbb. Nem kell ismerni az írásjelek jelentését és helyes sorrendjét, csak a képek használatát és működésének módját kell megtanulni. Tehát egy grafikus programnyelv használatához kevesebb elméleti háttérre van szükség, programtervezés során kevesebb figyelmet kell fordítani a helyesírásra, könnyebb koncentrálni a program végső felépítésére. A tapasztalt programozók mégis főleg szöveges nyelveket használnak. Ennek az oka, hogy a grafikus programnyelvek felhasználásának keretei sokkal szűkebbek, valamint nehezebb bennük gyorsan dolgozni.

Amíg tehát a kezdőknek a grafikus programnyelvek tanulása egyszerűbb, a tapasztaltabbak már szövegeset használnak. A kettő között viszont sok programnyelv esetében nincs átmenet. Mivel a programnyelveket – így bármely grafikus és szöveges

programnyelvet is – más fejlesztők állították össze különböző módon, ezért a grafikus programnyelvből szövegesbe történő átállás általában nehézséget okoz. Nem csak azért, mert különböző parancsok, függvények és funkciók vannak a két nyelvben, hanem azért is, mert a szöveges programnyelvben eltérő helyesírást kell használni, mint a grafikus felületen. Így a figyelem megosztása ismét előnytelenül alakul azok számára, akik szöveges programnyelvet szeretnének tanulni. (Leutenegger, Edgington, 2004)

Az itt leírt problémás esetek külön nehézséget okoznak azok számára, akik kevésbé érdekeltek a programozásban. Mivel a szöveges programozás tanulása minden esetben nehezebb, mint a grafikus felületeké, a tanulók motiválására különös figyelmet kell fordítani. Általában azok a diákok, akik már képesek programot létrehozni valamilyen grafikai programnyelv felületén, elveszítik a lelkesedésüket a szöveges nyelvre történő átálláskor. Jóval később lesz ugyanis sikerélményük, mint a vizuális program esetében, valamint az elméleti háttér elsajátítása is tovább tart.



**4. ábra: Code Hero, programozást oktató játék**  
*Forrás: Gamasutra, 2013*

A kidolgozott módszer alapötlete néhány számítógépes játék alapján fogalmazódott meg. Főleg kiskorúak esetében a programozás játékon keresztül történő oktatása egy kézenfekvő módszer a számítógép használata miatt. (Pap-Szigeti, Pásztor, 2008) (Brain, 2009) Több programot is írtak már ilyen célból, amely valamilyen vizuális, játékos módszerrel próbálja a programozás logikáját megmutatni és begyakoroltatni a tanulóval. (Anderson, McLoughlin, 2007) (Gamasutra, 2013) (Pinola, 2013) (Romano, 2013) (Finley, 2013) (Heussner, 2013) (Kurzweil, 2013) (Primerlabs, 2013) Hatékonyság

szempontjából valóban jó eredményt mutat a legtöbb, viszont a behatárolt funkciók miatt ezek meg sem közelítik egy valódi programozási nyelv komplexitását. (Teach Kids to Program, 2013) Ezért ezek a játékok nem alkalmazhatóak széles körben.

Azok a programozói programok, amelyek szintén grafikus nyelvre építenek és funkciókban is bővelkednek, általában már széles körben alkalmazhatóak. A legtöbb ilyen grafikus programnyelvbe lehet importálni szöveges kódokat is. A célja ennek, hogy a grafikus és szöveges programnyelvek előnyét is ki lehessen használni egy programnyelven belül. Azonban ezek általában egymást kiegészítő jellegű funkciók, vagyis nem lehet a grafikai megoldásokat egy az egyben lecserélni szöveges sorokkal. Csak néhány programban van erre lehetőség, a bemutatott oktatási módszer is ezen alapul.

## **2.1 A hagyományos elméletorientált oktatás**

A tradicionális, könyv alapú oktatás hatása még ma is jelen van a programozásban. Ez a módszer azzal az egyszerű feltételezéssel él, hogy amíg a diáknak nincs egy megfelelő elméleti háttere a programozás alapjairól, addig nem képes gyakorlatban alkalmazni azt. Éppen ezért minden programozói kurzus elméleti összefoglalóval kezdődik.

Mivel a számítógépes programozás főleg gyakorlatra épül, elengedhetetlen az elmélet oktatása mellett a tapasztalatszerzés is. Az előzőleg említett feltételezés miatt ezt úgy valósítják meg, hogy az egyszerű elemektől a bonyolultabbakig haladva, először azok elméleti használatát és a használat feltételeit magyarázzák el a tanárok. Ezután a diákoknak lehetőségük van kipróbálni ezeket az elemeket számítógépükön, néhány egyszerű példaprogramot készítve. Egy teljes program elkészítéséhez szükséges tudás – főleg az első órákon – még nem áll a diákok rendelkezésére, valamint rendkívül sok időt is venne igénybe minden elem kipróbálásához egy kész programot írni. Ezért a példaprogramok általában csak néhány, egyszerű funkciót tartalmaznak. Egy ilyen példa elkészítése ugyan alátámasztja az elem elméleti működését, viszont a tényleges alkalmazásának helyét és módját nem mutatja be.

A diákoknak szükséges a programozás alkalmazásához olyan tapasztalatokat szerezniük, amelyek során egy vagy több bonyolultabb és összetettebb programot hoznak létre. Ennek a megvalósítása érdekében gyakran kapnak a diákok házi feladatokat, amelyek keretében otthon, önálló munkával kell elkészíteniük a nehezebb feladatokat. Ez a módszer már lényegesen közelebb áll a valódi programozás begyakorlásához, hiszen itt a diáknak már önállóan kell elvégeznie a programtervezést és megoldani a korábban tanult elemek alkalmazhatóságát. Hátránya, hogy a szöveges programnyelvben a tanulók csak lassabban képesek munkát végezni, azaz a házi feladat is lassabban fog elkészülni. Számonkérés szempontjából tehát előnytelen, hogy erősen limitált a kiadható házi feladatok száma és bonyolultsága egy adott időszakon belül.

A népszerű, oktatásban is alkalmazott programnyelvek elmélete általában a programozási struktúrákra, kódok felépítésének módszerére tér csak ki. A mai számítástechnikában gyakorlatilag mindenhol megjelenő képek, videók, animációk és hangok használata legtöbbször csak érintőlegesen jelenik meg. Ennek egyik fő oka, hogy a szépen kidolgozott grafikai felületek programozásához ismét sok elméleti háttérre van szükség. Továbbá ennek az elméletnek ismeretében sem egyszerű mindezt megvalósítani, mivel egy grafikai felület megtervezése komoly munkát igényel. Arra, hogy egy diák egy mai grafikával rendelkező program elkészítését megtanulja a programozás oktatás keretein belül, szinte lehetetlen feladatnak bizonyul.

Mindennek a következménye, hogy egy programnyelv részletes, átfogó ismertetése a hagyományos módszerrel akár több évet is igénybe vehet. Ennek a hosszú időnek tudható be, hogy a diákok menet közben elveszítik kezdeti lelkesedésüket. (Tóth, Johanyák, 2007)

## **2.2 A gyakorlaton alapuló oktatás**

A modern számítógépeknek és szoftvereknek köszönhetően a hagyományos oktatási módszerrel szemben lehetőség van egy ellentétes szemléleten alapuló, gyakorlatiasabb oktatási mód használatára is. Ez a módszer elveti a hagyományos alapvető feltételezését, miszerint elengedhetetlen az elméleti ismeret a programozás gyakorlati alkalmazásához. Ehelyett azt állítja, hogy egy megfelelő grafikus program

használatával, minimális elméleti tudással is lehet már az alkalmazást tanulni. A minimális elméleti háttér kizárólag azokat az információkat tartalmazza, amelyek elengedhetetlenek a program használatához. Ezután a diák már képes egyszerű programokat írni és megfelelő motiváció esetén további funkciókkal kísérletezni az adott felületen.

A program kezelésének elsajátítása után a tanár – hasonlóan a hagyományos oktatáshoz – az egyszerű funkcióktól kezdve a bonyolultabbakig bemutatja az alkalmazást példaprogramok segítségével. A különbség, hogy amíg a hagyományos módszer esetében az új elemmel még egyáltalán nem találkozott korábban a diák, addig a gyakorlatias módszerrel a felület kezelésének elsajátítása során van némi előzetes információja róla. A motivált, előre dolgozó diákok tapasztalatot is könnyedén szerezhetnek az új funkció működéséről. Tehát az egyes elemek tanulása lényegesen gyorsabban történik, mint a hagyományos módszer esetében, ezért a gyakorlati alkalmazására is több idő jut.

Ahhoz, hogy a programnyelv helyesírása ne lassítsa le a tanulás folyamatát, mindezt grafikai programnyelv segítségével kell elvégezni. Befolyásolja továbbá a tanulás sebességét a diák motivációja, ezért fontos, hogy a választott programnyelv alkalmas legyen színes, grafikus megjelenésű programok vagy játékok elkészítésére a lehető legegyszerűbb módon. Ha a diák már a legelső órán képes létrehozni egy önállóan készített, színes képeket és animációkat tartalmazó programot, attól kedvet kap a tanulás folytatására, nagyobb lelkesedéssel fog az egyes elemek megismerésével és begyakorlásával foglalkozni.

A programtervezés során általában egy adott feladatnak kell megfelelni, a programnak végre kell hajtania a megadott funkciókat. Ezzel szemben ha a diák egy olyan feladatot kap, amely csak kevés konkrétumot és több általános leírást tartalmaz, az ismételten motivációt adhat neki, hogy kiegészítse a kötelező programot a saját kisebb-nagyobb ötleteivel. A számítógép szórakoztatóiparban betöltött igen jelentős szerepe miatt sokan foglalkoznak szívesen olyan program megírásával, amely használata szórakoztatja a felhasználót, örömet szerez. Tehát amennyiben a diákoknak egy kevésbé megkötött kötelező feladatot kell megoldaniuk, amely valamilyen számítógépes játék megtervezését és megalkotását foglalja magába, valószínűleg kreatívabb, egyedibb

megoldásokat eredményez, amellyel a tanulók is szívesebben foglalkoznak. A konkrét, életszerű és szabadon választott témák erősen motiváló hatásúak.

Ennek az oktatási módszernek problémája lehetne egyszerű grafikus programnyelv alkalmazása esetén, hogy nehezen lép ki a programnyelv által húzott szűk keretek közül, ezért nem ajánlható azok számára, akik komolyabb szinten szeretnék programozással foglalkozni. Ahhoz, hogy a módszer ne csak egy ízelítőt adjon a kezdő programozók számára, át kell térni szöveges programozásra, viszont a korábban leírt problémák miatt ezt nem könnyű megvalósítani. Javasolt egy olyan grafikus programnyelv használata, amelyben lehetőség van a képi elemek szöveges funkciókra történő lecserélésére. Amennyiben a programnyelv nem rendeli a szöveges formát a grafikai alá vagy fordítva, hanem egymást kiegészítve és helyettesítve is képes működni, az átállás a szöveges nyelvre kevésbé lesz ugrásszerű. Másik feltétele az egyszerű átállásnak, hogy a szöveges és grafikai nyelv ugyanazon az elven, hasonló szintaktikával működjön. Egy jól megválasztott szoftverrel és programnyelvvél könnyebb megtartani a diákok motivációját az átállás során is.

