[Külső fekete borítólap formátuma]

Széchenyi István Egyetem

Gépészmérnöki, Informatikai és Villamosmérnöki Kar

Informatika Tanszék

**SZAKDOLGOZAT**

**Szűcs Henrik**

**Gazdasági Informatikus BSc szak**

2022

| [Gerincen:] Hallgató Neve, Évszám {Titkosított} |
| --- |



**SZAKDOLGOZAT**

**Webalkalmazás fejlesztése visual scripting segítségével**

**Szűcs Henrik**

**Gazdasági Informatikus BSc szak**

**2022**

**[Ennek a bekezdésnek a helyére szerkessze be az aláírásokkal ellátott feladatkiírási lap szkennelt változatát.]**

# Nyilatkozat

Alulírott, Szűcs Henrik (G28AZI) Gazdasági Informatikus BSc szak szakos hallgató kijelentem, hogy a Webalkalmazás fejlesztése visual scripting segítségével című szakdolgozat feladat kidolgozása a saját munkám, abban csak a feltüntetett forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, 2022.09.09

hallgató

# Kivonat

Gazdasági Informatikus BSc szak

[1 oldalas, magyar nyelvű tartalmi kivonat]

A Construct 3 webes technológiákon alapuló visual scripting keretrendszer, melyet elsősorban játékok fejlesztéséhez terveztek. Mivel a legtöbb mai webalkalmazás a felhasználói interakciók, válaszidők és grafikai megjelenítés tekintetében is hasonló egy számítógépes játékkal, felmerül a kérdés, hogy ezt a keretrendszert fel lehet-e használni általános célú webalkalmazások készítésére, esetleg különféle bővítmények elkészítése árán. Ez a kezdeti befektetés megtérülhet, ha később kevésbé magasan képzett fejlesztők is bevonhatók egy összetett szoftver fejlesztésének folyamataiba.

# Abstract

[Title in English]

[1 oldalas, angol nyelvű kivonat]

