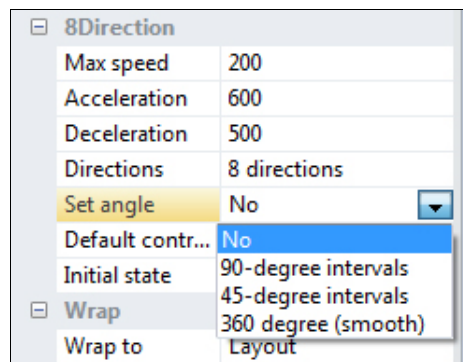
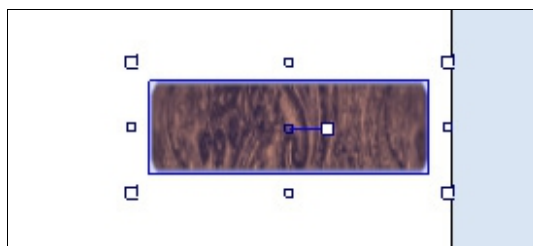


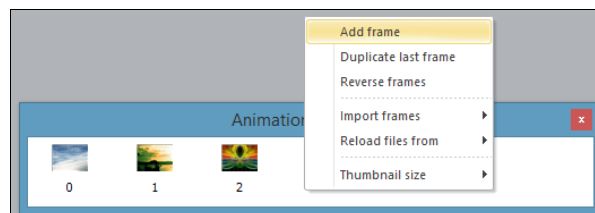
1. Melléklet: Képernyőképek



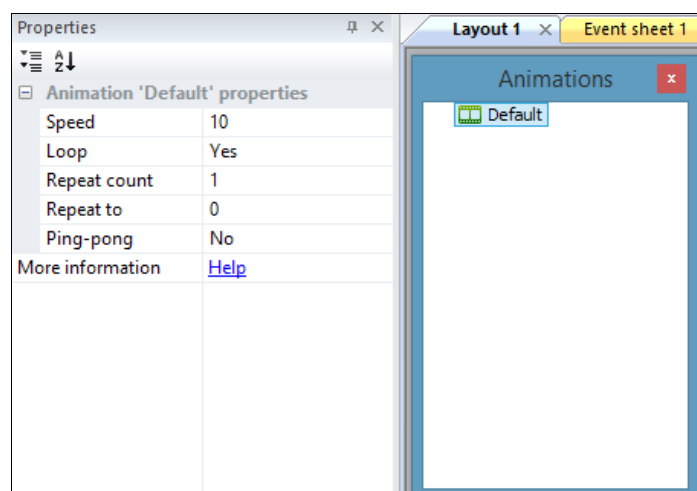
14. ábra, A 8Direction viselkedés beállításai.



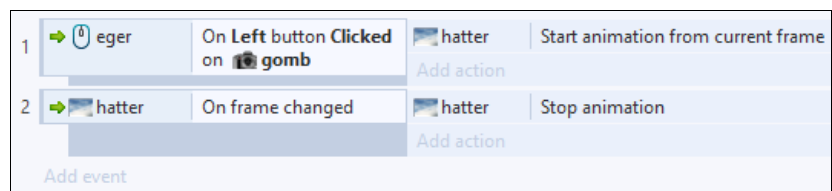
15. ábra, Sprite típusú objektum átméretezése.



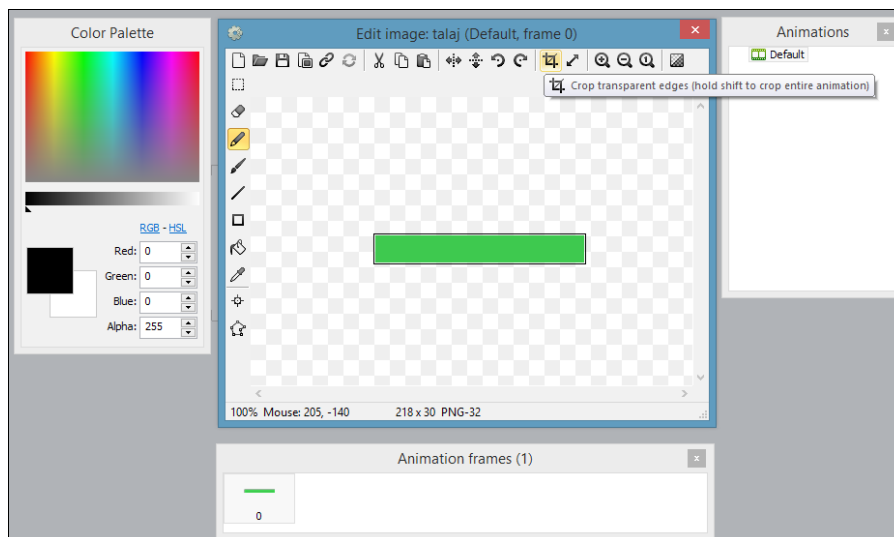
16. ábra, Az animáció képkockáit kezelő felület.



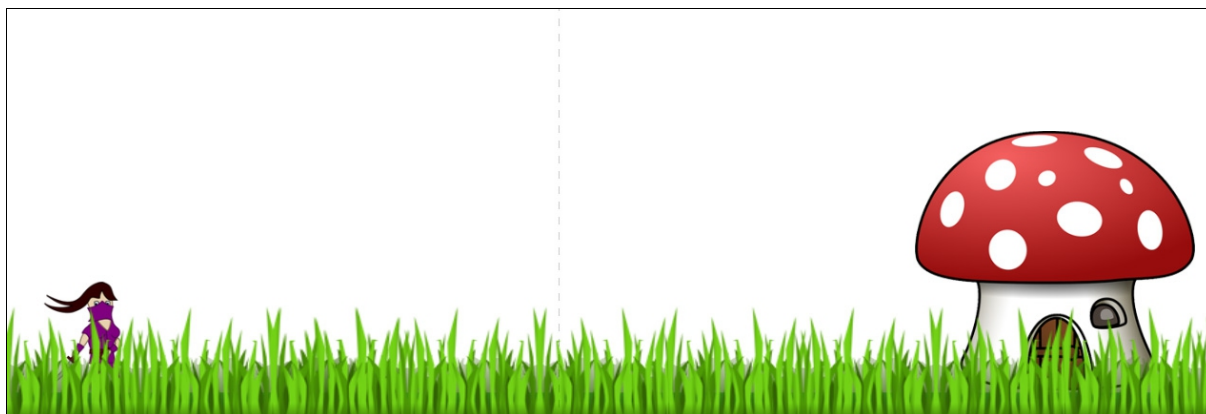
17. ábra, Az animáció beállításait kezelő felület.



18. ábra, A Tájépek feladat eseménylapja.



19. ábra, A zöld téglalap megrajzolása.



20. ábra, A Hazaút feladat elrendezésének felépítése.

Dialog box titled "Edit global variable".

Fields:

- Name:
- Type:
- Initial value:
- Description (optional):
- ☒ Static
- ☐ Constant

Buttons: [Help](#), ,

21. ábra, Globális változó deklarálása.

Global number **szam** = 0

1	kenyer	On DragDrop drop	System	Add 1 to szam
	kenyer	Is overlapping	eredmeny	Set text to szam
			kenyer	Destroy

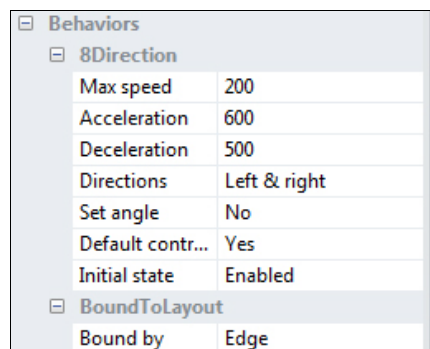
Buttons: [Add event](#), [Add action](#)

22. ábra, A Kenyerek a ládába feladat eseménylapja.