Mindent összevetve ez az oktatási módszer alkalmas arra, hogy a diákok szélesebb körét megtanítsa a szöveges programozásra. További előnye, hogy az egész folyamat a diákok lelkesedését magas szinten tartja, ezért ők is elégedettebbek lesznek ezzel a módszerrel, illetve ugyanannyi idő alatt sokkal több információt és technikai megoldást lehet megtanítani, mint a hagyományos módszerrel. A módszer kihasználja egyaránt a grafikus és szöveges programozási nyelvek előnyeit, miközben az oktatás folyamata végig a gyakorlatot helyezi előtérbe. Ennek a módszernek a hatására a diákok nem csak tanulják, hanem alkalmazni is tudják majd a programtervezést és programozást.

### 3. A fejlesztői eszköz

A bemutatott oktatási módszer alkalmazásához elengedhetetlen egy olyan programozói szoftver használata, amely a leírt szisztéma szerint épül föl és minden tekintetben jól használható.

Egy holland informatikatanár, Markus Hendrik Overmars kifejlesztett egy grafikus felületet alkalmazó játékfejlesztői programot, a Game Maker-t. A célja ezzel a programmal az volt, hogy azok is képesek legyenek számítógépes játékokat készíteni, akik nem tudnak programozni. (Overmars, 2004) Az első verziót 1999-ben mutatta be, amelyet az internetről ingyenesen lehetett letölteni, ez még csak kevés, kezdetleges funkcióval rendelkezett. Első népszerű változata a Game Maker 4.3, amely egy stabil motorral működő, szöveges programozást is tartalmazó szoftver. Különlegessége itt jelentkezett először: a grafikus és a szöveges programozási felület is ugyanazt a programnyelvet használja és egyik sincs alárendelve a másiknak.

A Game Maker programnyelve a GML (Game Maker Language), amely a játékfejlesztői szoftver szöveges felületének egyszerű alkalmazhatósága céljából lett létrehozva. Későbbi népszerű verziói a Game Maker 6.1 és a Game Maker 8.0. Az előbbi esetében a grafikus felhasználói felületet az elvárásoknak megfelelően finomították, valamint a GML rendkívül sok új funkciót kapott. Az utóbbi képes a számítástechnika legújabb módszereit alkalmazni, így a Game Maker egy naprakész fejlesztői eszközzé vált játékfejlesztők számára. Pár éve felvásárolta a programot egy angol cég, a YoYo Games. Azóta a korábbiakhoz képest más irányban történik a játékkészítő program fejlesztése. Legújabb verziója a Game Maker: Studio v1.2-es számot kapta, ez már sokkal inkább haladó programozóknak ajánlott mint kezdőknek, a grafikai felülettől függetlenül.



**5. ábra: Game Maker 8.0 logója**  
*Forrás: saját készítés*

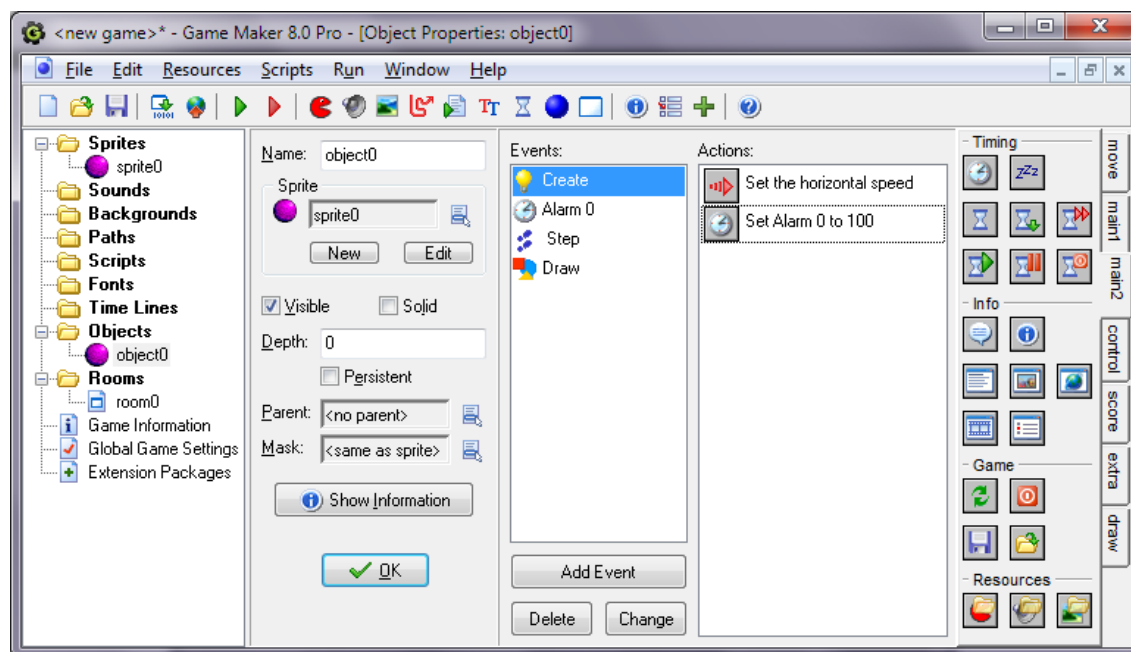
A gyakorlatias oktatási módszer alkalmazásához javasolt a Game Maker 8.0 használata, a továbbiakban erről a verzióról lesz szó. Ebben a grafikus programozási felület megegyezik a korábbi verziókban kifejlesztett, könnyen átlátható és kezelhető „fogd-és-vidd” felülettel, ami a kezdő programozók számára ideális. Ezzel szemben a szöveges felületen keresztül elérhető GML rendkívül sok funkcióval és lehetőséggel bővült, amely a szöveges programozásra történő átállás után rengeteg lehetőséget biztosít. Kezdő és tapasztalt programozók egyaránt tudják alkalmazni ezt a programot tudásszintjüknek megfelelően.

### **3.1 Grafikus felület és a GML**

A Game Maker népszerűségét az egyszerű és könnyen átlátható grafikus felületének köszönheti. A program objektum orientált felépítésű, amit a rajzokkal és különböző ablakokkal a program remekül szemléltet. Az egyes nyersanyagok beállításának elhelyezkedése szintén jól csoportosított, a leírások tömörek és érthetőek. A Game Maker súgó felülete, különösen a GML-et leíró rész jól szerkesztett és könnyen használható. Az egész programban a kezdők könnyedén kiismerhetik magukat, ami a későbbiekben felgyorsítja a munkát.

Ugyanez az egyszerűség jellemzi a „fogd-és-vidd” felületet is. A Game Maker eseményeket és akciókat használ a program futtatásához, amelyek meghatározásához két lehetősége van a fejlesztőnek. Egyik lehetősége a grafikus felület használata, ahol színes ábrák segítségével tud előre definiált parancsokat futtatni. Ilyen előre definiált parancsokból sok különböző típusú van, ezek mindegyike megfeleltethető egy vagy több parancsnak a GML programnyelvben. Minden ábrához tartozik egy rövid, pár szavas angol leírás, amely az ábra használatát részletezi. Mivel az összes esemény és akció egy helyen, felsorolásszerűen jelenik meg, a programozó azonnal láthatja, hogy milyen lehetőségei vannak a játékfejlesztés során, ez adhat újabb ötleteket a további fejlesztésekhez. Továbbá az ábrák között egyszerű, magától értetődő és bonyolultabb, programozási struktúrát alkalmazó funkció is szerepel. Így a programozó az egyszerű funkciókat felhasználva szinte azonnal létre tud hozni egy programot, de ugyanezt a felületet használva később a programozói struktúrákat is be tudja gyakorolni, mielőtt

áttérne a szöveges programozásra. A felhasználói felület felépítése, valamint az akciók leírása és beállításukat segítő ablaka nagyban megkönnyíti a programozás alapjainak megértését.



**6. ábra: Game Maker grafikus felülete**

*Forrás: saját készítés*

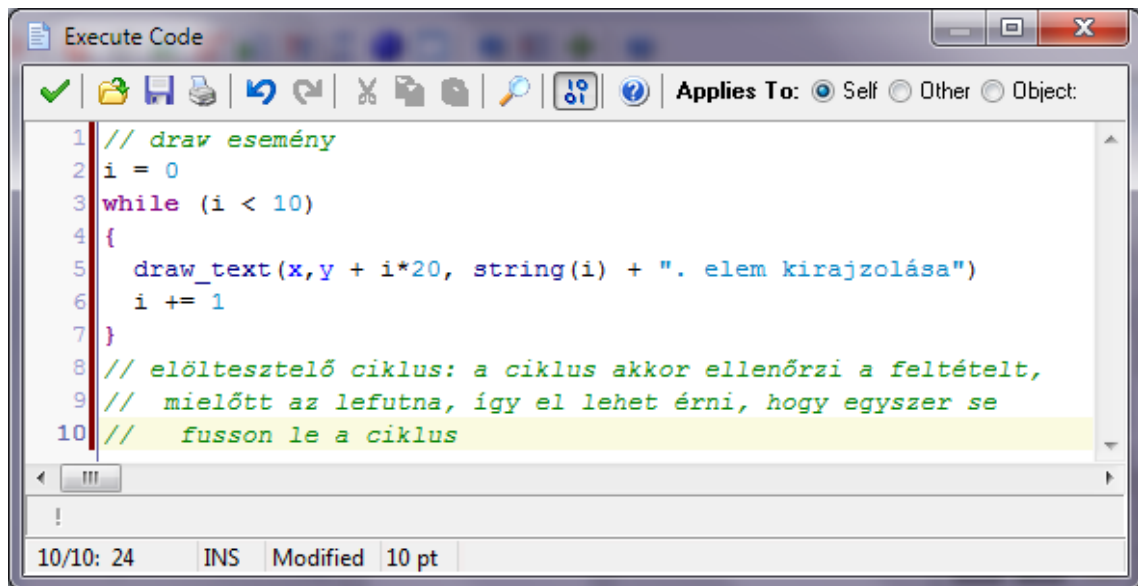
A „fogd-és-vidd” felület használata tehát a színes ábráknak és az ablakoknak köszönhetően nem okoz problémát egy kezdő számára. Ahhoz, hogy ezekkel a parancsokat tartalmazó akciókkal képes legyen programot fejleszteni, elég a programtervezésre és a funkciók összeállítására koncentrálni. Gyakorlatilag minimális elméleti tudással képes a színes ábrákat rakosgatni és megnézni, hogy hogyan fog hatásukra működni a program. Mivel csak a végső program felépítésének logikájával kell foglalkoznia, ezért sokkal hamarabb be tudja gyakorolni a programozási formákat és sokkal hamarabb el tudja sajátítani a programozói gondolkodásmódot. Az egyszerűbb akciók által nyújtott sikerélmény után a bonyolultabbak elméletének megtanulása szintén történhet gyakorlatiasabb módon, hiszen már képes a fejlesztő önálló programot létrehozni és valamilyen szinten módosítani azt. Az új funkciókat a korábbi programba beillesztve ki lehet egészíteni azt, vagy akár egy új programot létrehozni, amelyben az összes eddig tanult funkció szerepelhet. Mivel a Game Maker-ben egyszerűen, pár kattintással létre lehet hozni egy új programot, a funkciók tanulásához használt példafájlok lehetnek bonyolultabbak, valóságosabb programok is.