Lorem ipsum, dolor sit amet

# Tartalomjegyzék

[**Bevezetés**](#_cgzg9wnxn5k) **1**

[Témaválasztás indoklása:](#_ht5770aingsc) 1

[Probléma a végfelhasználó szemszögéből:](#_97qb6khpvas) 1

[Probléma a fejlesztő szemszögéből:](#_dnaq38lyhwip) 2

[Construct 3 felépítése:](#_u4k9lemlzn5n) 6

[Szerkesztő:](#_m1zu4yx2qdot) 6

[Működési környezet:](#_56ru8yv82s6t) 7

[**Szerkesztő**](#_m4n5yksiwj3g) **7**

[Kezelőfelület:](#_ue25a6ef6yy6) 7

[Bővítmények:](#_ovtpuuib1016) 8

[Bővítmények:](#_3hz53fffuxtr) 8

[Viselkedések:](#_yt4nayqp0nha) 8

[Effektek:](#_xo6f5tbfgfhd) 9

[Készített program felépítése:](#_atbfl5zg4h50) 9

[Szerkesztő SDK:](#_2vsfz4yoslfj) 10

[Fejlesztői mód:](#_vs39b1pir0i3) 10

[Környezet:](#_9miai889ddxc) 10

[Közös struktúra:](#_ju62kyi6p6g1) 11

[Bővítmény mappa struktúra:](#_l9o8ws369ej5) 11

[Viselkedés mappa struktúra:](#_gguka9obqal8) 12

[Effekt mappa struktúra:](#_y4s0g5mg33h1) 12

[**Futási környezet**](#_u8w6542dzn10) **12**

[Tömörítés:](#_29zzmwkoxsap) 12

[Nincs:](#_fkubjubmwmy5) 13

[Egyszerű:](#_vjchx2dlun3u) 13

[Haladó:](#_fvnzdyn1arh) 13

[**Kiegészítők**](#_11pug8uiw87e) **13**

[Szükséges kiegészítők:](#_jfk7is53oemk) 13

[**Tömeges email küldő**](#_djhl676mh1j1) **13**

[**Irodalomjegyzék**](#_1ac8w2pcohjb) **14**

[**Mellékletek**](#_4cdc7r4caisp) **16**

# Bevezetés

## Témaválasztás indoklása:

Munkám során a megrendelők kész programokat rendelnek, azonban szinte az első elkészítés után mindig felmerül a bővítés igénye. A hagyományos módon elkészített programokhoz csak speciális szaktudású programozó tud hozzátenni. Személyes célom, hogy egy megírt programot könnyen lehessen bővíteni és a programkészítés folyamata bárki számára elérhető legyen még ha nem is jártas a programozásban. A szakdolgozat egy problémán keresztül mutatja be az elképzelésem megvalósítását. A cél leginkább az, hogy az általam készített program támogatni tudja a felhasználók kreativitását, az önálló gondolatok szárnyalását. Ötleteket, eszközöket szeretnék adni az embereknek, ahhoz, hogy fejleszteni tudják az együttműködési képességüket, a csapatmunkát illetve a hatékony kommunikációt. A pályakezdő informatikusok számára rendkívüli nehézségekkel jár pályájuk kezdete. Egyes becslések a pályaelhagyókat 30-50 százalék közé teszik a informatikus pálya első öt évében. A program feladata lehet, hogy a informatikus jelöltek szakmai fejlődését elősegítse. Ennek a támogatásnak minden esetben a programozó igényeihez kell igazodnia.

## Probléma a végfelhasználó szemszögéből:

Small/home office, kisvállalati, oktatási környezetben jellemző, hogy évente egy-két alkalommal rengeteg felhasználónak kell köremail-t küldeni. Valós példák: Országos Magyar Vadászkamaránál az éves jegyhosszabbítás február végén történik és az esetleges tudnivalókról több ezer tagot kell értesíteni egy időben. Oktatásban évkezdéskor a tanároknak több száz diákot, szülőt kell értesíteniük a változásokról köremail formájában.

A köremail küldése nagyban eltér a megszokott, egyszerű email küldésétől. Az általános email küldés 1-25 főig probléma nélkül kivitelezhető szinte minden emailes rendszeren. Lehetséges beállítani, hogy mely emailcímek lássák egymást, illetve beszélgetés folytatása is kivitelezhető. Sok ember számára kiküldött, tömeges e-mailek esetében rengeteg problémába ütközhetünk. Az email küldő kliens egyáltalán nem tudja kezelni ha sok embert adunk meg címzettként (nem enged annyi embert beleírni vagy egyszerre akarja megjeleníteni és a gép nagyon belassul/lefagy). Nem tudjunk személyre szabni a levelet (Pl.: név), esetleg spam-nek ítéli a címzett. Továbbá a szerver nem fogja tudni egyszerre kiküldeni a leveleket. Ha a kiküldött levelekre valaki válaszol, akkor nem biztos, hogy azt a mi e-mail címünk helyesen kezeli és kategorizálja.

Lehetséges megoldások a problémára:

* Szolgáltatás igénybevétele: Egy tömeges email küldő szolgáltatásra való előfizetés. (Mailchimp, Mailgun stb.)
  + Előnyök: Azonnal rendelkezésre áll, nagy kínálat a szolgáltatások között
  + Hátrányok: Szolgáltatás mivolta miatt az egyszeri használat drága lehet (általában a cél a nagyobb csomag megvásárlása). Előfizetés kezelése (állapot, lemondás, előfizetés megújítása), általában külön fiók és előfizetés kell minden felhasználónak.
* Program használata:
  + Előnyök: teljesen független az előfizetésektől, bárki számára hozzáférhető, használható. Hosszútávon olcsóbb
  + Hátrányok: email fiók szükséges

A felhasználó közönséget figyelembe véve valószínűsíthető, hogy egy a felhasználó számítógépén futó program lehetne a jobb megoldás. Irodai dolgozók és tanárok a szolgáltatást nem akarják, illetve nem tudják kezelni, az előfizetés kezelése pedig a legtöbb intézményben kérdéses: hogy ki kezelje és ki fizesse. Jelenleg a következő programokat találtam alkalmasnak a tömeges email küldésre:

* Thunderbird:

Mozilla alkotóinak levelező kliense. Nyílt forráskódú, ingyenes, rengeteg platformra letölthető. Ingyenessége miatt nagyrészt közössége fejleszti azt. Nincsenek prémium funkciók és szolgáltatások a programhoz mellékelve. Használata kicsit archaikus, gyakorlást igényel ha másik levelező rendszerről térünk át használatára.