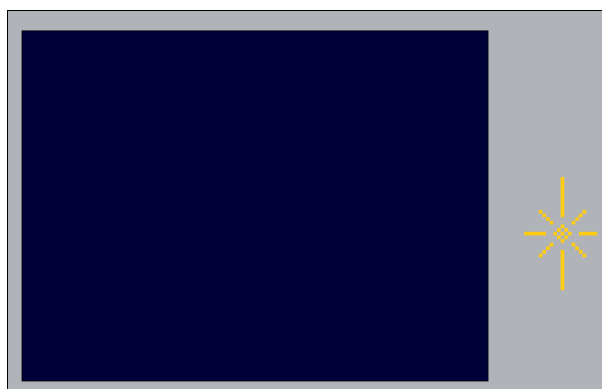
Physics settings:

Immovable	Yes
Collision mask	No
Prevent rotation	Yes
Density	1
Friction	0.5
Elasticity	0.2
Linear damping	0
Angular damping	0.01
Bullet	No
Initial state	Enabled

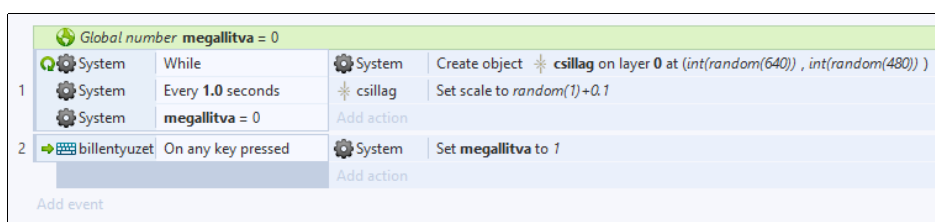
23. ábra, A Physics viselkedés beállításai.



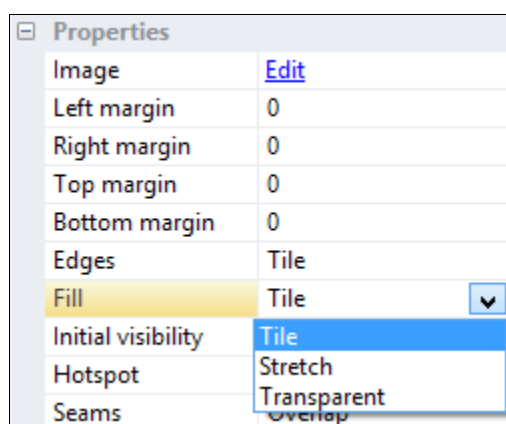
24. ábra, A 8 Direction viselkedés beállításai.



25. ábra, A csillag objektum elhelyezkedése szerkesztői nézetben.



26. ábra, A Csillagos éjszaka feladat eseménylapja



27. ábra, A 9-patch objektum beállításai.

1	  eger	On Left button Clicked	 fiu	Find path to (eger.X, eger.Y)
	 eger	 Cursor is over  fal	Add action	
2	  fiu	On  Pathfinding path found	 fiu	Move along path
			Add action	
Add event				

28. ábra, A Labirintus feladat eseménylapja

Positioned audio	
Panning model	HRTF
Distance model	Linear
Listener Z height	600
Reference distan...	100
Maximum distan...	1300

29. ábra, A „hangok” objektum beállításai.

1	 System	On start of layout	 radio	Set position to (<i>LayoutWidth/2, LayoutHeight/2</i>)
			 hangok	Play zene looping at volume 0 dB at object  radio (inner angle 360, outer angle 360, outer gain 0 dB) with tag
			 hangok	Set listener object to  gyerek
			 gyerek	Set position to (<i>int(random(1466)+34) , int(random(1448)+51) </i>)
			Add action	
2	 System	While	 gyerek	Set position to (<i>int(random(1466)+34) , int(random(1448)+51) </i>)
	 gyerek	Is overlapping  radio	Add action	
	Add event			

30. ábra, A Valahol a sötétben feladat eseménylapja a véletlenszerű pozicionálás lekezelésével

1	→ billentyuzet	On Space pressed	Add action	
	turista	Is overlapping ember_magyar at offset (50, 0)	System	Set nyelv to "magyar"
2	turista	Is overlapping ember_magyar	fuggveny	Call "koszones" ()
		- or -	Add action	
3	turista	Is overlapping ember_angol	System	Set nyelv to "angol"
	turista	Is overlapping ember_angol at offset (50, 0)	fuggveny	Call "koszones" ()
4	turista	Is overlapping ember_francia at offset (50, 0)	Add action	
	turista	Is overlapping ember_francia	System	Set nyelv to "francia"
Add action				

31. ábra, A turista és az emberek átfedésének vizsgálata az eseménylapon.

5	fuggveny	On "koszones"	szoveg	Destroy
			System	Set melyik_koszones to <code>int(random(3))</code>
			Add action	
6	System	nyelv = "magyar"	System	Create object szoveg on layer 0 at (<code>ember_magyar.X</code> , <code>ember_magyar.Y-70</code>)
			Add action	
7	System	melyik_koszones = 0	szoveg	Set text to "Jó napot kívánok!"
			hangok	Play koszones_magyar_1 not looping at volume 0 dB (tag "")
			Add action	
8	System	melyik_koszones = 1	szoveg	Set text to "Isten hozta!"
			hangok	Play koszones_magyar_2 not looping at volume 0 dB (tag "")
			Add action	
9	System	melyik_koszones = 2	szoveg	Set text to "Üdvözlöm!"
			hangok	Play koszones_magyar_3 not looping at volume 0 dB (tag "")
			Add action	
10	System	nyelv = "angol"	System	Create object szoveg on layer 0 at (<code>ember_angol.X</code> , <code>ember_angol.Y-70</code>)
			Add action	
14	System	nyelv = "francia"	System	Create object szoveg on layer 0 at (<code>ember_francia.X</code> , <code>ember_francia.Y-70</code>)
			Add action	
			Add event	

32. ábra, A magyar köszönések kezelésének kódja az eseménylapon.

2. Melléklet: A Construct 2 ismertetése

A program leírása

A Construct 2 egy vizuális programozást alkalmazó, egyszerűen kezelhető alkalmazásfejlesztő környezet. Jelenleg csak Windows operációs rendszerre elérhető (a cég korlátozottságai miatt), azonban tervezve van egy átírás MAC és Linux rendszerekre is. A szoftver HTML5 alapú motorra építi az elkészült projekteket. Ebből adódóan az elsődleges célplatform a Web (habár nem korlátozódik csupán erre a területre). XML szabványt alkalmazva tárolja a projektek adatfájljait, így azok a program nélkül is szerkeszthetők és módosíthatók hozzáértő felhasználók számára (habár lefordításukhoz természetesen szükség van a Construct 2 környezetre).

Licenszek

A program ingyenesen letölthető néhány korlátozással. Az ingyenes változat szabadon terjeszthető (pénzért nem árusítható), oktatásban alkalmazható, viszont a vele készített termékek kereskedelmi forgalomba nem hozhatók. Ez a licenz emellett még néhány funkcióbeli korlátozást is felsorakoztat: csökkentett lehetőségek célplatformok esetén exportáláskor (mobil, valamint asztali alkalmazások nem exportálhatók); limitált eseményszám a programkódban; korlátozott réteghasználat az elrendezéseken; korlátozott effekthasználat; csak a saját számítógépen tesztelhető a projekt (bővebben az *Előnézet, tesztelés* alfejezetben); valamint néhány kényelmi funkció hiánya (például almappák az objektumok rendezéséhez).

Mindezek ellenére a program ingyenes változata is tökéletesen alkalmas az oktatásra és az algoritmikus készség kialakítására, fejlesztésére.