Azok a kezdő programozók, akik a Game Maker grafikus felületét használják, hamar szembesülhetnek azzal, hogy hol vannak ennek a felületnek a határai. Ugyanakkor ez az idő éppen elegendő ahhoz, hogy legyen a fejlesztőnek már egy elképzelése a programok összeállításáról, a programtervezésről és kellő mennyiségű gyakorlattal rendelkezik már ahhoz, hogy elkezdje a szöveges programozást megtanulni. Ilyenkor képes már önállóan játékot készíteni, amely képet, hangot, animációt kezel, és amelynek a logikáját a fejlesztő saját maga építette össze. Vagyis mire eléri a „fogd-és-vidd” felület korlátait és el kell kezdenie a szöveges programozást megtanulni, addigra már rendelkezni fog egy általános programfejlesztési gyakorlattal, sokkal könnyebben tud már saját programot tervezni. Ilyenkor kerül előtérbe a GML szerepe.

A GML, azaz Game Maker Language a Game Maker beépített programnyelve, amelyet szöveges programozás esetén kell használni. Ez a nyelv rendkívül sok beépített parancsot tartalmaz, amelyek egy része egyezik a grafikus „fogd-és-vidd” felületben megismert akciókkal, de a népszerű szöveges programozási nyelvek alapvető felépítését használja. Ez a programnyelv alkalmas arra, hogy a grafikus felületen megismert akciókat vagy azok egy részét a fejlesztő szöveges formátumúra cserélje. Ennek a lépésnek azért van jelentős szerepe, mert a már megismert funkciókat szöveges formára átírva a programozónak kizárólag a helyesírásra kell figyelnie ahhoz, hogy a megírt sor működjön. Ilyen módon lecserélve az összes ábrát szöveges formára továbbra is átlátható marad a program felépítése és a szöveges nyelv alkalmazását is azonnal át lehet látni. Ezek után a megismert szöveges funkciókat a fejlesztő képekben gondolkozva, de szövegesen leírva fogja alkalmazni a korábbi tapasztalatok alapján.

A szöveges formának ilyen szintű begyakorlása után már lehetősége van a fejlesztőnek a többi GML funkció megismerésére és használatára, átlépve a korábban megjelenő határt. Ebben nagy segítségére lehet a Game Maker súgó ablaka, amely tartalmazza az összes GML parancsot és azoknak minden lehetőségét. A sikerélménye továbbra is megmarad a programozónak, hiszen az első program létrehozása óta ugyanazt a módszert kell alkalmaznia újak elkészítéséhez, mindig egy kisebb résszel kiegészítve. A programfejlesztés tanulásának ilyen szintű egymásra épülő, fokozatos elsajátítása nagy hatékonysággal bír, hiszen a programozó a Game Maker első megismerése után a saját tudásának megfelelő mértékben jut újabb információkhoz. Végeredményben a GML

teljes szintű megismerésével bármilyen, mai követelményeknek megfelelő játékot vagy egyéb grafikus programot képes létrehozni. Továbbá mivel a GML szerkezete hasonlít a többi népszerű objektum orientált programnyelvhez, a programozónak kevésbé lesz idegen ezeknek a nyelveknek a megértése, valamint a GML-ben szerzett alkalmazásokkal kapcsolatos tapasztalatát más programnyelvekben is könnyen fel tudja használni



```
1 // draw esemény
2 i = 0
3 while (i < 10)
4 {
5     draw_text(x,y + i*20, string(i) + ". elem kirajzolása")
6     i += 1
7 }
8 // előtesztelő ciklus: a ciklus akkor ellenőrzi a feltételt,
9 // mielőtt az lefutna, így el lehet érni, hogy egyszer se
10 // fusson le a ciklus
```

**7. ábra: GML-ben írt kódrészlet**

*Forrás: saját készítés*

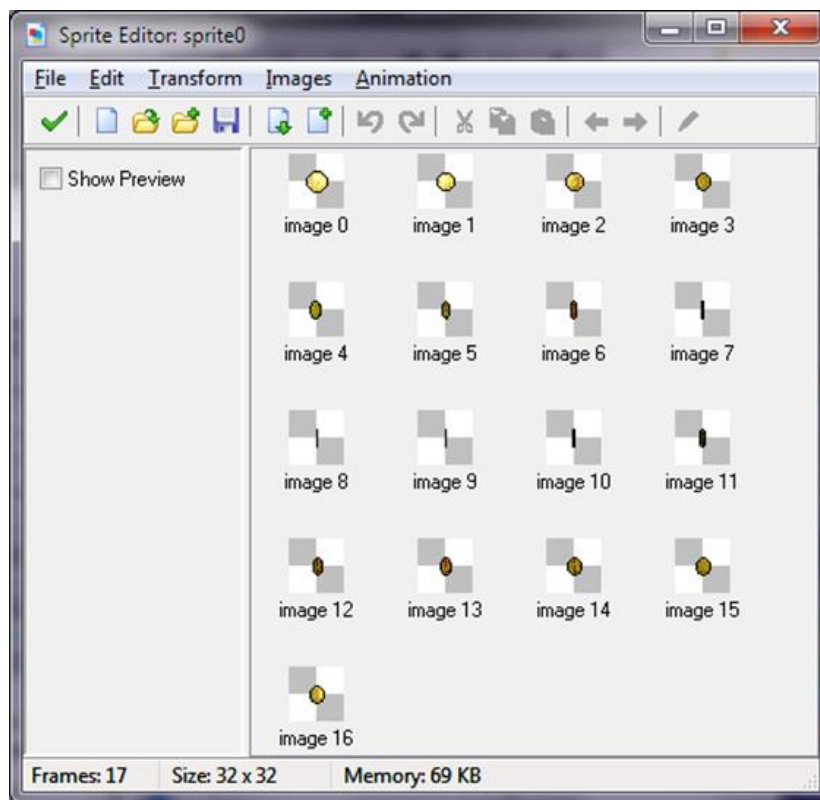
A Game Maker-ben a játékfejlesztéssel kapcsolatos funkciók az egész programban jelen vannak, ezért a programban nagy figyelmet fordítottak a képek és hangok kezelésére, valamint az ütközésellenőrzésre is. Ennek köszönhető, hogy a Game Maker nem csak játékfejlesztésre, hanem bármilyen animált program elkészítésére is alkalmas. Ezt a tulajdonságát kihasználva könnyedén lehet vele olyan programot is fejleszteni, amely nem csupán funkcióiban felel meg az elvárásoknak, de mindezt szép, eladható formában teszi.

## 3.2 Multimédia

A mai számítógépes rendszerek – asztali számítógépek, laptopok, táblagépek, telefonok – mindegyikében a grafikus operációs rendszerek és a grafikus felhasználói felületek a meghatározók. Ennek az oka, hogy könnyebben kezelhetőek széles körben, mint a

parancssoros felületek. Nem meglepő tehát, hogy a legújabb fejlesztésű szoftverek is mind grafikus felületet használnak. Viszont a programozás szempontjából nehezebb feladat grafikus felület létrehozni, mivel ott a funkcionalitáson kívül sok egyéb szempontot is figyelembe kell vennie a programozónak. Ilyen szempontok a grafika megtervezése (sok esetben megrajzolása is), grafikus elemek megjelenítésének programozása, azok hibamentes működése, programozott animációk megírása, esetleg hangok beillesztése, felhasználóbarát felület készítése és az egyértelmű kezelés megvalósítása. Ezeknek a szempontoknak egyaránt megfelelni nehéz feladat nem csak programozási szempontból, hanem mert megfelelő, multimédiával kapcsolatos ismereteket is igényel.

A játékfejlesztés oktatásának egyik előnye, hogy tartalmaz valamennyi multimédia ismeretet is, amely ad némi rálátást a digitális képek, animációk, videók és hangok szerkesztésére. Egy ilyen, alapszintű ismeret megszerzése a számítástechnika korában bármely területen hasznossá válhat, a programozásban pedig szinte elengedhetetlen napjainkban. A Game Maker a programozáson túl betekintést ad a raster grafikus képek szerkesztésének alapjaiba, valamint az animáció készítésbe. Ugyanakkor beépített hangszerkesztő funkciója nincsen, ezért ennek megismeréséhez egy másik program, az Audacity javasolt.



**8. ábra: Animáció készítés a Game Maker felületén belül**

*Forrás: saját készítés*

Az Audacity egy ingyenes, nyílt forráskódú, magyar nyelvet is támogató szoftver. (Audacity, 2013) Alkalmas bármilyen digitális hanganyag vágására, módosítására és felvételére. A programozók ezen a programon keresztül könnyedén elsajátíthatják a digitális hangokkal és zenékkal kapcsolatos általános tudnivalókat, amely tudás később jól alkalmazható játék- és programfejlesztéskor.

Végül egy másik ajánlott program a Windows Movie Maker, amely segítségével bárki könnyedén megismerheti a videó szerkesztés sajátosságait. Ez egy ingyenes, magyar nyelvet is támogató videó vágó és módosító program, amely általános használatra tökéletes. (Windows Movie Maker, 2012) Programozók esetében bemutató videók megvágására javasolt, egyszerű kezelőfelülete gyors használatot biztosít.

## **4. Az oktatási módszer alkalmazása**

A kidolgozott gyakorlati oktatási módszer alkalmazását valamilyen időbeosztás alapján, képzések formájában kell alkalmazni. Terjedelme fél vagy egy éves, a célcsoport sajátosságaitól függően. A programozás alapszinten történő elsajátítása és begyakorlása alapszintű multimédia ismeretekkel kiegészítve fél évet, azaz 12-13x1,5 órás alkalmat vesz igénybe azok számára, akik korábban még sosem programoztak. Azoknak a diákoknak, akik elvégezték a kezdő kurzust és megtanulták az alapokat, további fél évre, azaz összesen 25-26x1,5 órás alkalomra van szükségük, hogy haladói szinten elsajátítsák a programozást.