* + Előnyök: ingyenes
  + Hátrányok: nincs „1st party” (közvetlen) megoldás tömeges email küldésre, csak „3rd party” (külsős) kiegészítővel lehetséges, nehézkes beüzemelése (kiegészítők megbízhatósága kérdéses)
* Sendblaster:

Delivery Tech Corp. által fejlesztett Windows-os szoftver (kinézetre .NET, WPF applikáció), A fejlesztő által „befejezett” szoftvernek látszik (2020.04.10 utolsó frissítés). 99 euró + frissítések és kiegészítők további költséggel járnak.

* + Előnyök: egyszerű használat, kész alkalmazás
  + Hátrányok: drága (a fő verziókat külön kell megvenni), csak Windows rendszerre elérhető, jövőbeli frissítések kérdésesek (2020.04.10 utolsó frissítés)

Cél: Kellene egy program, amivel egyszerűen lehet tömegesen leveleket küldeni megadott e-mail-címről. Elérhető legyen a legtöbb platformra és olcsóbb legyen.

## Probléma a fejlesztő szemszögéből:

A probléma nem érint nagy csoportot, így nem gazdaságos a programot teljesen alapjaitól megírni, érdemes kiválasztani egy keretrendszert a program elkészítéséhez. Keretrendszer kiválasztásakor a legfontosabb szempontok:

* **platformfüggetlen**: Az elkészített és futtatható programok működjenek ugyanúgy a különféle rendszereken. Lehetőleg csak megkerülhetetlen platformspecifikus megkötések legyenek (Pl.: Asztali környezeti tálca ikon). További előny ha a szerkesztő is platformfüggetlen használatra képes.
* **bővíthető**: A szerkesztő program alap funkcióit ki lehessen saját tartalmakkal, kódokkal, funkciókkal egészíteni. Előny, ha a kiegészítés teljesen vagy részlegesen kompatibilis más programokban használt kiegészítőkkel, tartalmakkal vagy támogatást biztosít ezek importálásához.
* **könnyen kezelhető**: A szoftverhez legyen támogatás, dokumentáció és frissítés, ami a könnyebb kezelhetőséget segíti. A szoftver kezelése könnyen elsajátítható legyen (hasonló programoktól kevés eltérő elem). A program lehetőleg „Out of Box” működjön vagyis beüzemelése ne igényeljen más külső, nem mellékelt keretrendszereket, extra beállításokat, illetve haladó konfigurációt (Pl.: önállóan futtatható legyen, hordozható program, ne legyen külső SDK letöltés, ne kelljen környezeti változókat, regisztereket beállítani stb.).

A piacon rengeteg rendelkezésre álló szoftver van, itt a legnépszerűbbeket néztem át. A következő keretrendszerek irányadóak lehetnek:

* **React**: Ingyenes és nyílt forráskódú JavaScript keretrendszer, kliens oldali alkalmazásokhoz. A Meta cég és a közösség tartja karban, elsősorban a felhasználói felület készítését segíti. Programozói tudás és fejlesztői környezet telepítése, beállítása szükséges.
* **Vue**: Ingyenes és nyílt forráskódú JavaScript keretrendszer, kliens oldali „egylapos” alkalmazásokhoz. Funkcióját tekintve hasonló a React-hez. Egy HTML oldalt generál és a rajta lévő elemeket interaktívan kezeli.
* **Flutter**: Felhasználó felület építő keretrendszer, azonban itt a közös kódot a fordító platformspecifikus natív kóddá alakítja, így a platformok között eltérés tapasztalható.