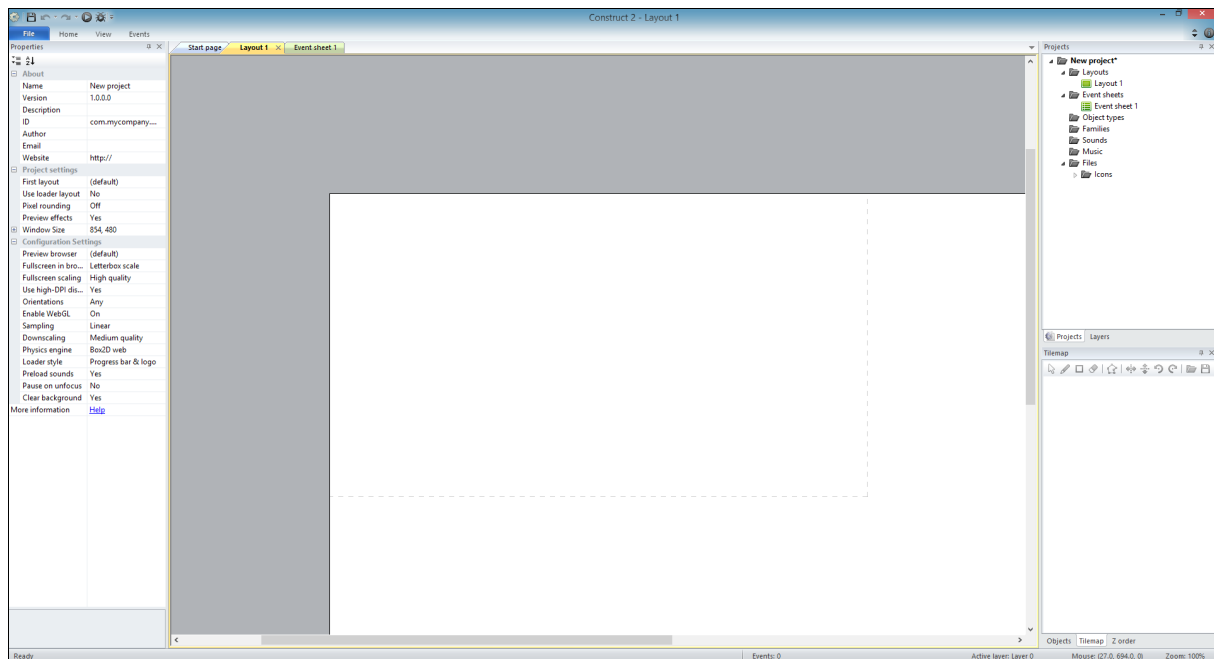
Az ingyenes változatot követi két fizetős és az oktatási licenszek, melyek között csupán a célközönség a különbség, ugyanis mindegyik feloldja az összes korlátozást és a program teljes körű használatát lehetővé teszi. Emellett a kész termékeket is kereskedelmi forgalomba bocsáthatjuk.

- Az első fizetős licenz a magánszemélyeknek szól (jelenleg 130 \$-os áron). Ezt csupán egy személy használhatja alkalmazásfejlesztésre. Hasonlóan bármelyik licenszhez, korlátlan számú alkalommal felhasználható, hogy az adott személy minden számítógépére telepíthető legyen.

- A második fizetős licenz az üzleti réteget célozza meg (jelenleg 430 \$-os áron). A magánszemélyeknek szóló verzióval ellentétben itt már a csapat (vagy vállalat) egésze használatba veheti a teljes funkcionalitását a szoftvernek és korlátlan mennyiségben árusíthatja a kész terméket. Amennyiben egy magánszemély meglépi a szoftverrel készített termékkel keresett 5000 \$-os bevételt, a licenzét fejlesztenie kell üzleti licenszre.
- Az oktatási licenz célja, hogy az oktatási intézmények számára is elérhetővé tegye a program fizetős opcióit elérhető áron. Ez a licenz korlátlan számú számítógépre telepíthető és használható a tanulók és oktatók által egyaránt az intézmény keretein belül. Az intézmény választhat, hogy havi (30 \$) vagy éves (260 \$) bontásban kíván előfizetni a szoftverre amennyiben azt helyben használatra szerzi be. Lehetőség van úgynevezett „Take home” licenszek vételére is, amely feljogosít egy tanulót, hogy otthon is telepíthesse a Construct 2 teljes verzióját. Ezt ugyancsak havi (1,5 \$/tanuló) vagy éves (17 \$/tanuló) előfizetéssel igényelheti az intézmény. A licenszek ezen típusai csak azokat az iskolákat érintik, amelyek igénybe akarják venni a program fizetős opcióit. Az ingyenes verzió továbbra is szabadon felhasználható oktatási kereteken belül is.

Kezelőfelület

A program kezelőfelülete első ránézésre bonyolultnak tűnhet, azonban néhány óra használat után felfedezhető annak logikus és kényelmes elrendezése. Fontos, hogy szem előtt tartsuk a felület értékelésekor, hogy a fejlesztők úgy tervezték meg azt, hogy komolyabb és meglehetősen komplex alkalmazások készítésekor is átlátható legyen a fejlesztési folyamat minden aspektusa.



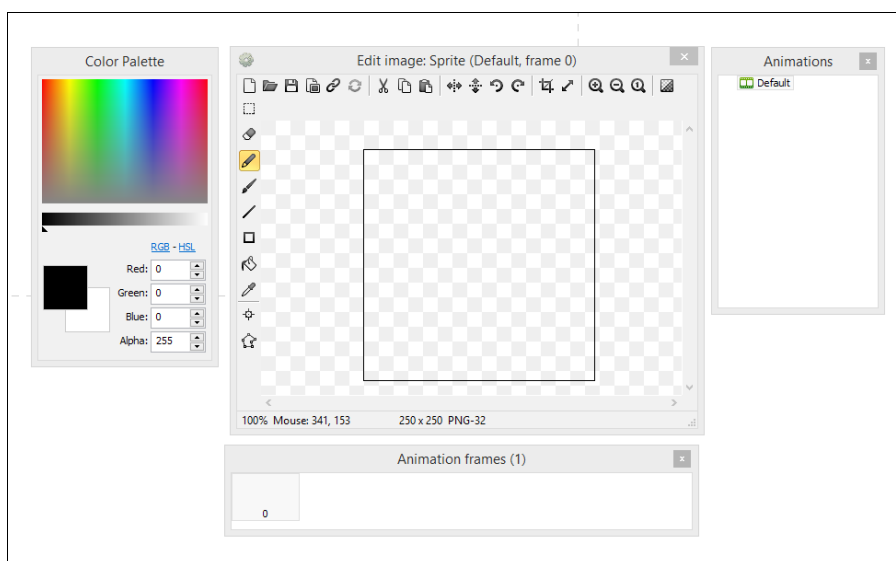
33. ábra, A Construct 2 kezelőfelülete.

A felhasználói felület négy fő területre különíthető el (33. ábra).

- A felső sávban találhatóak a már más szoftverekből megszokott alapvető menüelemek. Itt nyílik lehetőségünk a program és a fejlesztés alatt álló projekt fő kezelésére. Hozzáférhetünk a Construct 2 beállításaihoz, de a projektet is itt menthetjük el, futtathatjuk le, vagy indíthatunk exportálást. Emellett még a kezelőfelületet beállításait is itt találjuk, illetve az eseménylapok egyszerű funkcióit is elérhetjük.
- A bal oldali oszlop tárolja az éppen kijelölt elem vagy objektum beállításait. A projektet kijelölve itt adhatjuk meg az alapvető információkat, mint például a program neve, készítő(k), leírás, verziószám stb. Az egyes elrendezéseket is ezen a felületen tudjuk testreszabni (akárcsak az azokon belül található rétegeket). Összesítve: Amennyiben egy szoftverelem (legyen az pálya, eseménylap, réteg, objektum, vagy bármi egyéb) rendelkezik beállítási lehetőségekkel, azokat ezen a területen érhetjük el az elem kijelölését követően.
- A jobb oldali oszlop elsődleges funkciója a projekten belüli navigáció és az átláthatóság megkönnyítése. Itt kapott helyet az objektumlista, amely az adott pályaelemen elhelyezett objektumokat jeleníti meg. Ezt követi a projektfa. Itt találhatjuk az aktuális munkánkhoz kapcsolódó összes fájlt, elemet, objektumot (ez a felület szolgál elsődlegesen az elemek kiválasztására, illetve az adott elrendezéshez való hozzáadásra). Ugyancsak itt láthatjuk a rétegeken beállított z-sorrend

kezelőpanelt is, illetve a „tilemap” objektum könnyebb kezelését lehetővé tevő fület is. A sor a rétegek fülével zárul, ahol az aktuális elrendezésnek a rétegeit kezelhetjük, illetve állíthatjuk be.