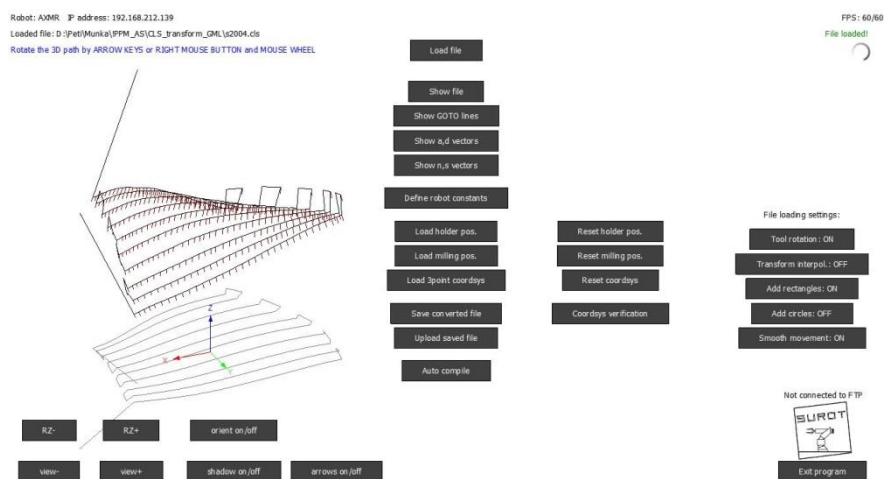
A tanórákat kis létszámú, legfeljebb 20 fős csoportok részére kell megtartani. A kisebb közönség előnye, hogy az órát könnyebb interaktívvá tenni, ezáltal az előadások érdekesebbek és tanulságosabbak lehetnek a hallgatók számára. Szükséges, hogy minden óra számítógép teremben legyen megtartva, ahol az összes diák képes folyamatosan követni az előadó magyarázatát a saját gépeiken keresztül. Számítógép használata nélkül a programozás tanulása rendkívül nehézé válna. Javasolt, hogy egy kurzust egy előadó tartson meg, legfeljebb a kiegészítő órákra hívható meg vendég előadó, amennyiben ez szükséges. Amennyiben egy kurzus anyagát több előadó adná le, az előadások lineáris felépítése miatt megtörné az egységes folyamatot és könnyen előfordulhatna, hogy a diákok számára kevésbé átláthatóvá válna a tananyag. Az előadók előadásmódja, szakmai tudása és segítőkészsége között minél nagyobb a különbség, ennek bekövetkeztére annál nagyobb az esély.

### **4.1 A kurzusok célcsoportja**

A kurzusok elsősorban a kezdő és teljesen kezdő programozóknak szólnak, kortól függetlenül. Mivel a programozás ilyen szinten történő elsajátításához elegendő a középfokú matematikai érettségi tudás, ezért a célközönségbe a középiskolások és a felsőoktatásban humán jellegű szakon tanuló diákok is beletartoznak. Érdeklődési kör szempontjából tehát nem tesz különbséget a módszer.

Ugyanakkor azoknak, akik már régebb óta foglalkoznak programozással, de még újdonság lehet számukra a haladó programozói kurzus, jóval nehezebb dolguk van. Mivel a haladó órák a kezdőt követve, folytatólagosan épülnek fel, a haladó diákoknak a kezdő alkalmakkor kihagyott ismereteket be kell pótolniuk. Ez azért problémás főleg, mert a GML egy kevésbé népszerű nyelv, amelyet kizárólag a Game Maker használ. A haladó kurzus viszont végig ezzel a nyelvvel foglalkozik, tehát aki nem ismeri korábban a GML-et, annak nagyon nehéz ezt pótolni. Ezért a haladó csoportban szükség van 2 felzárkóztató órára, ahol a kezdő kurzus órái kerülnek összegzésre, hogy utána minden diák közel azonos tudással folytassa a tanulást.

A GML alkalmas profi programok elkészítésére is, amennyiben valaki több éven keresztül aktívan használja a Game Maker-t. Ugyanakkor, ahogy a grafikus felületnek, úgy a GML-nek is van egy felhasználhatósági kerete. Ezt a határt a GML esetében nagyon sokáig tart elérni, és utána is léteznek további alternatív módszerek, például más programnyelvből importált parancsok és függvények használata. Viszont azoknak a programozóknak, akik valamilyen másik, népszerűbb programnyelvben már tudnak profi szinten programozni, nem javasolt a GML-re történő áttérés.



**9. ábra: Ipari robotokat vezérlő program GML-ben elkészítve**  
*Forrás: saját készítés*

A programozás oktatása, ahogy a hagyományos modellben, a gyakorlatias módszerrel is számítógépek segítségével történik. Ezért alapfeltétele az oktatásnak, hogy rendelkezésre álljanak számítógépek, amelyeken futtatható a Game Maker 8.0-ás változata. A módszer várhatóan kis, 15-20 fős csoportok esetén hatékonyabb az így

interaktívvá tehető órák miatt. Ezeknek a feltételeknek a megvalósítása – érdeklődéstől függően – elsősorban felsőoktatási intézményekben, valamint olyan középiskolákban jöhet szóba, amelyekben a diákok matematikai tudása már megfelelő szintű (főleg koordináta geometriára és függvénytani alapismeretekre van szükség).

## **4.2 A kurzusok tananyaga**

Az alábbiakban egy összesítés található a kurzusok során átadott, bemutatott tananyagról. Mind a kezdő, mind a haladó kurzus esetében jól lehet látni, hogy milyen jellegű feladatokra lesznek képesek a diákok kurzus elvégzése után.

### **4.2.1 Kezdő kurzus**

A kezdő kurzus azoknak a diákoknak szól elsősorban, akik még sosem programoztak életükben és éreznek motivációt, hogy belefogjanak (például kíváncsiság vagy a játékfejlesztés megismerése miatt). A kurzus fél éve körülbelül 12, egyenként 1,5 óra hosszú előadást foglal magába, amely három részre oszlik témakörök alapján.

Az első rész (négy alkalom), amellyel a kurzus is indul, a grafikai programozási felület részletes megismeréséről szól. Itt már az első órán a Game Maker általános kezelésének megismerése után lehetőséget kapnak a diákok első programjuk elkészítésére, amely egy labirintusos játék alapvető elemeit tartalmazza. Innentől kezdve minden óra egy egyre bonyolultabb játék létrehozásával zárul, amely tartalmazza az aktuális előadáson bemutatott funkciókat is. A második és harmadik órán a diákok megismerkednek az akciók és események használatával különböző példafájlokon keresztül. Az utolsó alkalom további beállítási lehetőségekről és egy bonyolultabb program tervezéséről és elkészítéséről szól, amely főleg önálló munkát jelent.

A kurzus második része (négy alkalom) a GML bevezetéséről és alapvető használatáról szól. Az első két alkalmon az akciók kóddal történő lecserélésének és a kódok használatának bemutatása a cél. A harmadik és negyedik órán az általános programozási eljárások és struktúrák szerepelnek, amelyek a többi programnyelvben is

megtalálhatóak, és amelyek nem találhatóak meg az akciók között. Mivel ezekből a szerkezetekből csak kevés van, ismét több időt lehet fordítani az óra közbeni önálló programfejlesztésre. Az utolsó alkalom végére képes lesz a diák egy olyan programot írni programsorok segítségével, amely a hagyományos programozási oktatásban szereplő eljárásokat tartalmazza.

A kurzus előbb említett két fő része nem feltétlenül folytatódólagos, közülük ékelve jelenik meg a harmadik rész (négy alkalom), amely az egyéb kiegészítéseket tartalmazza. Ez a négy alkalom a témakörtől függően illeszkedik be az órák anyaga közé. Ezek közül két alkalom kép, animáció és videó szerkesztésről, egy alkalom a hangszerkesztésről, valamint egy a programtervezés elméletéről szól. Az alábbi felsorolás az órák időrendi sorrendjét határozza meg rövid tartalmi leírással kiegészítve.

1. Game Maker kezelőfelületének megismerése, első program elkészítése
2. Egyszerűbb akciók használata, példafájlok készítése
3. Kiegészítés: raster grafika alapjai és videó szerkesztés
4. Bonyolultabb akciók használata, példafájlok készítése
5. Kiegészítés: programtervezés elmélete és gyakorlata
6. További beállítások, Game Maker lehetőségei, komplex példafájl készítése
7. GML megismerése, leggyakrabban használt funkciók, példafájlok készítése
8. GML megismerése, gyakran használt funkciók, példafájlok készítése
9. Kiegészítés: vektor grafika alapjai és animáció készítés
10. Egyéb programozási struktúrák, ciklusok és tömbök
11. Kiegészítés: hangszerkesztés és zene vágás alapjai
12. Egyéb programozási struktúrák, eljárások és objektumok kezelése

Az itt leírt 12 alkalmas kurzus számonkérése két házi feladat segítségével történik. Az első feladatsort rögtön az első óra után kell kiadni és a hetedik előadásig bezárólag kell beszélni. A második feladatsort a hetedik előadás után kell kiadni és az utolsó előadás utáni hét végéig kell beszélni. Mindkét feladatsor esetében lehetőség van egyaránt a „fogd-és-vidd” módszer és a GML szöveges nyelvének alkalmazására. A cél a meghatározott funkciók megfelelő működése és csak másodlagos szempont a használt parancsok „szépsége”.



**10. ábra: Játékkészítés a kezdő kurzus alapján**

*Forrás: saját készítés*

Ennek a kurzusnak a végére tehát a tanuló képes lesz programsoros kódokkal, objektum orientált nyelvben, grafikus felhasználói felületet alkalmazó programot vagy játékot létrehozni.

#### 4.2.2 Haladó kurzus

Ahogy ez korábban is említésre került, ennek a kurzusnak fontos feltétele, hogy a kezdő kurzusban elhangzott tananyaggal valamilyen szinten tisztában legyen a hallgató. A haladó kurzus szintén fél éve körülbelül 12, egyenként 1,5 óra hosszú előadást foglal magába, amely első két alkalma a Game Maker kezelőfelületével és a GML nyelvvel történő általános ismerkedés. Erre azért van szükség, hogy a maradék 10 alkalmat minden diák körülbelül azonos szintről folytathassa.

A további 10 órából 9 a GML különböző témaköreinek megismeréséről és a Game Maker határainak tágításáról, külső programok alkalmazásáról szól. Ezen kívül egy alkalommal a programtervezés kerül előtérbe, ahol bemutatásra kerül, hogy hogyan lehet megfelelni a mai számítógépes programok általános elvárásainak. Az alábbi felsorolásban az órák időrendi sorrendje olvasható rövid tartalmi leírással kiegészítve.

1. Ismétlés: A Game Maker felhasználói felületének általános ismertetése, példafájl készítése

2. Ismétlés: A GML általános ismertetése, gyakori funkciók, példafájl készítése
3. Függvények írása és alkalmazása, példafájlok készítése
4. Mozgások és felhasználói beavatkozások GML-ben, példafájlok készítése
5. Útvonalak és időcsíkok készítése különleges esetekhez, példafájlok készítése
6. Ütközésellenőrzések, példafájlok készítése
7. Csempék és nézetek kezelése, megjelenítés, felületrajzolás, példafájlok készítése
8. Kiegészítés: programtervezés, elvárások és megvalósítási módszerek
9. Nyersanyag, változó- és tömbkezelő funkciók, kódfuttatás, egyéb programozói adatstruktúrák, példafájlok készítése
10. Szöveges és bináris fájlok kezelése, programfuttatás, példafájlok készítése
11. Valóságű fizika használata, példafájlok készítése
12. Hálózat és internet kezelése, példafájlok készítése

Ahogy ez a felsorolásból is látszódik, ez a kurzus már csak azoknak ajánlott, akik komolyabban szeretnének foglalkozni a programozással a Game Maker-en belül. Ennek a kurzusnak az oktatási módszere ugyan nem tartalmazza a grafikus programozási felületek akcióit, viszont továbbra is gyakorlat orientált. Ahogy a kezdő kurzusban, itt is a játékfejlesztésen keresztül történik az elmélet ismertetése és a gyors sikerélmény megszerzésén alapul a diákok motivációja.