A fenti keretrendszerek speciális programozói tudást igényelnek, telepítésük, kód karbantartása, továbbadása és verziókezelése nehézkes lehet egy kezdő számára, így egyszerűbb keretrendszereket kerestem:

* **Wordpress**: Eredetileg blogger keretrendszernek készítették. Webszerverre történő telepítés után a felhasználó webes böngészőn keresztül testre szabhatja weboldalát grafikus kezelőfelületen. Kiegészítőkkel bővíthető, azonban a legnagyobb hátránya, hogy beépítve kevés jól működő alapfunkciója van. A keretrendszert használó pedig eltévedhet a rengeteg félkész, fizetős, kis funkcionalitást biztosító kiegészítők tengerében.

Bővíthetőség, könnyen kezelhetőség adott, de kizárólag webszerverre telepítve, webest tartalmakat generál, offline futtatható alkalmazás készítésére nem alkalmas és a kiegészítők összességében drágává tehetik.

Tovább bővítettem a keresést játékmotorokra is, mivel ezekkel a rendszerekkel olyan programokat lehet írni, amik gyorsan tudnak reagálni a felhasználói bevitelre, animációkat, rendszer funkciókat kezelnek és önállóan is futtathatóak.

* **Unity**, **Unreal Engine**, **Godot**: Ezeknek a motoroknak az alapelve, hogy stabil, jól bevált nyelveken, (C, C++, C#, Python stb.) fejlett funkciókat valósítanak meg, amelyeket a programozó felhasználhat. Mivel a felhasznált nyelvek legtöbb funkcióját kihasználja a motor, a fejlesztőnek is haladó tudással kell rendelkeznie az adott programozási nyelvről és motorról.
  + Előnyök: Nagy közösség, kiterjedt funkcionalitás, fejlett szerkesztő
  + Hátrányok: Használata speciális tudást igényel (programozási nyelv, függvények/motor ismerete)
* **Scratch**, **Wick Editor**: Oktatási célokra használják, nagyon könnyű megtanulni, illetve programozási alapismeretek nélkül is használható. Az elkészített program inkább csak a programozás tanulásához, bemutatásához használható. Valós környezetben szinte használhatatlan. Bővítési lehetőség nagyon nincs külsős programozók számára.
  + Előnyök: Bárki számára könnyen elsajátítható
  + Hátrányok: A készített program önállóan nem futtatható, korlátozott funkcionalitással rendelkezik, nem bővíthető
* **Construct 3**, **Game Maker: Studio**, **Gdevelop**: Ezen programok az előzőekben bemutatott programok ötvözete. Gyors tanulási lehetőséget biztosítanak vizuális programozási lehetőséggel (visual scripting), azonban a motorhoz való bővítést és hagyományos programozási nyelveket is támogatnak a haladó felhasználók részére.
  + Előnyök: Bárki számára könnyen elsajátítható, bővíthető
  + Hátrányok: kevésbé professzionális megoldások, esetleges sebességbeli korlátok

A kész program költségcsökkentése céljából olyan keretrendszerek közül választanék amelyek nem igényelnek speciális képzést és tudást, így a munkaerő költsége, illetve a kész program elkészítési ideje is csökkenthet. Végül a **Construct 3**, **Game Maker: Studio**, **Gdevelop** programokra szűkítem a keretrendszer választásomat, de az alábbiakban a lehető legjobb választási eredmény érdekében a programokat, egymással is összehasonlítom.

**Game Maker: Studio**:

C++-ban írt fizetős asztali játékprogram szerkesztő. Úgy a vizuális programozást (GML Visual) mint a szerkesztő saját programozási nyelvét is támogatja (GML Code). A szerkesztő saját C++ fordítót használ a kész program kódjának elkészítéséhez. Változatos funkciókat támogat ennek fizetős modellje.

**Gdevelop**:

Javascriptben írt webes szerkesztő. Az exportált program is JavaScriptben készül egy környezeti motorba beágyazva. A program teljesen ingyenes. Csak vizuális szerkesztés lehetséges. Kiegészítők készítésére is lehetőség van, azonban haladóbb, programozói szinten nem lehet bővíteni ezt a programot. A program felépítése kisség összefüggéstelen (nehéz átlátni mely objektumokból hogyan épül fel a program).