- A középső és egyben legnagyobb felületen építjük a szoftvert, itt zajlik a vizuális programozás és a grafikai kialakítás fő folyamata. Három fő funkcióval rendelkezik:
 - Az első a kezdőlap, amely a program indításakor üdvözlí a felhasználót és egy gyors eligazító oldalként szolgál. Innen gyorsan betölthetőek a korábbi projektek, vagy új munka elkészítéséhez is hozzáfoghatunk (habár ezek a funkciók a felső menüből is elérhetőek). Ezek mellett még helyet kapott néhány hasznos hivatkozás is (például: a szoftver kézikönyve, felhasználói segédletek, közösség stb).
 - A második funkció az úgynevezett „elrendezés szerkesztő”. Itt nyílik lehetőségünk arra, hogy a felhasználó számára megjelenő grafikai felületet elkészítsük, elrendezzük. Más szavakkal ez egy WYSIWYG (What You See Is What You Get) szerkesztő, azaz amit a szerkesztési felületen látunk, az úgy is fog megjelenni a kész projektben.
 - A harmadik fő funkció pedig az eseménylapok szerkesztése (bővebben az *Eseménylapok, vizuális programozás alfejezetben*).
- A program a további kezelőfelületeket felugró ablakokban jeleníti meg. Így mindig az éppen aktuálisan szükséges beállításokat tudjuk elvégezni. Ilyen funkció például az objektumok hozzáadása, azok beállítása, vagy akár a korábban említett eseménylap szerkesztése közben az „építő-/kódelemek” hozzáadása is ilyen módon történik.



34. ábra, A beépített grafikai szerkesztőfelület.

- Külön figyelmet érdemel a Construct 2 beépített grafikai szerkesztője (34. ábra). Ez a felület szintén felugró ablakként jelenik meg, amint egy szerkesztésre alkalmas grafikai objektumnak a képét vagy animációját szeretnénk módosítani. Természetesen nem egy kiforrott grafikai szoftvertől elvárt funkciókra kell számítani, viszont az alapvető rajzolóeszközök megtalálhatóak itt is, hogy gyorsan kijavíthassunk vagy módosíthassunk valamit az adott képen. Emellett az objektumok animációit is ezen a felületen állíthatjuk be.

Az alkalmazások felépítése

A Construct 2 projektjeinek felépítését a projektfán követhetjük, amely a következő kategóriákból tevődik össze: elrendezések, eseménylapok, objektumok, családok, hangok, egyéb fájlok. Ahogy a „Program leírása” alfejezetben említettem, a környezet ezeket az elemeket XML formátumban, valamint az ezekhez tartozó egyéb fájlokban (például képfájlokban) tárolja. Lehetőségünk van kétfajta mentési formátum közül választani:

Az első formátum a projektként mentés. Ebben az esetben a program almappákba szétosztva tárolja el a mentési fájlokat (XML és egyéb fájlokat).

A második formátum az egy fájlként történő mentés. Az előzőhöz képest az eltérés annyiban nyilvánul meg, hogy a program egy ZIP archívumba tömöríti a fájlokat. Természetesen a fejlesztői környezet meg tudja nyitni a tömörített capx kiterjesztést is, habár ilyen mentési formátum mellett számolnunk kell a mentés során fellépő többletidővel is, amely nagyobb projektek esetén zavaró lehet.

A projektek elemeinek leírása:

- Elrendezések (*Layouts*): A felhasználó szeme elé táruló felület. Ezen a „vászon” helyezkednek el az egyes elemei, objektumai a programunknak.
- Eseménylapok (*Event sheets*): A programunk forráskódjait tartalmazó lapok. Akkor kerülnek lefutásra, ha kapcsolódnak valamelyik elrendezéshez, vagy egy már futó eseménylap meghívja őket.
- Objektumok (*Objects*): A „szereplői” programunknak. Objektumok nélkül az alkalmazásunk csupán egy üres felület, amin nem történik semmi egyéb a rendszerműveleteken kívül. Minden elem, amelyet megjeleníteni látunk egyfajta

objektum, melyhez eseménylehetőségek hosszú sorai kapcsolódnak. Érdekesség, hogy a Construct 2 nemcsak a szokásos elemeket (például kép) sorolja az objektumok közé, de az egérrel és billentyűzettel történő vezérlést is itt találjuk.

- Családok (*Families*): Az objektumok egyes csoportjait jelentik. Segítségükkel egyszerűbbé és optimalizáltabbá tehetjük a forráskódot. Például egy azonos csoportba sorolva minden gomb típusú objektumot, egyetlen utasítással megadhatjuk, hogy amikor nem mutatunk rá az egérmutatóval, akkor 50%-os átlátszósággal jelenjenek meg. Hozzáteszem, hogy a családok használata már a fizetős licenszhez tartozik.
- Hangok (*Sounds, Music*): Azok a hangelemek, amelyek lejátszásra kerülnek az eseménylapokon leprogramozott pillanatokban. Fontos szem előtt tartani, hogy a Construct 2 alapvetően egy HTML5 alkalmazásfejlesztő környezet, ezért a hangfájlokat szükséges átkonvertálni olyan formátumba, amely már kompatibilis a böngészőkkel. Erre szolgál a beépített konvertáló felület, ami az új hangok importálásakor jelenik meg. Amennyiben a konvertáló kezeli a kiválasztott bemeneti formátumot, megpróbálja előállítani a szükséges M4A és OGG hangfájlokat.
- Egyéb fájlok (*Files*): A projekthez kapcsolódó egyéb fájlok. Gyakorlati használatban nem túlságosan jelentős kategória (mivel az átlagprojektek nem használnak olyan fájlokat, amelyek kívül esnek a fenti kategóriákon) azonban mindenképp említésre szorul.

Objektumok

Az objektumok a programunk „szereplői”. Ezeket az elemeket jeleníti meg a szoftver az elrendezéseken, ezekkel tud a felhasználó interakcióba lépni. Az eseménylapokon választható utasítások készlete is a hozzáadott objektumok függvénye. A legtöbb objektum lokális elemként funkcionál, avagy csak azokon az elrendezéseken lesz elérhető, ahova beillesztettük, vagy létrehoztuk utasítás segítségével (a már létrehozott objektumot számtalan elrendezésen felhasználhatjuk). Emellett néhány objektum globális tulajdonságokkal rendelkezik így külön hozzáadás nélkül elérhetőek lehetőségei az egész program számára (például: billentyűzet).

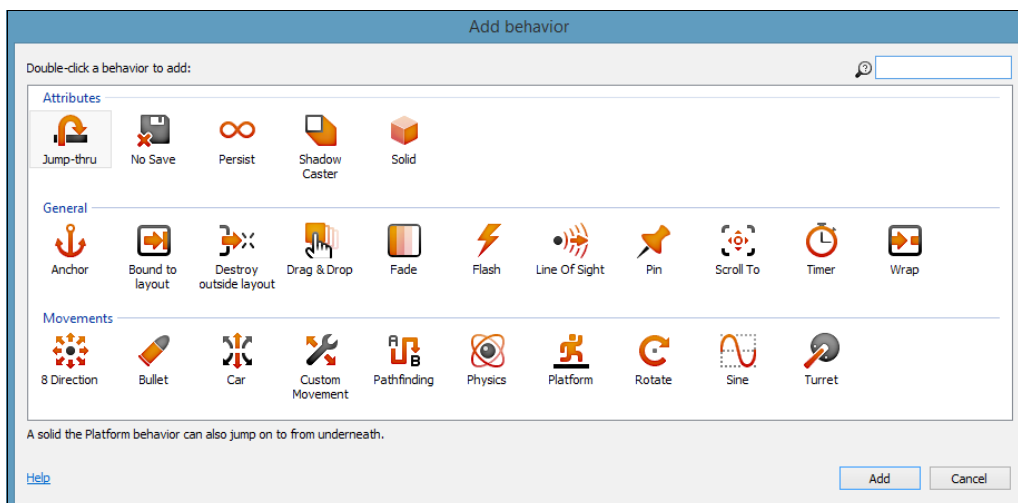
A következő kategóriákba sorolja az objektumokat a Construct 2:

- Adat és tároló objektumok (*Data & Storage*): Ezekkel az objektumokkal főként adattárolást és fájlkezelést valósíthatunk meg. Lehetőségünk van WebStorage használatára, de akár XML támogatást is hozzáadhatunk az alkalmazáshoz.