A kurzus számonkérése ismét házi feladat segítségével történik, de ebben az esetben csak egy feladat elvégzése kötelező. Kiadása a nyolcadik óra után, beszédése az utolsó óra utáni hétig bezárólag történik. Ennek a feladatnak egy komplex programnak kell lennie, amely átfogóan számon kéri az összes eddig tanult módszert és eljárást.



**11. ábra: Kész játék a haladó kurzus alapján**

*Forrás: saját készítés*

Ennek a kurzusnak a végére a tanuló széles körű, általános ismeretet szerez a mai programozási problémákról és lehetséges megoldásokról. Az utolsó óra végére képes lesz a hallgató egy tetszőleges, komplex program megtervezésére és megírására.

## **5. A próbakurzusok elégedettségvizsgálatának bemutatása**

A gyakorlati programozás oktatására kidolgozott módszer hatékonyságát az elmélet megfogalmazása után ki is kell próbálni. Ebből a célból készült egy felmérés, amely során a kezdő kurzus modelljének egy lerövidített és módosított változatát volt lehetőségük kipróbálni a jelentkezőknek. A felmérés célja, hogy igazolja vagy cáfolja az oktatási módszer várt eredményeit, valamint hogy rámutasson azokra a pontokra, amelyeken változtatni kell a későbbiekben. (Falus, 2004)

A felmérés kiírása során különböző internetes közösségi oldalakon keresztül történt a jelentkezés, önkéntes alapon. A jelentkezők egy kérdőív kitöltésének segítségével adtak információt magukról és programozói múltjukról, valamint regisztrálták magukat arra a próbakurzusra, amely során megismerhették a kidolgozott oktatási módszer lényegi részét. A jelentkezők nagy létszáma miatt egymással párhuzamosan három próbakurzus is indult, amelyek anyaga pontosan ugyanazt tartalmazta.

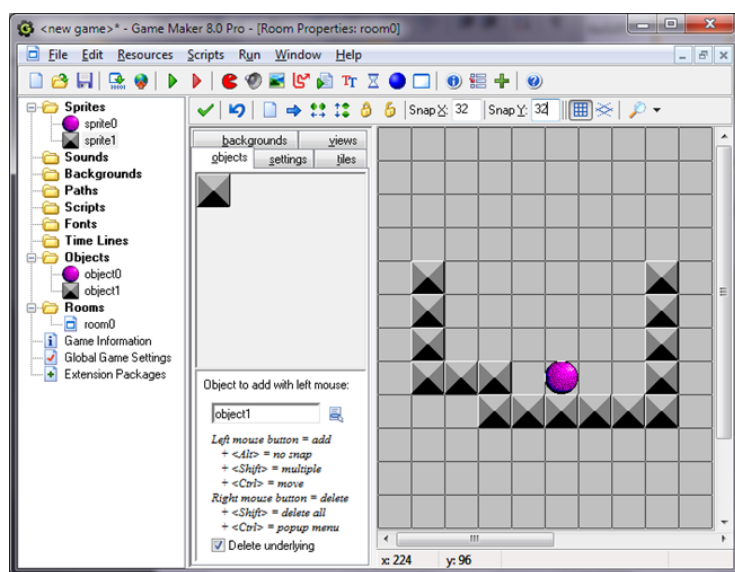
A három próbakurzus a BME Műszaki Pedagógia Tanszék támogatásával, a tanszék által biztosított számítógépteremben került lebonyolításra. A kurzusok négy alkalmasak voltak, ebből három alkalom egy-egy két órás előadást tartalmazott. A negyedik alkalmon egy egyórás dolgozat keretében történt a számonkérés a leadott anyagból, majd egy második, részletes értékelő kérdőívet kellett kitölteniük a jelentkezőknek. Továbbá volt lehetőségük egy házi feladatot leadni, amely eredménye beleszámított a dolgozat értékelésébe.

A felmérés során elsősorban a jelentkezéskor kitöltött kérdőív, a dolgozatírás után kitöltött kérdőív és a dolgozaton meg nem jelenő hallgatók számára összeállított kérdőív alapján készültek a statisztikai adatok. Ezen kívül az előadások jelenléti íveinek adatai, valamint a dolgozatok és házi feladatok eredményei is felhasználásra kerültek az elemzésben. (Kata, 2006)

A jelentkezők eloszlása a három kurzuson nagyjából egyenletes volt. A hallgatók beosztása az alapján történt, hogy melyik hallgató mikor ért rá az előadásokra bejárni és nem az alapján, hogy milyen korábbi tudással rendelkezett.

## 5.1 Az előadások tapasztalatai

A kurzusok mindhárom előadását a dolgozat készítője tartotta meg személyesen, jelenléti ívet vezetve. Az előadásokat a rendelkezésre álló két óra közel percre pontos kihasználtsága jellemezte, szünet nélkül. Mindhárom tartalmazott elméleti részeket, technikai megoldások ismertetését és példafájlok bemutatását. Céljuk a tananyag egy átfogó ismertetése és a Game Maker-ben rejlő lehetőségek bemutatása volt.



**12. ábra: Előadáson bemutatott példafájl**

*Forrás: saját készítés*

Mivel ezeket a próbakurzusokat a kezdő kurzus modellje alapján le kellett rövidíteni, a szerkezetében és tartalmában alapvető változtatások történtek. Ez a négy alkalom az előadások tartalmát tekintve a következőkből állt időrendi sorrendben.

1. Game Maker ismertetése, beállítási lehetőségek, első program létrehozása
2. Egyszerűbb és bonyolultabb akciók bemutatása, példafájlok készítése
3. A korábban megismert akciók áthelyezése GML-be, általános programozási struktúrák ismertetése, példafájlok készítése
4. Dolgozat írása a három előadás alapján, kérdőív kitöltése

Ahogy a felsorolásból is látszódik, a kezdő kurzus modelljének hat előadás anyaga lett három alkalomba sűrítve a próbakurzusok során (bár itt egy alkalom másfél óra helyett két óra hosszú volt), lényegesen kevesebb időt hagyva az anyag feldolgozására. Ezért az előadásoktól várható volt, hogy a hallgatók tömörnek, egyes esetekben érthetetlennek és rendkívül gyorsnak fogják ítélni.

Ugyanakkor mivel a próbakurzusokra önkéntes alapon jelentkeztek a hallgatók következmények nélkül, voltak, akiket nem fűzött semmilyen érdek, hogy otthon is foglalkozzanak a témával szabadidejükben. Azok, akik eljöttek az előadásokra és figyelmüket az előadóra fordították, biztosan kaptak legalább egy képet az anyagról. A felmérés szempontjából elengedhetetlen volt, hogy a hallgatók megismerjék az anyagot és megpróbálják azt elsajátítani, ezért az előadások megtartásának és „kötelezővé tételének” egyik fő célja ennek a problémának a minimalizálása volt.

A kitöltendő kérdőívek alapján várt eredményhez képest jóval sikeresebbek lettek az előadások. Érthetőségét tekintve sokkal pozitívabb eredményt mutatott a felmérést, mint amire számítani lehetett. Ugyanakkor a legtöbben úgy vélték, hogy gyors volt az előadás, de jó értelemben. Nem voltak unalmasok az órák, pörgősek voltak, nem unatkoztak alattuk a hallgatók. Természetesen ezen adatok esetében, főleg a tempó megítélésében nagy a szórás, vagyis volt, akinek ideális volt a tempó, de volt, akinek rendkívül gyors. Ennek az oka, hogy a jelentkezők rendkívül eltérő programozói múlttal rendelkeztek, voltak, akiknek minden információ új volt, és voltak, akik már ismerték az összes elméleti részt. Abban viszont minden hallgató egyhangúan egyetértett, hogy az előadások voltak az anyag megismerésének elsősorú forrása.

## **5.2 Az előadások alapján készült jegyzetek**

A próbakurzusok mindhárom előadásához tartozott egy-egy írásos segédanyag is. Ezek a jegyzetek sorban 25, 34 és 18 oldal hosszúságúak és mindegyik az adott előadás anyagát tartalmazta színes képekkel illusztrálva. A céljuk az volt, hogy a hallgatóknak legyen lehetőségük visszaolvasni az előadáson elhangzott elméleti háttérrel és otthon is tudják a leírtak alapján kezelni a Game Maker-t. A második jegyzethez mellékeltek megkapták az első két előadáson bemutatott példafájlokat is.

A jegyzetek gondos részletességgel készültek, sok illusztrációval, ezért várható volt, hogy a hallgatók inkább ezek alapján fognak a dolgozatra készülni, mint az előadások alapján. Szakmailag helytálló és kellő részletességgel és tömörséggel mutatták be a jegyzetek a Game Maker lehetőségeit és alkalmazásait.

Ugyanakkor, ahogy már az előadások esetében is említésre került, nem volt garancia a hallgatók többségénél arra, hogy egyáltalán elolvassák a jegyzeteket. Ezért a kurzusokat nem lehetett csupán a jegyzetekre és otthoni önálló munkára alapozni, szükség volt a jelenlétre és az előadások meghallgatására is a felmérés szempontjából.

Mindezt a kitöltött kérdőívek statisztikája is alátámasztja. Körülbelül a jegyzetek 50%-át olvasták el a hallgatók részben azért, mert az előadásról mindenre emlékeztek, részben meg azért, mert nem voltak rákényszerítve az otthoni munkára. A jegyzetek olvasottsága és a dolgozatok eredménye között nem húzható egyértelmű összefüggés a felmérés alapján. A többség hasznosnak és jól felépítettnek, érthetőnek minősítette a jegyzeteket minden szempontból. Ugyanakkor érdekes megemlíteni, hogy a jegyzetekben kidolgozott példafájlokat csak pár tanuló készítette el saját magának otthon. Ennek az oka, hogy a jegyzet kellő részletességgel leírja ezeket, a diákok nem érezték szükségét, hogy ezzel otthon külön időt töltsenek.

### **5.3 Az előadó**

A próbakurzusok három előadását és a dolgozat összeállítását, továbbá a feladatsorok elkészítését és értékelését is az oktatási módszer kidolgozója és ennek a dolgozatnak készítője végezte el.

Közel kilenc éve foglalkozom a Game Maker programmal és a GML programnyelvvel, készült már több díjazott számítógépes játékom is. (Scorpion games, 2013) Továbbá közel áll hozzám a tanítás is, négy éve rendszeresen tartok kis- és nagycsoportos különórákat és előadásokat általános iskolásoknak, gimnazistáknak és egyetemistáknak egyaránt. A sok pozitív visszajelzés alapján várható volt, hogy segítőkészségemmel és

magyarázataimnak érthetőségével, valamint szakmai tudásommal nem lesz problémájuk a hallgatóknak.

A kezdő és haladó kurzus modelljének sikerességéhez szükség van több előadóra is, akiknek az alábbi feltételeket maradéktalanul teljesíteniük kell. Az előadónak ismernie kell a Game Maker és a GML minden részletét, lehetőségét. Ehhez a tapasztalatszerzés elengedhetetlen a GML programnyelven belül, valamint javasolt további programnyelvek és multimédiás programok ismerete is alapszinten. Ha valaki a kezdő és haladó kurzust képes intenzíven, a legjobb eredményeket elérve végigjárni, az már elegendő a GML kezdő kurzus oktatásának elkezdéséhez. Haladó kurzus oktatásához szükséges az egy év tanulás után további egy év gyakorlat, hiszen az oktatónak itt jóval átfogóbb tudásra van szüksége és minden felmerülő kérdésre tudnia kell azonnal válaszolni.