**Construct 3**:

Javascriptben írt fizetős webes szerkesztő program. Az exportált program is JavaScriptben készül egy környezeti motorba beágyazva. Vizuális programkódot és a JavaScript programozási nyelvet is támogatja. Kiegészítők készítésére is lehetőség van JavaScript nyelven. A nagy hangsúly van a generált kód optimalizálásán, modern API-okat, illetve technológiákat használ.

**Game Maker: Studio** – **Gdevelop** összehasonlítása:

„Game Maker: Studio” fejlettebb C++ motorral, jobb funkciókkal, megvalósítással és támogatással rendelkezik. Gdevelop viszont ingyenes és beüzemelése is egyszerűbb. „Game Maker: Studio” bővíthetősége jobb mint a Gdevelop programnak. Az egyszerűbb beüzemelés használat miatt, én mégis Gdevelopot részesítem előnyben.

|  | Game Maker: Studio | Gdevelop |
| --- | --- | --- |
| Funkciók | Fejlettebb | Kevésbé fejlett |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Fizetős (1550 Ft/hó) | Ingyenes |

**Construct 3** –  **Game Maker: Studio** összehasonlítása:

„Game Maker: Studio” első látásra fejlettebbnek tűnik, azonban Construct 3-ban az optimalizáció hangsúlyososabb (<https://www.construct.net/en/blogs/construct-official-blog-1/javascript-beats-gamemaker-1570>). Construct 3 egyszerűbben elérhető, könnyebben kezelhető, bővíthetősége rugalmasabb. Dokumentáció és SDK jobban elérhető Construct 3-hoz.

|  | Game Maker: Studio | Construct 3 |
| --- | --- | --- |
| Funkciók | Kevésbé fejlett | Fejlettebb |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Fizetős (1550 Ft/hó) | Fizetős (3240 Ft/hó) |

**Construct 3** – **Gdevelop** összehasonlítása:

Gdevelop programot a Construct 3 ingyenes kis testvérének nevezhetném. Mindkettő JavaScript kódot generál, visual scripting alapú a programlogika. Gdevelop ingyenességéből azonban több hátrány is fakad: funkciók implementáláshoz költséghatékonyabb és rosszabb minőségű programokkal rendelkezik (motor sebessége is elmarad a Construct 3-hoz képest). Construct 3-hoz minden keretrendszert egybecsomagoltak és beágyaztak, így nem szükséges további rendszerek telepítése.

|  | Gdevelop | Construct 3 |
| --- | --- | --- |
| Funkciók | Kevésbé fejlett | Fejlettebb |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Ingyenes | Fizetős (1550 Ft/hó) |

Személyes választásom a Construct 3-ra esett. Fizetős a program, de az általam támasztott követelményeknek legjobban ez felel meg. Kiegészítőkkel bővíthető, visual scripting segítségével könnyen használható, a generált JavaScript kód pedig önállóan futtatható és elég gyors.

## Construct 3 felépítése:

A szerkesztő az „editor.construct.net” oldalon elérhető, a keretrendszerhez kötődő további tartalmak (fórum, kiegészítő bolt, dokumentáció) pedig a „construct.net” címen találhatók. Keretrendszere **szerkesztő** (edit time) és **működési környezet**re (runtime) bontható. A szakdolgozat készítésekor a stabil verzió **r308.2**, új béta ciklus nem kezdődött.



### Szerkesztő:

JavaScript alapú szerkesztő, offline futtatható, némely exportálási funkciója opcionálisan használhat felhő szolgáltatást. Vizuálisan lehet szerkeszteni a projekteket.

### Működési környezet:

Natív HTML, CSS, JavaScript program, amit a projekt vezérlési logikája alapján generál a szerkesztő. Minden platformon a JavaScript kód kerül futtatásra a platformra szabott különböző „konténerekben” (Cordova, NW.js, WebView stb.)

# Szerkesztő

Az „editor.construct.net” oldal megnyitásakor a szerkesztő elkezdi letölteni és frissíteni a különböző forrásokat. A letöltés végeztével a program internet nélkül is használható, a felhő tárhelyhez, a felhő exportáláshoz és a licenckulcs frissítéséhez heti egyszer internetkapcsolat szükséges.