- Űrlapok kezelése (*Form controls*): Alapvető űrlap elemeket illeszthetünk be. Érdekes, hogy a használatukhoz és vezérlésükhöz nem szükséges hozzáadni külön vezérlőobjektumot.
- Általános (*General*): Itt találhatóak a leggyakrabban alkalmazott objektumok. Ezek szinte minden alkalmazásban fellelhetők. Itt tudunk képet, háttérképet, szöveget és egyéb szokványos elemet beilleszteni.
- Bemenet (*Input*): A program interakcióját biztosító objektumok. Például amennyiben egér bevitelt szeretnénk alkalmazni (más szavakkal: az eseménylapon az egérhez kapcsolódó utasításokhoz hozzáférni) akkor az egér objektumot hozzá kell adni a projekthez. A jelenleg támogatott beviteli perifériák: Gamepad, billentyűzet, egér és érintőképernyő. Minden bemeneti objektum globális elemként viselkedik.
- Média (*Media*): Ebben a kategóriában találhatjuk a hangok vagy videók lejátszásához szükséges elemet, valamint a böngészőprogramok által támogatott felhasználó médiákhoz férhetünk hozzá (például webkamerákhoz).
- Monetizáció (*Monetisation*): Az ebbe a kategóriába tartozó objektumokkal van lehetőségünk különböző népszerű reklámszolgáltatások hirdetéseit az alkalmazásunkba ágyazni, illetve alkalmazáson belüli vásárlások is megvalósíthatóak a legnépszerűbb mobil platformokat támogatva.
- Platform specifikus (*Platform specific*): Amennyiben nem az alapértelmezett HTML5 alkalmazást választjuk exportáláskor, hanem más a célplatform, akkor ezek közül az objektumok közül a megfelelőt kiválasztva az adott platform egyéni lehetőségeihez férhetünk hozzá.
- Web: Az internet böngészésével kapcsolatos objektumok helye. Itt alkalmazhatunk Ajax lekéréseket, vagy kezelhetjük az alkalmazásunkat futtató böngészőprogramot.

Viselkedések

A szoftver rendelkezik néhány előre elkészített viselkedésmintával, amelyeket az objektumokhoz hozzárendelve a fejlesztési folyamat jelentősen felgyorsul. Szemléltetve a funkciót arról van szó, hogy egy adott elem viselkedésének a leprogramozását kikerüljük és helyette a már előre elkészítettet szabjuk testre. Ezzel időt és kódot takarítunk meg, nem is beszélve arról, hogy a program ezt nem számolja fel eseményként, így az ingyenes licenz korlátozásában sem kerülünk közelebb a határértékhez.



35. ábra, Az objektumokhoz rendelhető viselkedések.

A következő kategóriákba sorolhatjuk a viselkedéseket (35. ábra):

- **Attribútumok:** Alapvető tulajdonságok, amelyekre további viselkedések reagálnak. Például: tömör objektum (nem lehet áthaladni rajta).
- **Általános:** Szokványos tulajdonságokkal ruházza fel az objektumot, amelyek nem kapcsolódnak közvetlenül az interakcióhoz, inkább a háttérben meghúzódó viselkedések. Például: áttűnés, elrendezésen kívüli törlődés.
- **Mozgások:** Az objektumok mozgását befolyásoló viselkedések. Segítségükkel „hozhatjuk be” a felhasználót a szoftverbe. Ezekkel a modulokkal határozhatjuk meg, hogy milyen interakció jöhet létre az elemek, vagy a felhasználó és bizonyos elemek között. Például: 8 irányú mozgás, forgatás, egyéni mozgás, útkeresés.

Amennyiben olyan funkciót szeretnénk hozzárendelni az objektumhoz amely nem található az előre leprogramozottak listájában, abban az esetben vagy létrehozunk azt az eseménylapokon, vagy kiegészítő viselkedést írunk a fejlesztői környezethez (bővebben a *Kiegészítések, kódolás* alfejezetben).

Eseménylapok, vizuális programozás

A szoftver igazi különlegessége az eseménylapok kezelésében és a vizuális programozás folyamatában rejlik. Hierarchikus felépítést alkalmaz és követi az alapértelmezett futtatási sorrendjét az utasításoknak (fentről-lefele) az adott eseményen belül.

Első lépésként meg kell határoznunk egy eseményt (*event*), amely teljesülése esetén az ahhoz hozzárendelt utasítások lefutnak. Egy eseményen belül több feltételt is megadhatunk, így kialakítva AND vagy OR kapcsolatokat. Ezt követően tudunk parancsokat adni a programnak.

Minden esemény és utasítás valamilyen objektumhoz kell, hogy kapcsolódjon (amennyiben nem adtunk hozzá a projekthez, az esetben a rendszerobjektum áll csupán rendelkezésünkre).

Példa: Amennyiben a bal egérgommbal kattintottunk, akkor a változó értéke növekedjen meg 10-zel. Ehhez a példához szükségünk van az egér objektumra, illetve egy szám típusú változóra.

A program három típusú változót kezel (melyeknek nincsenek további típusai): szám, szöveg és logikai. A változókat három helyen határozhatjuk meg:

- Globális változókat az eseménylapokhoz rendelhetünk.
- Lokális változókat az eseményblokkokhoz rendelhetünk.
- Példány változókat az adott objektum adott példányához rendelhetünk.

Az eseménylapon férhetünk hozzá (és először találkozhatunk vele) a rendszerobjektumhoz (System). Ezt az elemet nem szükséges (és nem is lehetséges) hozzáadni a projekthez, hiszen minden Construct 2 alkalmazás része. Segítségével kezelhetjük a globális és lokális változókat, ciklusokat indíthatunk, válthatunk különböző elrendezések között, kezelhetjük azok rétegeit stb.

A támogatott ciklusok:

- For: Egy index változó segítségével ismétli az utasításokat.
- For Each: Az adott objektum példányszámával megegyező alkalommal lefut.
- Rendezett For Each: Az adott objektum példányszámával megegyező alkalommal lefut, egy kifejezés segítségével rendezett sorrendben.
- Egyszerű ismétlés: A megadott számú alkalommal ismétli a benne foglalt utasításokat.
- While: A meghatározott feltétel teljesüléséig ismétli a ciklust.

A forráskódunkat lehetőségünk van kényelmesen rendezni, melyet két fő kategóriába sorolok:

- Esztétikai elrendezés: Elsődleges célja a kód átláthatósága. Csoportokat hozhatunk létre, melynek alárendelhetjük az egyes eseményhalmazokat, vagy megjegyzéseket szúrhatunk be.

- Funkcióbeli elrendezés: Lehetőségünk van alesemények létrehozására, amelyek csak a fő esemény minden utasítása után futnak le. Ezzel további feltételek megadását spóroljuk meg, kialakítva egy AND kapcsolatot a szülőeseménnyel és egyben átláthatóbbá téve a kódot.

Mindezek mellett az eseménylapok rugalmasan módosíthatóak. Az egyes utasítások vagy események objektumait kicserélhetjük, vagy a megadott értékeket átírhatjuk a teljes utasítás szerkesztése nélkül. Az egyes elemeket „fogd és vidd” módszerrel könnyedén mozgathatjuk.

Erőteljes tulajdonsága az eseménylapoknak, hogy egymásba ágyazhatók. Így elkészíthetők a program különböző részeit vezérlő eseménylapok, majd azok meghívásra kerülnek szükség esetén. A beágyazott lapok természetesen hozzáférnek a szülőelem változóihoz. Fontos megemlíteni, hogy ez a funkció nem megegyező a függvények használatával hagyományos programozási nyelvekben. Arra külön lehetőséget biztosít a Construct 2.

A Construct 2-vel történő vizuális programozás során megvalósulnak a programozás szekvencialitás, szelekció és iteráció alapelvei. Az utasítások végrehajtása egymást követő, szekvenciális sorrendben történik. Az eseményblokkok szelekció révén futtatják le a beléjük foglalt utasításokat. Valamint lehetőségünk van a korábban bemutatott feltételes ciklusok, iterációk használatára.

Hálózatkezelés

A fejlesztői környezet lehetőséget biztosít arra, hogy a WebRTC technológiára támaszkodva (W42, 2014) hálózati programokat is készítsünk. A HTML5 lehetőségeiből kiindulva a kapcsolat a kliensek között peer-to-peer elven működik és létrehozásához egy ún. jelző szerverre van szükség. Ez a szerver csupán a kapcsolat kialakításában játszik szerepet, az adatátvitelben már nem. Az első kapcsolódó kliens lesz a szerver, amelynek a kapcsolódási adatait a jelző szerver átadja a többi kliensnek, így azok rákapcsolódnak. Ez a megközelítés ugyan felvet néhány hálózatkezelési problémát, de a Scirra számos optimalizációt és konfigurálást végzett a motor hálózatkezelő részén, illetve a jelző szerveren is, így ez a megvalósítás a gyakorlatban működőképes és a szervert a lehető legminimálisabb adatforgalom terheli.