A kiegészítő órák ugyan kissé eltérnek az anyag fő irányvonalától, viszont az ezeket az előadásokat megtartó oktatónak képesnek kell lennie az anyagot összekapcsolni a Game Maker használatával. Ehhez szintén elengedhetetlen a Game Maker és a GML alapos ismerete.

A próbakurzusok végén hasonlóan a többi részhez, az előadót is véleményezhették név nélkül a hallgatók. A vártnak megfelelő eredményt mutatott a statisztika, vagyis az előadó szakmabeliségével és segítőkészségével valóban nem volt probléma. Ugyanakkor érdekes eredményt mutat az a válasz, miszerint a többség szerint nagyban befolyásolja a leadott anyag megértését az előadó személye. Erre az oktatási módszer további kidolgozásakor érdemes nagyobb figyelmet fordítani.

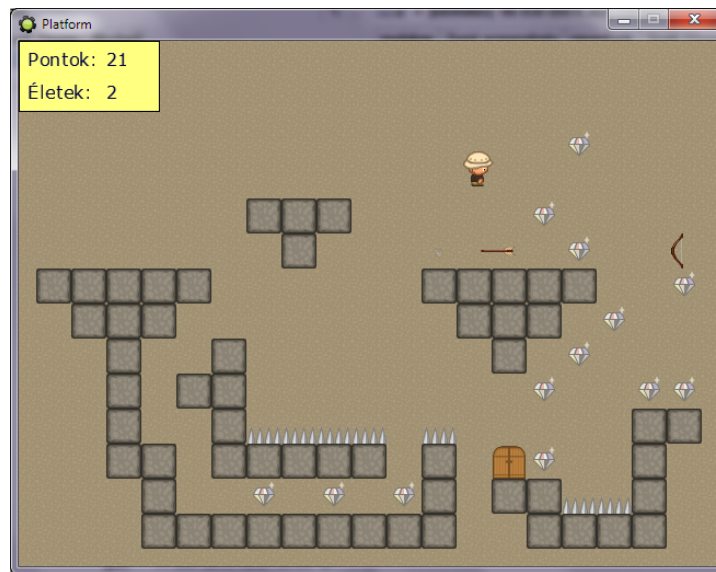
## **5.4 Számonkérés**

Mindhárom próbakurzus egy számonkéréssel végződött, amely egy dolgozat megírásából állt az utolsó alkalmon. Erre a hallgatók előre számíthattak, ugyanakkor fontos megjegyezni, hogy mivel a kurzusoknak nem volt következményük rájuk nézve, ezért nem is lehetett elvárni mindenkitől, hogy komolyan vegye a számonkérést és a jelenlétet. Az egyetlen motivációjuk az volt, ami miatt önként jelentkeztek a felmérés

meghirdetésekor, viszont mivel ez a motiváció személyenként nagyon eltérő, várható volt, hogy a dolgozatok eredményében, valamint a jelenlétben is nagy lesz a szórás.

Ebből a megfontolásból a dolgozatok feladatsorai szándékosan nehezek voltak. A hallgatók kaptak az utolsó alkalom előtt egy héttel korábban egy közepesen nehéz próba feladatsort, amely alapján szembesülhettek az általános követelményekkel és a feladatsor nehézségével. A három kurzuson három hasonló feladatsort kellett megoldaniuk egy óra alatt, amelyek mindegyike a próba feladatsor alapján készült, de több helyen nehezítve a feladatokat. Mindegyik feladatsor kizárólag az előadáson elhangzott anyagot és az ott megemlített kiegészítéseket kérdezte vissza, egyszerűbb és bonyolultabb feladatok formájában. A dolgozat helyes megoldása azért vált nehezzé, mert az egy óra alatt nagyon sok feladatot kért számon. Ha egy hallgatónak nem kellett külön gondolkoznia, a gombok vagy a súgó felületében keresgélnie, hanem azonnal tudta a megoldását a feladatoknak, akkor képes volt pont időben a feladatsor a végére jutni. Tehát a feladatsor azoknak a hallgatóknak a képessége között is különbséget tudott tenni, akik sokat készültek a dolgozat megírására, vagy akik már sok programozói tapasztalattal rendelkeztek. Egy valódi kurzus esetében természetesen nem lenne érdemes egy ilyen nehéz számonkérést készíteni, de a felmérésnek ez előnyére vált.

A hallgatók a második előadás után kaptak három fakultatív házi feladatot is. Ezek szintén egy-egy feladatsort alkottak jóval komplexebb és hosszabb feladatokkal, amelyek helyes megoldásához elengedhetetlen volt az első két előadás anyagának pontos ismerete. Egy házi feladat megoldásával legfeljebb 15%-ot (vagyis egy érdemjegyet) lehetett javítani a dolgozat eredményén, ez egy nehéz feladat pontszámával egyezett meg a dolgozat feladatsorából. A házi feladat megoldása tehát kevesebb motivációt adott a diákoknak, viszont itt megint visszautalva arra, hogy a felmérés fakultatív jellegű volt következmények nélkül, nem is lehetett komolyabb eredményeket elvárni az otthoni feladatokról. A házi feladatok célja az volt, hogy legyen lehetőségük a diákoknak otthon gyakorolni, valamint hogy megtapasztalják, milyen egy komplexebb játék fejlesztésén dolgozni a Game Maker felületén.



**13. ábra: Egy elkészített házi feladat**  
*Forrás: saját készítés*

A nehezen megoldható dolgozatoktól várható volt, hogy elveszik a jelentkezők kedvét és lelkesedését, mivel megírásuk után úgy érezhették, hogy a három előadás alapján kevesebbet kaptak az elvárttól, nem tudták a feladatokat megfelelően elvégezni. Ez olyan szempontból hátránya a felmérésnek, hogy a dolgozat megírása után a kérdőív azon kérdéseinél, ahol a jelentkező a saját elégedettségéről nyilatkozik, a kevésbé elégedett irányba torzíthatja az eredményt. Ugyanakkor a csalódott, kisebb sikerélményt elért jelentkezők nagyobb valószínűséggel írnak negatívumokat a szöveges értékelésbe, ezáltal jobban kihangsúlyozzák a kurzus gyengeségeit, ami a további munkák és változtatások szempontjából hasznos lehet.

A házi feladatok fakultatív jellege miatt várható volt, hogy azokat csak kevesen oldják meg. Mindössze az érdekes feladatkiírások miatt lehetett számítani néhány megoldásra és arra, hogy a diákok otthon, szabadidejükben is foglalkoznak vele valamennyit.

Összesen két házi feladat érkezett be az utolsó hét végéig, mindkettő esetben 80%-os elért eredménnyel. Ezzel szemben a társaság fele foglalkozott a házi feladatokkal. A kis beadási arány valóban a komplex feladatoknak és a rendelkezésre álló szűkös időnek volt köszönhető.

A hallgatók a dolgozatokon átlagosan 53%-ot értek el, az eredmények szórása viszont rendkívül nagy, 27%-os lett. Ez azt jelenti, hogy nagyon szélsőséges eredmények

születtek. Látszódtott a dolgozatok javítása közben, hogy ki mennyit foglalkozott a témakörrel otthon, valamint hogy ki milyen korábbi tapasztalatokkal érkezett a próbakurzusra. A statisztikából kiderül, hogy a legnagyobb problémát a többségnek a rendelkezésre álló egy óra és az ehhez képest sok feladat aránytalan eloszlása jelentette. Ha a diákok dolgozat miatti csalódottsága befolyásolta is valamilyen szinten a válaszokat, még így is egyértelműen arra lehet következtetni, hogy szinte mindenki sikeresnek ítélte a kurzust és a rajta való részvételt, ami a felmérés egy rendkívül pozitív eredményének tekinthető.

## **5.5 Motiváció, jelenlét**

A próbakurzus során fontos szempont volt, hogy a jelentkezők lelkesedése ne lankadjon, és hogy legalább annyira motiválja a jelentkezőket a próbakurzus előadás sorozata, mint amennyire a kezdő kurzus modelljétől is elvárható. A négy alkalmon lévő személyes jelenlét is emiatt volt „kötelező”. Személyesen sokkal nagyobb hatást lehet gyakorolni a diákokra, mint interneten keresztül. A jelentkezéskor tehát egy internetes kérdőív kitöltésével teljesen objektív módon voltak képesek eldönteni, hogy szeretnének-e részt venni a felmérésben és ízelítőt kapni a játékfejlesztésből. Ekkor minden jelentkezőnek volt valamilyen személyes oka és elvárása, ami miatt szándékukban állt megjelenni az előadásokon. Az első előadásokon tehát nagy volt a jelenlévők létszáma. Ilyenkor az előadó legfontosabb feladata a diákok meggyőzése és érdeklődésüknek fenntartása volt. Ezt kismértékben háttérbe szorította a tény, hogy a három kétórás előadásba nagyon nagy mennyiségű tananyagnak kellett beleférnie.

A többi előadáson már kevesebben vettek részt, átlagosan a jelentkezők fele. Ennek több oka is lehetett, általában személyes jellegű. Ha valakinek mégsem sikerült fenntartani az érdeklődését, mert másra számított, mint amit kapott, vagy pont a hosszú és tömör előadás vette el a kedvét, akkor várhatóan szintén nem jelent meg a többi előadáson. Ezen kívül fontos befolyásoló tényező volt az is, hogy sok jelentkezőnek egyéb elfoglaltsága volt (főleg iskolai és munkahelyi), amelyeket fontosabbnak tartottak a felmérésnél, ezért hagyták ki a személyes előadásokat. Ugyanakkor a társaság fele minden előadáson részt vett és megírta a dolgozatot egyéb elfoglaltságaik ellenére is, ami egy igen ígéretes eredmény. Ez azt jelenti, hogy a hallgatók fele elég érdekesnek és

motiválónak tartotta az előadásokat és a leadott anyagot, hogy egy hónapon keresztül foglalkozzon velük.

Mivel közvetlen a dolgozat megírása után került sor a második kérdőív kitöltésre, ezért ez csak azoknak a jelentkezőknek a véleményét mutatja, akik jelen voltak az utolsó előadáson. Készült egy kérdőív azok számára is, akik nem tudtak megjelenni a dolgozat időpontján valamilyen oknál fogva, viszont – valószínűleg motiváció hiányában – ezt már jóval kevesebben töltötték ki. Főleg azok írták le véleményüket, akiket érdekel a téma és szerettek volna a dolgozatírás időpontján megjelenni, csak nem volt rá lehetőségük. Sajnos azoknak a diákoknak a véleményét, akik a jelentkezési lapot kitöltötték, de dolgozatíráson nem jelentek meg és a hiányzók számára létrehozott kérdőívet sem töltötték ki, a statisztika egyáltalán nem tartalmazza.