Frissítések heti rendszerességgel érkeznek: stabil verzió, gyors javítások amik alverzió formájában jelennek meg (pl.: 285.3). A szerkesztőt különböző verziókban, módokon lehet elindítani:

* Normál / stabil verzió: <https://editor.construct.net/>
  + Stabil kiadás az azt megelőző béta ciklus után (8-10 verzió) érkezik. A stabil verziók a béta frissítéséket is egyben tartalmazzák.
* Béta verzió: <https://editor.construct.net/beta>
  + Új funkciók, komplexebb javítások első megjelenése.
* Megadott verzió betöltése: https://editor.construct.net/rXYZ pl.: 308.2-es vezió → <https://editor.construct.net/r308-2>
  + Adott verzió betöltése, ez hosszabb távú projekteknél lehetővé teszi a hiba mentesebb fejlesztést és új verzióra való importálás halasztását.
* Biztonságos mód: <https://editor.construct.net/?safe-mode>
  + Minden “3rd party” (külső fél) kiegészítőjét kikapcsolja, hogy biztonságosan elindulhasson a szerkesztő ha egyik kiegészítő hibásan működött.

## Kezelőfelület:

Az első indításkor a kezdőképernyő fogadja a felhasználót, ahol hasznos linkeket, ajánlott minta projekteket és a legutóbb megnyitott projekteket ér el.

A felső sávon balra lévő “Menu” gomb állandó elem. Ennek segítségével hívható elő a szerkesztő összes funkciója. Új projekt megnyitásakor automatikusan létrehoz egy alapértelmezett felületet, amit teljesen testre szabhatunk. A testreszabás lényege, hogy minden elemre (mindent bezárva egy nagy elem marad) 5 fajta irányba illeszthetünk egy másik elemet: jobbra, balra, fent, lent, ráillesztés (ilyenkor kis lapfülek segítségével válthatunk az egymáson lévő elemek között). Mobilon a különböző ablakokat nem lehet testreszabni, oldali behúzással lehet váltogatni. (<https://www.construct.net/en/make-games/manuals/construct-3/overview/start-page>)

## Kiegészítők:

Ez a szerkesztő önmagában nem rendelkezik sok beépített funkcióval. A legtöbb dolgot kiegészítők segítségével lehet elkészíteni. A menüből elérhet kiegészítő kezelő mutatja a szerkesztőbe betöltött bővítményeket. Alapértelmezetten is betöltődik rengeteg kiegészítő, de ezeket a felhasználó is tudja bővíteni. Háromféle kiegészítő típus létezik: **bővítmény**, **viselkedés** és **effekt**. Opcionálisan lehetőség van a szerkesztő kinézetét is megváltoztatni, de ez csak a szerkesztőre vonatkozik, a kész program viselkedését semmilyen formában nem befolyásolja.



### Bővítmények:

Ezek a projekt építőelemei. Ezek a modulok bármilyen funkciót megvalósíthatnak. A bővítményeknek háromféle típusa létezik: látható, nem látható és példányosítható, nem látható és nem példányosítható.

* Látható bővítmények: Ezek a bővítmények az elrendezésen láthatóvá válnak és bármennyi példány készíthető belőlük. Például: Szöveg, Gomb, Videó
* Nem látható és példányosítható bővítmények: Olyan bővítmények, melyek az elrendezésen nem láthatóak (mert nincs értelmük), de különböző példányban létrehozhatjuk őket. Például: Adatszerkezetek (szótár, tömb, bináris tömb, json objektum, xml objektum)
* Nem látható és nem példányosítható bővítmények: Ilyen típusú bővítmények nem kerülnek megjelenítésre és nem is hozhatunk belőlük létre több másolatot. Ezeket olyan funkciók implementálására használják, ahol több példány létrehozása nem megoldható, nem vezet konzisztens eredményhez. Például: egér bevitel, érintés bevitel, asztali platform funkciók kezelése, tárhelykezelés

### Viselkedések:

Ezek bővítményekhez társítható kiegészítők. Olyan modulok, amelyek önmagukban nem értelmezhetőek (pl.: mozgás), de egy bővítmény példányára helyezve azt kezelni tudja.