A Scirra saját jelző szerverét futtat, amelyet a fizetős licensszel rendelkező felhasználók korlátlanul igénybe vehetnek, de lehetőséget biztosít a jelző szerver szoftverének megvásárlására is (24 \$), amennyiben nagyobb, központosított hálózatot igénylő projektben gondolkodunk.

Előnézet, tesztelés

A programfejlesztés során is fontos elem, a Construct 2 esetében pedig kiemelkedően fontos a készülő projekt előnézete és tesztelése. Így nemcsak a hibásan működő elemekre kapunk rálátást, de szembesülünk eddigi munkánk eredményével is, amely (különösképp a fiatal fejlesztők esetében) további lendületet adhat a munka folytatásához.

Az előnézet minden esetben az egyik telepített böngészőprogramban történik (ez kiválasztható a projekt beállításainál). Az előnézet gombra kattintva a legutóbb kiválasztott elrendezés kerül betöltésre, ezért amennyiben az egész projektet tesztelni szeretnénk, célszerű a legelső elrendezést kijelölni az előnézet indítása előtt.

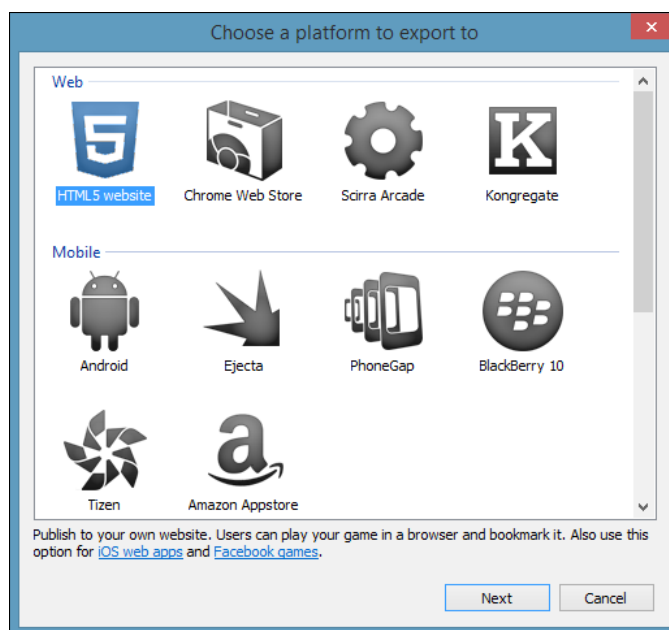
A program felkínálja az úgynevezett LAN hálózaton keresztüli előnézetet a fizetős licensszel rendelkező felhasználók számára. Ezt engedélyezve a Construct 2 egy lokális webszervert hoz létre, amelyen keresztül elérhető a teljes lokális hálózat számára az éppen tesztelés alatt lévő szoftver a kiszolgáló számítógép helyi IP címének beírásával. Ezzel a módszerrel több számítógépen és akár több platformon is egyszerre letesztelhető a készülő projekt. Amennyiben új tesztet indítunk, az összes eszközön megnyitott böngészőben automatikusan frissül az előnézeti verzió.

Ugyancsak a fizetős funkciók közé tartozik a debug előnézet opció (W43, 2014). Ennek a speciális előnézeti módnak a feladata, hogy a fejlesztők a program futása közben láthassák a főbb rendszerfolyamatokat és könnyedén kiszűrjék, hogy a programkód mely része okozza a hibát. Ez a mód is, a hagyományos előnézeti módokhoz hasonlóan, futtatható LAN hálózaton keresztül.

Exportálás

Amikor a kész projektünket exportálni szeretnénk, létrehozunk egy a környezet használata nélkül is futtatható alkalmazást a kiválasztott platformra. Sok esetben amikor egy fejlesztő több platformot céloz meg, akár a teljes forráskódot át kell írnia, hogy mindegyikre meg tudja

azt jelentetni. Szerencsére a HTML5 rugalmassága miatt a Construct 2 esetében erről nincs szó, mivel elég néhány apróbb módosítást elvégeznünk a programunkon, hogy alkalmazkodni tudjon a platformok közti differenciálódáshoz.



36. ábra, A projektexportáló felület.

Az exportáló modul egyik beépített funkciója a HTML5 kód „kisebbitésének” a lehetősége. A HTML5 kód a Google cég Closure fordító motorjával tömörítésre kerül, valamint torzításra is, így jelentősen megnehezíti a visszafejtési folyamatot. Ezzel is növelve a kész projekt biztonságát.

A program r187-es verziójától kezdve a natív asztali alkalmazások exportálásához szükséges modult külön kell letölteni a hivatalos weboldalról (W44, 2014), erőforrás optimalizációs okokból.

Kiegészítések, kódolás

Egyes fejlesztők kimeríthetik a Construct 2 nyújtotta lehetőségeket, vagy egyszerűen nem találják meg benne azt a funkciót, amelyre szükségük lenne a projekt elkészítéséhez.

Lehetőség van külső kiegészítő modulokat írni a fejlesztői környezethez, amelyeket betöltve új tulajdonságok és viselkedések rendelhetők az objektumokhoz. A kiegészítő modulokat JavaScript nyelven kell leprogramozni. Teljes dokumentáció áll rendelkezésre a szoftver mellé, hogy a fejlesztők használatba vehessék a hozzá csatolt SDK csomagot (W41, 2013).

Fontos megjegyezni, hogy ez az egyetlen módja annak, hogy hagyományos programozási nyelvet alkalmazzunk a projektünk elkészítése során.

A szoftver felhasználói közössége számos érdekes és hasznos kiegészítőt tett közzé szabad felhasználással a Construct 2 közösségi oldalán (W26, 2013). Ezeket felhasználva az átlagfelhasználó (JavaScript ismerete nélkül) is még kényelmesebbé teheti a fejlesztési folyamatot.

Korlátok, hátrányok

Természetesen a Construct 2 esetében is korlátokba és hátrányokba ütközhetünk, melyek talán egyes emberek számára nem a legoptimálisabb választássá teszik a szoftvert.

- A program angol nyelvű, amely az angolul nem tudó felhasználók számára negatívan hat. Hozzátenném, hogy a használatához nem szükséges kiemelkedő angol nyelvtudás, mivel csupán néhány beállítást kell megjegyezni, valamint eligazodni az eseménylapok lehetőségei között. Igény szerint elkészíthető a nyelvi fájlok fordítása más nyelvekre (egy közel 6000 soros XML fájl átírásával).
- A Scirra cég erőforráshiány okából harmadik felekre támaszkodik bizonyos exportálási funkciókat illetően. Ennek fő hátránya a harmadik fél felé való kiszolgáltatottság, ugyanis amennyiben annak a szolgáltatása megszűnik, a program adott modulja értelmét veszti. Amennyiben megteheti, a fejlesztőcég integrálja a harmadik forrásból származó modulokat a fejlesztőkörnyezetbe (például az asztali natív alkalmazásokhoz alkalmazott node-webkit). A többi esetben (iOS, Android) viszont ez az állapot fenn áll és nem is várható változás, ameddig a cég nem bővíti az alkalmazottjai számát.
- A HTML5 elsősorban webes használatra lett kifejlesztve. Ebből adódóan nem szükségszerűen a legmegfelelőbb választás asztali vagy mobil natív alkalmazások készítéséhez a konvertálási folyamatok okából. Érdekes azonban hozzátenni, hogy a fejlesztőcég folyamatos teszteléseket és optimalizálásokat végez annak érdekében, hogy ez ne okozzon problémát, és sem kompatibilitás, sem pedig optimalizálási szempontból ne legyen felfedezhető különbség.
- A fejlesztési folyamat során nem alkalmazhatunk hagyományos programozási nyelvet a konkurenciával ellentétben. Erre nem feltétlenül van szükség, ugyanis az eseménylapok segítségével elméletben minden komplexebb kód leprogramozható,

mégis néhány veterán programozó fejlesztőt ennek a lehetőségnek a hiánya zavarhat. A fejlesztőcég ennek orvoslására a kiegészítők írásának lehetőségét hozza példának, amellyel bár ténylegesen bővíthető a program funkcionalitása, mégis el kell ismernünk, hogy ez csupán egy félmegoldás.