Főleg a szöveges értékeléseket végigolvasva, de a diagramok alapján is kijelenthető, hogy a felmérés résztvevői látnak potenciált ebben az oktatási módszerben. A Game Maker program iránt nem köteleződött mindenki teljesen el, de mint eszköz, megfelel a célnak. A próbakurzus sok kisebb-nagyobb hibáját megemlítették a diákok, viszont a kezdő képzés elméletével és alkalmazásával majdnem mindenki egyetért. Össze lehet hasonlítani azt is, hogy három alkalom alatt mennyi tudást és önbizalmat, továbbá kreativitást szereztek a hallgatók a témával kapcsolatban.

Az érdekessége ennek, hogy a legnagyobb javulási tendenciát a programozói elméletekre vonatkozó kérdés mutatja. Az oktatási módszer a gyakorlati programozást és a tapasztalatszerzést helyezi előtérbe és ezzel kezdi az oktatást, ezért az elméleti részeket már képes hozzákapcsolni a diák a megszerzett gyakorlati ismeretekhez. Ilyen módon a programozás elméleti tanítása nagyságrendekkel hatékonyabban működik, mint fordított sorrendben.

Továbbá a programozás játékfejlesztésbe történő illesztése kiválóan alkalmas a diákok kreativitásának és önbizalmának növelésére is. Látható a statisztikák alapján, hogy mindössze négy alkalom után nemcsak többen szeretnék megvalósítani saját játékötletüket, de képesnek is érzik magukat erre. Ez szintén egy nagy előrelépés a jelentkezéskor felmért állapothoz képest.

## 5.6 Összefoglaló értékelés

A felmérés eredménye összességében rendkívül eredményesnek ítéltető. Szerencsére a próbakurzus alapvető hibáitól és korlátaiktól eltekintve is a legtöbb hallgató pozitívan értékelte a felmérést. Szinte minden jelentkező lát ebben az oktatási módszerben előrelépési lehetőségeket és négy alkalom elég volt arra, hogy teljesen meg is győzze őket. A diákokkal történő személyes beszélgetések alapján valóban sokan tapasztalták a hagyományos módszer hátrányait és akadályait, ez az oktatási módszer viszont egy új nézőpontot ad nekik és a többi kezdő programozónak, hogy az akár éveket is igénybe vevő tanulási folyamatot fél évre lerövidítse.



**14. ábra: Egy jelentkező által készített játék**

*Forrás: saját készítés*

A felmérés eredményének érdekessége, hogy a programozáshoz kapcsolódó általános sztereotípiáktól eltérően a jelentkezők 43%-a, vagyis közel fele volt nő, valamint az egy órás dolgozaton a legjobb eredményt is nő érte el. Ennek oka az lehet, hogy a grafikus felületen történő tanulás a szöveggel ellentétben szélesebb körben fölkelte a diákok érdeklődését. Ugyanezt a változatosságot lehet látni a jelentkezők szakmáinak felsorolásából is. Ez azt támasztja alá, hogy a számítógépek jelenleg betöltött nélkülözhetetlen szerepe miatt érdeklődési körtől függetlenül mindenkinek van igénye a programozás tanulására.

A mellékletekben található eredmény 35 jelentkező véleményét foglalja magába a próbakurzusok kapcsán. A jelentkezési lapot ugyan negyvenen töltötték ki, viszont ebből a 40 főből öten nem jelöltek be egyértelmű időpontot az előadásokon történő részvételre, ezért az általuk megadott adatokat nem tartalmazzák a kiértékelő lapok. A jelenléti ívek alapján a három kurzus összes óráján 59%-os részvételi arány figyelhető meg. A hiányzók számára készített kérdőív próbálja felmérni ennek az aránynak az okát. A visszajelzések alapján a hiányzások fő oka a felmérés következmények nélkül történő kiírása volt, a legtöbben valamilyen fontosabb tevékenység miatt hagyták ki az előadásokat.

|                                                                  |      |
|------------------------------------------------------------------|------|
| <b>Összes jelentkező száma:</b>                                  | 40   |
| <b>Jelentkezők, akik bejelöltek 1 időpontot a 3-ból:</b>         | 35   |
| <b>Jelentkezők, akik eljöttek legalább 1 előadásra:</b>          | 29   |
| <b>Jelentkezők, akik az első három előadáson megjelentek:</b>    | 11   |
| <b>Dolgozatot írtak:</b>                                         | 18   |
| <b>Beadott házi feladatok száma:</b>                             | 2    |
| <b>A 4 előadáson résztvevők aránya a 35 főhöz képest:</b>        | 59%  |
| <b>1. és 2. előadás közötti átlagos otthoni munkaórák száma:</b> | 1,19 |
| <b>2. és 3. előadás közötti átlagos otthoni munkaórák száma:</b> | 3,3  |
| <b>A dolgozatra készülések átlagos óraszám:</b>                  | 3,11 |
| <b>Házi feladatok eredményének átlaga:</b>                       | 80%  |
| <b>ZH eredmények átlaga:</b>                                     | 53%  |
| <b>Legjobb ZH eredmény:</b>                                      | 102% |
| <b>ZH eredmények szórása:</b>                                    | 27%  |

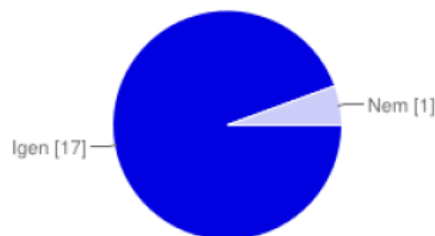
### 15. ábra: Dolgozat eredmények melléklete

*Forrás: saját készítés*

Az átlagos felkészülési óraszámokon jól látható, hogy az első előadás után még kevésbé, viszont a programozás részleteibe történő betekintés után már több energiát befektetve foglalkoztak a hallgatók az otthoni feladatokkal és a jegyzetek tanulásával. Az első előadás nagyobb részvételi létszáma nem befolyásolja ezt az adatot, mivel a munkaórák a második előadástól számítva, a jelenléti ívvel párhuzamosan kerültek felmérésre.

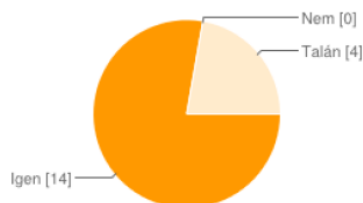
A próbakurzusból levonható következtetéseknek csak egy része származik a számszerű statisztikákból. Egy feleletválasztós kérdőívben könnyen előfordulhat, hogy a kérdőív írója és a válaszadó különböző módon értelmezi az egyes kérdéseket, valamint hogy a diákok különböző értékrend alapján máshogy minősítik az elért eredményeket, ezért jelentős szerepe van a szöveges értékeléseknek is. A mellékletekben olvasható szöveges értékelések az egyes hallgatók szubjektív véleményét tartalmazzák véletlenszerű sorrendben. Az ilyen jellegű kérdések közül főleg a próbakurzus előnyeire és hibáira utaló válaszok adták meg a felmérés összesített eredményét. A válaszok tartalmából és hangvételéből is egyaránt látszik, hogy a dolgozatírók jelentős része pozitív véleménnyel zárta a próbakurzust.

**Megérte jelentkezni a kurzusra?**



|      |    |     |
|------|----|-----|
| Igen | 17 | 94% |
| Nem  | 1  | 6%  |

**Szerinted érdemes lenne ennek az oktatási módszernek a kidolgozását folytatni, az eljárásokat finomítani?**



|       |    |     |
|-------|----|-----|
| Igen  | 14 | 78% |
| Nem   | 0  | 0%  |
| Talán | 4  | 22% |

**16. ábra: Dolgozat utáni kérdőív részlete**

*Forrás: saját készítés*

## **6. További munkák**

A felmérés – habár sok ponton eltér a kezdő kurzus modelljétől – megmutatta, hogy érdemes tovább foglalkozni a témával. Azonban ahhoz, hogy az oktatási módszert hosszú távon is alkalmazni lehessen szükséges újabb statisztikai adatokat gyűjteni olyan felmérések eredményéből, amelyek sokkal kevésbé térnek el az eredeti modelltől.

### **6.1 Újabb felmérés indítása**

A próbakurzusok sok hibára rámutattak, amiken érdemes változtatni. Az egyik legnagyobb probléma a felmérés rövid ideje volt. Három alkalom alatt ugyanis nagyon nehéz olyan előadásokat tartani, amelyek nem túl gyorsak, érthetőek és nem ijesztik el a jelentkezőket, ugyanakkor mégis kellően tartalmasak ahhoz, hogy az oktatási módszer lényegi részét fel lehessen vele mérni. Éppen ezért a következő felmérésben az egyik legnagyobb változtatás az időtartam meghosszabbítása lesz. A kezdő kurzus modelljének teszteléséhez szükséges a 12 alkalom, az új felmérés is ezt tűzi ki célul.

Ezen kívül másik nagy problémája volt a felmérésnek, hogy a jelentkezők önként vállalkoztak a felmérésre. Így az egyetlen dolog, amivel munkára lehetett őket bírni, az az érdeklődésük fenntartása volt. Amennyiben valamilyen egyéb érdek is fűzné a diákokat a tanulásra (vagyis lenne következménye a tárgyon történő megbukásnak), akkor azoknak a véleményét is fel lehetne mérni, akiket kevésbé fogott meg az előadás anyaga. Továbbá a diákokat így otthoni munkára is lehetne fogni, a számonkérést házi feladatok formájában is el lehetne végezni. Az érdek megteremtéséhez valamilyen módon jutalmazni kell a jól tanuló diákokat. Ez többféleképpen is történhet, de talán az egyik leghatásosabb az eredmények beszámítása a tanulmányokba. Felsőoktatási intézményekben kézenfekvő a kreditrendszer kihasználása, hiszen amennyiben a kurzust szabadon választható tárgyként végzik el a diákok, ez a probléma máris megoldott. Középiskolák esetében a tanrend jóval kötöttebb, ezért ott csak szakkörök formájában lehet véghezvinni a felmérést. Ahhoz, hogy a diákok érdekeltek legyenek a szakkörökön való jó részvételben, lehetőséget kell nekik adni, hogy a számítástechnika órán kapott érdemjegybe beleszámítson a szakkörön elért teljesítmény is. Viszont ez a

módszer csak azokban a középiskolákban nem borítaná föl a tanmenetet, ahol a szakkör elvárásai magasabb szinten vannak, mint a normál tanórai követelményei.

Amennyiben a fél éves időtartam megvalósítható és a diákok is érdekeltté tehetők, a kezdő kurzus modelljét gyakorlatilag egy az egyben ki lehet próbálni és egy nagyon jó közelítéssel bíró felmérést lefolytatni. Mivel a felmérésnek következményei is lesznek a diákok eredményeire nézve, javasolt először csak a felsőoktatási intézmények kreditrendszerébe beilleszteni ezt, mivel egy esetleges sikertelenség itt jóval kisebb következményekkel jár, mint a középiskolákban. Ahhoz, hogy ez a gyakorlatban is megvalósítható lehessen, szükség van egy részletes terv elkészítésére az előadásokat, az oktatókat, a célközönséget, a helyszínt és a felmérés módját illetően. Ez az egy szemeszter hosszú felmérés már a 2013/14/2 félévben kezdetét veheti, hogy 2014 nyár elejére rendelkezésre álljanak az eredmények. Továbbá szükség lehet ismét több kurzus párhuzamosan történő indítására is, hogy a végső statisztika minél tágabb célterületet fedjen le és pontosabb következtetéseket lehessen levonni a nagyobb jelentkezési szám alapján.

## **6.2 Távolabbi célok**

A Game Maker program oktatása az Amerikai Egyesült Államok területén és Hollandiában a legnépszerűbb. Amennyiben a felmérések sikeresnek bizonyulnak és az itt leírt oktatási módszerrel valóban alkalmassá válik a program a programozás hatékonyabb oktatására, akár országszerte lerövidítheti a kezdő programozók tanításának idejét, javítva a magyar oktatás minőségét. Mivel a Game Maker angol nyelvet használ, nem okoz problémát idegen nyelvű tanítása sem.

Ahhoz, hogy ezeket az eredményeket el lehessen érni, természetesen a pedagógiai módszer további felméréseire és a GML programnyelv oktatók közötti népszerűsítésére is szükség van.

## 7. Összefoglalás

A mai digitális világban különös figyelmet kell fordítani a számítógép- és médiakezelés, valamint a programozás szerepére és használatának oktatására.

A legtöbb magyar középiskolai és felsőoktatási képzésben közel azonos, hagyományos pedagógiai elvek és algoritmusok szerint történik a tanítás. Ez a dolgozat egy új módszer kidolgozásáról és alkalmazásáról szól, ahol a diákok legelső programozói feladataikat nem egy hagyományos szöveges, hanem egy speciális, grafikus és szöveges felületet is támogató program segítségével írják meg. Az oktatási módszerben fontos szerepet kap a számítógépes játékfejlesztés, mint motiváló eszköz, amely a programozás alkalmazhatóságát és gyakorlásának hatékonyságát képes nagymértékben növelni.



**17. ábra: Egy GML-ben írt játék**

*Forrás: saját készítés*

A téma keretein belül kialakuló projekt kidolgoz két féléves kurzust, amely a speciális programozói felület használatának tanításáról szól. A módszer elméletének és a kurzusok sikerességének alátámasztásához egy próbakurzus keretében zajlott egy felmérés, amelyre önkéntesek jelentkeztek. Ennek a tesztnek az eredménye megmutatja, hogy az elmélet valóban hatékonynak bizonyul, hiszen a próbakurzus három előadása alatt a tanulásban érezhető javulás és a hallgatók motivációjának jelentős növekedése figyelhető meg. Több különböző kérdőív kitöltéséből összeállított statisztika alapján

levonható a következtetés, miszerint érdemes tovább foglalkozni ezzel a módszerrel és egy új felmérést indítani, amely a képzési módról egy valósabb képet ad.

A gyakorlatorientált oktatási módszer kidolgozásának folytatása igen nagy jelentőséggel bírhat. A programozás tanulásának hatékonysága jelentősen megnövekedhet és az oktatás ideje akár felére csökkenhet. Ezáltal a diákok nem csupán nagyobb kompetenciával rendelkeznének, hanem magabiztosabbak és elszántabbak lennének, kreativitásuk fokozása a kutató-fejlesztői munkák során és az iparban egyaránt eredményesebbé tenné őket. Ez a módszer rávilágít arra is, hogy a továbbiakban a programozás önálló tanulásához milyen eszközöket érdemes a diákoknak használni, illetve hogyan tudják a megszerzett tudásukat munkájuk során alkalmazni.

A módszer további nagy előnye, hogy független az előzetes programozói tudástól és korosztálytól is, nem tesz különbséget alkalmazás szempontjából a középiskolai, egyetemi vagy felnőttoktatási intézmények között. Teljesen kezdő és tapasztaltabb programozók is alkalmazhatják, a hagyományos oktatási módszerrel tanított programnyelvek ezzel kiegészíthetőek. Tehát ez egy alternatív módszer, amelynek segítségével nagyságrendekkel jobbminőségű oktatás valósítható meg.

## Irodalomjegyzék

- Anderson, E. F., McLoughlin, L. (2007): *Critters in the Classroom: A 3D Computer Game-Like Tool for Teaching Programming to Computer Animation Students*, The National Centre for Computer Animation, Bournemouth University
- Audacity (2013): <http://audacity.sourceforge.net/>, 2013. október 27.
- Benedek, A. (2007): *Mobiltanulás és az egész életen át megszerezhető tudás*. In: Világosság, 2007/9. pp. 21-28.
- Bleed, R. (2005): *Visual Literacy in Higher Education*, Maricopa Community Colleges, ELI Explorations, August 2005
- Brain, M. (2009): *Teaching your kids how to write computer programs*, <http://marshallbrain.com/kids-programming.htm>, 2013. október 27.
- Chen, W. K., Cheng, Y. C. (2007): *Teaching Object-Oriented Programming Laboratory With Computer Game Programming*, IEEE Transactions on Education, Vol. 50, No. 3, August 2007
- Falus, I. (2004): *Bevezetés a pedagógiai kutatások módszereibe*, összefoglaló: [http://pihgy.hu/files/Falus\\_Bev\\_ped\\_kutat%C3%A1s\\_m%C3%B3dszereibe.pdf](http://pihgy.hu/files/Falus_Bev_ped_kutat%C3%A1s_m%C3%B3dszereibe.pdf), 2013. október 28.
- Finley, K. (2013): *World of Codecraft: 3-D Game Teaches Kids 'Big Ideas' of Programming*, <http://www.wired.com/wiredenterprise/2013/07/programming-game-engagement/>, 2013. október 27.
- Gamasutra (2013): *Games That Teach Programming: A Brief Overview*, [http://www.gamasutra.com/blogs/RobLockhart/20130905/199667/Games\\_That\\_Teach\\_Programming\\_A\\_Brief\\_Overview.php](http://www.gamasutra.com/blogs/RobLockhart/20130905/199667/Games_That_Teach_Programming_A_Brief_Overview.php), 2013. október 27.
- Haden, P. (2006): *The Incredible Rainbow Spitting Chicken: Teaching Traditional Programming Skills Through Games Programming*, Eighth Australian Computing Education Conference (ACE2006), Conferences in Research in Practice in Information Technology, Vol. 52.
- Heussner, K. M. (2013): *Robots and rocket launchers: Kuato Studios uses gaming to teach kids the basics of coding*, <http://gigaom.com/2013/03/26/robots-and-rocket-launchers-kuato-studios-uses-gaming-to-teach-kids-the-basics-of-coding/>, 2013. október 27.

- Kata, J. (2006): *Szakképzés-pedagógia* (Szerk. Benedek András) Budapesti Műszaki és Gazdaságtudományi Egyetem Gazdaság- és Társadalomtudományi Kar Alkalmazott Pedagógia és Pszichológia Intézet, Typotex, 2006., 49-62. oldal
- Katona, K., Kurdi, Zs. Zs. (2008): *A Szoftvertervezés Gyakorlati Oktatása*, Informatika a felsőoktatásban 2008, Debrecen
- Kirkland, D., O’Riordan, F. (2008): *Games as an Engaging Teaching and Learning Technique: Learning or playing?*, Griffith College Dublin
- Kurzweil (2013): *A video game that teaches how to program in Java*, <http://www.kurzweilai.net/a-video-game-that-teaches-how-to-program-in-java>, 2013. október 27.
- Leutenegger, S., Edgington, J. (2004): *A Games First Approach to Teaching Introductory Programming*, Conference’04, Month 1–2, 2004
- Mikoluk, K. (2013): *Best Programming Languages to Learn in 2013: The Elite Eight*, <https://www.udemy.com/blog/best-programming-language/>, 2013. október 27.
- Minden, ami Közgáz (2010): *Oktatás + módszertan = oktatásmódszertan*, [http://kozgaz.blog.hu/2010/06/17/oktatas\\_modszertan\\_oktatasmodszertan](http://kozgaz.blog.hu/2010/06/17/oktatas_modszertan_oktatasmodszertan), 2013. október 27.
- Miskolczi, I. (2010): *Tanulás vagy E-tanulás? – Oktatásmódszertan a XXI. században*, Szolnoki Tudományos Közlemények XIV., Szolnok, 2010
- Moore, D. S. (1997): *New Pedagogy and New Content: The Case of Statistics*, International Statistical Review (1997), 65, 2, 123-165
- Nyíri, K. (2003): *From Texts to Pictures: The New Unity of Science*, Works online
- Overmars, M. (2004): *Teaching Computer Science through Game Design*, Georgia Tech Research Institute
- Pap-Szigeti, R., Pásztor, A. (2008): *A LEGO Programozható Robotjaival Segített Programozásoktatás Bevérlésvizsgálata*, Informatika a felsőoktatásban 2008, Debrecen
- Pinola, M. (2013): *How and Why to Teach Your Kids to Code*, <http://lifehacker.com/how-and-why-to-teach-your-kids-to-code-510588878>, 2013. október 27.
- Primerlabs (2013): <http://primerlabs.com/>, 2013. október 27.
- Rajaravivarma, R. (2005): *A games-based approach for teaching the introductory programming course*, Newsletter ACM SIGCSE Bulletin, Volume 37 Issue 4, December 2005 Pages 98-102

- Romano, B. (2013): *Dan Shapiro Builds a Board Game to Teach 3-Year-Olds Programming*, <http://www.xconomy.com/seattle/2013/09/03/dan-shapiro-builds-board-game-teach-3-year-olds-programming/>, 2013. október 27.
- Scorpion games (2013): <http://scorpeti.jatekoldal.net/main.php?p=1>, 2013. október 27.
- Siemens, G. (2005): *Connectivism: A Learning Theory for the Digital Age*, [http://www.itdl.org/Journal/Jan\\_05/article01.htm](http://www.itdl.org/Journal/Jan_05/article01.htm), 2013. október 27.
- Siemens, G. (2006): *Knowing Knowledge*, ISBN 978-1-4303-0230-8
- Teaching Kids to Program (2013): <http://www.teachkidstoprogram.com/>, <http://www.teachkidstoprogram.com/teaching-a-class-to-program>, 2013. október 27.
- Terrón, A. M. (2003): *Historical origins of statistics applied to pedagogy*, Revista Electrónica de Metodología Aplicada 2003, Vol. 8 n° 1, pp. 1-23
- Tóth, Gy. F., Johanyák, Zs. Cs. (2007): *A Szoftverfejlesztés Reformja a Versenyszféra Visszajelzései Alapján*, V. Pedagógiai Értékelési Konferencia, 2007
- Visual Education (2013): <http://centerforvisualeducation.com/about-vised/new-blog-entry/>, 2013. október 27.
- Windows Movie Maker (2012): <http://windows.microsoft.com/hu-hu/windows/get-movie-maker-download>, 2013. október 27.

## **Mellékletek, függelék**

1. Próbakurzus statisztikája – jelentkezési kérdőív (5 oldal)
2. Próbakurzus statisztikája – dolgozat eredmények (1 oldal)
3. Próbakurzus statisztikája – dolgozat utáni kérdőív (27 oldal)
4. Próbakurzus statisztikája – hiányzók kérdőíve (5 oldal)