- Bár a vizuális programozást alkalmazó programok használata jelentősen felgyorsítja a programozási folyamatot a hagyományos programozási nyelvekkel ellentétben, mégis a motor határaiba ütközhetünk. Amennyiben abból a tényből indulunk ki, hogy a Construct 2 típusú szoftvereket is hagyományos programozási módszerekkel írták, rájöhetünk, hogy azoknál egy mindenképp korlátozottabb rendszert kapunk kézhez. Minden fejlesztőnek el kell döntenie, hogy számára melyik a legcélravezetőbb eszköz, amellyel elkészítheti a szoftverét.

Dokumentáció, leírások

A program kiemelkedően jól kidolgozott dokumentációval rendelkezik, melyet nem csupán tartalombeli bősége és hasznossága miatt értékelhetünk, hanem elérhetőség és nyitottság szempontjából is.

- Hivatalos kézikönyv: Elérhető online (W8, 2013) és offline (W36, 2013) formában is (utóbbi letöltve, vagy a program mellé csatolva, a dolgozathoz csatolt CD melléklet *manual* mappájában is megtalálható). A kézikönyv a legutóbbi stabil programverzióra alapszik és annak minden funkcióját és lehetőségét részletesen taglalja, ismerteti. Emellett integrálva van a fejlesztőkörnyezetbe, amely hivatkozásokként funkcionál az éppen aktuális párbeszédpanelhez igazodva (más szavakkal megfogalmazva: az adott munkafolyamatnál felhasználásra kerülő funkcióval releváns kézikönyv hivatkozás jelenik meg). A kézikönyv angol nyelvű.
- Felhasználói segédletek: A fejlesztőcég weboldalán hivatalos menüpontként szerepelnek a felhasználók által készített, ingyenesen hozzáférhető segédletek (W9, 2013). Ezek elsősorban leírásokat és példaprogramokat jelentenek (néhány esetben videoklipet is), melyek egy adott problémát írnak körül és annak megoldását prezentálják. Ilyen jellegű segédletet bárki létrehozhat, majd azt a többi felhasználó értékelheti. Így a közösség válogatja ki, hogy melyek azok a leírások, amelyek a leginkább hasznosak. Egy közelmúltbeli frissítést követően a leírások fordítása is

lehetséges további nyelvekre (W37, 2013). Ennek hála az angolul nem tudó felhasználók is találhatnak online segédletet anyanyelvükön íródva (feltételezve, hogy előzetesen valaki elvégezte annak lefordítását).

A konkurens szoftverek tekintetében nem találkoztam ehhez fogható dokumentációval. A leginkább elterjedt GameMaker eszköztárából hiányzik a könnyen elérhető kézikönyv (W38, 2013), míg a Stencyl fejlesztőcsapata nem tesz közzé abból letölthető verziót (W39, 2013). Emellett egyik konkurens szoftvernél sem láttam ilyen jól kidolgozott, közösség támogatott, egységes leírási segédleteket.

3. Melléklet: Tartalmi kiegészítések

T1: Szükséges beszélni a szoftver telepítésének módjáról, a szükséges rendszerkövetelményekről, illetve a beszerzési forrásokról (utóbbit gyakorlatban is megmutatni a fejlesztőcég weboldaláról). Fontos, hogy a tanuló önállóan is fel tudja telepíteni a fejlesztőkörnyezetet az otthoni számítógépére (amennyiben rendelkezik ilyennel – bár ez napjainkban általános), hogy elvégezhesse a házi feladatokat, illetve felfedezhesse saját igénye szerint a programot.

T2: Első lépésben a tanár, amennyiben nem rendelkezik efféle szolgáltatással, létrehozhat egy közös tárhelyet maga és a tanulói számára, ahol (minimum tanári oldalról) elérhető az összes órai, esetleg otthoni munka is. Fontos, hogy az órai munkát biztonsági okokból mindenki a saját számítógépén létrehozott mappában végezze és csak a végeredményt töltsse fel a közös mappába. Erre a célja bármilyen web-, vagy FTP tárhely megfelel, de az ingyenes felhőalapú szolgáltatások szolgálnának a legpraktikusabban (például: Dropbox (W30, 2014), Google Drive (W31, 2014), Microsoft OneDrive (W32, 2014)). Ezt a tárhelyet használnák a csoport tagjai a tanulás során, így nyomon követhető lenne munkájuk, nehézségeik és egyéni fejlesztéseik.

T3: A modulok feladatai a Construct 2 r184-es stabil verziójával lettek elkészítve, visszafelé nem kompatibilisek. A forráskódok megtalálhatóak a dolgozathoz csatolt CD melléklet *feladatok* mappájában.

4. Melléklet: Feladatok technikai kiegészítése

Feladat 1: Békás autóút

F1: A teljes képernyős megjelenítést kapcsoljuk ki (*Fullscreen in browser* beállításnál az *Off* opciót válasszuk). A teljes képernyős megjelenítés kikapcsolására azért van szükség, mert a Construct 2 alapértelmezetten egy méretarányokat megőrző teljes képernyős beállítást ad meg, amelyre a feladat megoldásához nincs szükségünk.

F2: Az elrendezés mérete az ablakon belül megjelenítendő terület nagyságát határozza meg. Az elrendezés margója pedig azt a területet jelöli, ahol a program fő eseményei történni fognak (a program futtatásakor nincs szerepe, a szerkesztési folyamatban szolgál segítségként).

F3: A *Tiled Background* objektum általános háttérelemként szolgál. A szerkesztőfelület koordináta rendszerének origója az elrendezés bal felső sarka, ahonnan jobbra indul az x tengely, illetve lefele az y tengely, mindkettő pozitív intervallumban.

F4: A *8Direction* viselkedés 8 irányú mozgást tesz lehetővé. A viselkedés beállításainál a *Set angle* tulajdonságot állítsuk *No* értékre. A *Set angle* opció konfigurálja, hogy az objektum automatikusan forgatásra kerüljön-e. Alapértelmezett esetben 360 fokban forgatja az elemet.

Feladat 3: Ugrálás a párkányokon

F5: A *Solid* viselkedéssel ellátott objektumokon a *Platform* viselkedést használó elemek nem tudnak áthatolni.

F6: Az objektum adott példányát kijelölve megjelennek a kijelölés szélén az átméretező gombok (15. ábra).

F7: A *Platform* viselkedés vízszintes és függőleges (ugrás) mozgást valósít meg, miközben az objektumra szimulált gravitációs erő hat.

Feladat 4: Tájképek

F8: A képkockákat az objektum grafikai szerkesztő felületén az *Animation frames* ablakban kezelhetjük (16. ábra).

F9: A *Speed* tulajdonság az animáció sebességét (kép/másodperc) állítja, a *Loop* opció pedig az automatikus ismétlést. Egy ilyen egyszerű animáció 10-es *Speed* értékkel már az emberi szemnek azt a látszatot kelti, hogy a program késleltetés nélkül megváltoztatja a képkockát.

F10: A *Mouse* objektum teszi lehetővé, hogy a program hozzáférjen az egér beviteli periféria használatához minden objektumra vonatkozóan. Ahhoz, hogy a „gomb” objektum kattintható legyen, szükségünk van a *Mouse* objektumra is az egérkezeléshez.

F11: Az eseményeket és utasításokat mindig a hozzá kapcsolódó objektum számára rendelkezésre álló lehetőségei közül kell kiválasztani.

F12: Az *On any frame changed* feltétel az animáció képkockaváltásának megtörténését figyeli.

Feladat 5: Hazaút

F13: A *ScrollTo* viselkedés segítségével az adott objektum kerül az elrendezés középpontjába. Így mikor az kilépne az ablak területéről, az elrendezés görgetésre kerül, hogy a *ScrollTo* viselkedéssel ellátott elem mindig a képernyőn maradjon.

F14: A „talaj” objektum fogja meggátolni, hogy a „lány” elem kiessen az elrendezésből.

F15: A „talaj” objektum a „fu” objektum mögött található (mivel előbb adtuk hozzá az elrendezéshez, mint a „fu” elemet), az elrendezés aljához igazítva.

Feladat 6: Kenyerek a ládába

F16: A *Text* objektumok alapértelmezett szövege „Text”.

F17: A *Drag & Drop* viselkedés „fogd és vidd” típusú mozgathatóságot valósít meg az adott objektumra. A bal egérgombot nyomva tartva az elemet mozgathatjuk azt.

F18: A „szam” változó fogja tárolni az elpakolt kenyerek számát. A feladat későbbi szakaszában ennek a változónak az értékét fogjuk kiírni. Az eseménylapokon létrehozott változók fő típusa a globális változó.

F19: Az *On drop* feltétel a *Drag & Drop* viselkedés velejárója. Azt vizsgálja, hogy az adott objektumot „letettük-e.”

Feladat 7: Almaszedés

F20: Figyelmeztetés: A *Physics* viselkedés nagy számításigényű magas objektumszám esetén, ezért gyengébb teljesítményű számítógépeken erre külön figyelmet kell fordítani.

F21: Az *Inmovable* tulajdonság lehetővé teszi, hogy a *Physics* viselkedéssel ellátott elemre bár hatnak a szimulált fizikai környezet hatásai, mégis elmozdíthatatlan legyen. Két *Physics* viselkedéssel rendelkező objektum interakcióba lép egymással a szimulált fizikai értékek figyelembe vételével.

F22: Az *On start of layout* feltétel az elrendezés betöltődésekor aktiválódik.

F23: A *Set world gravity* utasítás a *Physics* viselkedéshez kapcsolódó utasítás és a szimulált gravitációt konfigurálja. 0 értékkel beállítva az objektumra nem hat gravitációs erő.

F24: 10-es erősségű gravitációs hatást még követni tud a felhasználó szemmel.

Feladat 8: Keverőpult

F25: A szöveg színét a *Text* objektum *Color* beállításánál adhatjuk meg.

F26: A *Volume* beállítás a lejátszott fájl hangerejét állítja be.

Feladat 9: Lisztes zsákok megszerzése

F27: Amennyiben projektünkben utasítás segítségével hozunk létre objektumokat a program futása közben, a létrehozandó objektum egy példányát legalább egy elrendezésbe be kell illeszteni a HTML5 technikai felépítése okából.

F28: Az *Every X seconds* esemény a benne foglalt utasításokat a paraméterként megadott idő elteltével mindig újból végrehajtja.

F29: A *random(K)* utasítással véletlenszámot generálunk 0 és az argumentumként megadott K értéktől egyel kevesebb intervallumban. Más szavakkal a $[0; K-1]$ intervallumban. Az *int()* utasítással pedig egész számmá alakítjuk az előbb generált véletlen számot.

Feladat 10: Úrhajó követése

F30: A *Bound To Layout* viselkedés meggátolja az objektumot, hogy kilépjen az elrendezés határain kívülre. A *Directions* opció alatt adhatjuk meg a 8 *Direction* viselkedés mozgási irányait.

F31: A *Wait* utasítás szünetelteti a következő utasítások és alesemények lefutását az adott blokkon belül.

F32: A *Trigger once while true* feltétel a hozzá kapcsolt további feltételek teljesülése esetén csak egyszer engedélyezi a blokk lefutását egészen addig, ameddig újra meghívásra nem kerül. Üresen hagyva végrehajtja feltétel nélkül a hozzá kapcsolt utasításokat.

F33: Az *urhajo.X* hivatkozás lekérdezi az „urhajo” nevű elem pozíciójának x koordinátáját.

Feladat 11: Csillagos éjszaka

F34: Amennyiben nem képet használunk háttérnek, lehetőségünk van egyszínű háttérkitöltést megadni a rétegek beállításainál.

F35: A *While* ciklus párosítva az *Every X Seconds* feltétellel azt eredményezi, hogy a ciklusmag egyes végrehajtásai között a feltételben beállított intervallumnak el kell telnie.

F36: Az *int()* és *random()* függvények működése a 9. feladatban ismertetve van.

F37: A *Set scale* utasítás méretarányos átméretezést valósít meg a megadott argumentum alapján.

F38: A *Compare variable* feltétellel megvizsgálhatjuk egy változó értékét különböző relációk alapján.

Ezzel a feltétellel kiegészítve a ciklus csak abban az esetben fog lefutni, amennyiben a „megallitva” változó értéke 0.

F39: A *Set value* parancs segítségével adhatunk egy változónak új értéket.

Feladat 12: Labirintus

F40: A *9-patch* objektum segítségével lehetőségünk van ismétlődő textúrát használó objektumokat hatékonyan létrehozni, amely átméretezésénél a kép nem torzul, helyette ismétlődik. Esetünkben ideális fal objektum létrehozásához.

F41: A *margin* beállítások 0 értékre állításával megadjuk, hogy a *fal.png* kép nem tartalmaz margókat. Az *Edges és Fill* opciók *Tile* beállítása utasítja az objektumot, hogy ne nyújtsa, hanem egymás után ismétlődően ugyanazzal a képpel töltsse fel az objektumot.

F42: A *Solid* viselkedéssel rendelkező objektumokat fogja akadályként tekinteni az útvonalkereső algoritmus.

F43: A *Pathfinding* viselkedés automatikus útvonalkeresést valósít meg. A *Cell border* beállítás segítségével tudjuk meghatározni, hogy az objektum mennyire közelítheti meg, illetve hatolhat bele az akadályként meghatározott objektumokba az útvonal keresése során.

F44: A *Find path* utasítás a megadott koordinátákra elvégzi az útvonalkeresést. Az *eger.X és eger.Y* hivatkozás lekérdezi az eger pillanatnyi X és Y koordinátáit.

F45: A *Move along path* utasítás végigmozgatja az objektumot a megtalált útvonalon.

F46: Lehetőségünk van egyes feltételek „megfordítására” a feltételen való jobb egérgommbal való kattintás után a helyi menüből az *Invert* opció kiválasztásával. Ilyenkor a feltétel tagadását vizsgálja a program. A projekt esetében azt vizsgálja, hogy az egérmutató nincs a „fal” objektum fölött.

Feladat 13: Erre csörög a dió

F47: Az előnézeti böngészőprogramot a projekt tulajdonságai közül a *Preview browser* pontnál tudjuk megadni. A *Chrome* böngésző használata azért szükséges, mert a projektben alkalmazott hangkezelést jelenleg csak az a böngésző támogatja.

F48: A Construct 2 minden további hozzáadott réteg hátterét automatikusan áttetszőre konfigurálja. Ezt a beállítást tudjuk a *Transparent* opcióval állítani. Az *Opacity* opcióval lehetőségünk van a réteget (és annak teljes tartalmát) részlegesen áttetszővé tenni. A 80-as érték 80%-os láthatóságot jelent.

F49: A *Reference distance* opció azt a referencia távolságot adja meg (pixelben), amely az elhalkulás kiszámításához szükséges. A *Maximum distance* opció azt a maximális távolságot adja meg (pixelben), amely után a lejátszott hang nem halkul tovább.

F50: A *LayoutWidth* és *LayoutHeight* függvények segítségével lekérdezhető az aktuálisan futtatható elrendezés szélessége és magassága pixelekből.

F51: A *Play at object* utasítás segítségével játszhatunk le hangokat különböző objektumok pozíciójához rendelve.

F52: A *Set listener object* adja meg azt az objektumot, amely szemszögéből a síkban elhelyezett hangokat felfogjuk.

Feladat 14: Valahol a sötétben

F53: A koordináták véletlen generálásakor figyelembe vesszük a „gyerek” objektum méretét, hogy az biztosan az elrendezésre kerüljön és ne lógjon le róla.

F54: Az *Is overlapping another object* feltétel azt vizsgálja, hogy a két objektum átfedésben van-e (részleges átfedés esetén is teljesül a feltétel).

Feladat 15: Köszönés

F55: A *Default controls* opció kikapcsolásával az ember objektumokat nem tudjuk az alapértelmezett billentyűkkel (a kurzormozgató nyilakkal) mozgatni.

F56: A *Function* objektum teszi lehetővé a hagyományos programozási nyelvekben is alkalmazott függvények használatát.

F57: Az *On function* feltétel definiálja a függvényt és az ehhez a blokkhoz rendelt utasítások lesznek a függvény részei.

F58: Esetünkben amennyiben a turista bal oldalról 50 pixelre megközelíti az adott ember objektumot, a feltétel teljesül.

F59: A *Call function* utasítással meghívhatunk korábban definiált függvényeket.