### Effektek:

A viselkedésekhez hasonlóan önmagukban az effektek sem értelmezhetőek. Ezek kizárólag látható példányokra, rétegekre, elrendezésre helyezhetők. Kizárólag vizuális módosításokat hajtanak végre.

## Készített program felépítése:

Minden construct 3-as projekt ugyanolyan elemekből épül fel, ez nagyban megkönnyíti, hogy a különböző részeken, más szakképesítéssel rendelkező felhasználók gyorsan átlássák egymás munkáját is. Minden program egy “Elrendezés”-ekből áll (“Layout”). Ezek az elrendezések jelentik magát a felületet, amit a végfelhasználó látni fog. Ezekhez az elrendezésekhez egy vagy több “esemény lapot” (“Event sheet”) rendelünk, amelyek a vezérlést végzik. Az eseménylapok blokkokból épülnek fel, amelyek lehetnek: “Műveletek” (“Actions”), “Feltételek” (“Conditions”), “Függvények” (“Expressions”) (angol szavak rövidítve: “ACEs”).

* Művelet: a program “cselekvése”. A kiegészítők a program valamely végrehajtását funkcióját indítják el (pl.: írás, olvasás, áthelyezés, kiírás). Lehet szinkron, aszinkron, és várakoztató művelet is,
  + szinkron művelet: végrehajtása és befejezése azonnali
  + aszinkron művelet: a végrehajtás kezdése és befejezése nem történik meg azonnal
  + várakoztató művelet: a végrehajtásban következő blokkok, alblokkok várakozásra vannak kényszerítve, addig nem folytatódnak a blokkok végrehajtása, amíg várakozik a program.
* Feltételek: Legelső építő elem. Tartalmazhat más feltételeket, függvényeket és műveleteket. A feltételben lévő blokkok akkor fognak teljesülni, ha a feltétel igaz. Feltételeknek 3 fajtája van: Normál, Esemény, Ciklus
  + Normál feltétel: akkor lép életbe ha futtatáskor a feltétel igaz.
  + Esemény feltétel: ezt valamilyen kiváltó ok állítja aktiválja a kiváltás pillanatában (futtatási helytől függetlenül).
  + Ciklus feltétel: hasonló a normál feltételhez, de a feltétel újra indíthatja magát és a benne lévő blokkokat
* Függvény: kizárólag dinamikus értékmegadásra szolgál. Szöveget vagy számot eredményez.



Az eseménylap annyiszor fut le, ahányszor a kép frissítésre kerül a felhasználó számára (általában 60-szor). Az esemény feltételek külön aktiválódnak. Az események végrehajtási sorrendje fentről-lefelé majd kívülről-befelé (balról-jobbra) történik.

## Szerkesztő SDK:

### Fejlesztői mód:

Ahhoz, hogy bővítményeket tudjunk fejleszteni be kell kapcsolnunk a Construct fejlesztői módját. Ezt a beállítások ablak szélére történő sokszori kattintással, majd a megjelenő fejlesztői mód jelölőnégyzet bejelölésével tehetjük meg. Ilyenkor a kiegésző kezelő nem csak helyi fájlokból tudja betölteni a kiegészítőket, hanem URL címről is. A biztonsági házszabályok miatt csak tanúsítvánnyal ellátott és titkosított “localhost” címekről lehetséges a betöltés. Jelenleg egy electron környezetben megírt és futtatott, saját webszervert (EDS - Easy Development Server) használok a bővítmények betöltésére, de használható Xampp vagy egyéb ismert webszerver is. Fontos továbbá, hogy a szerveren és a HTTP fejlécben kapcsoljunk ki és tiltsunk le mindenféle gyorsítótárazást, mert a gyorsítótárazás bekapcsolásával az esetleges kiegészítő változtatások a szerkesztőben nem fognak érvénybe lépni. A szakdolgozat során a szabványos (dokumentáció által ajánlott) elnevezéseket használom, ez nagyban segíti a kiegészítők későbbi értelmezését.

### Környezet:

A különböző interfészek a “self.SDK” objektumon belül érhetőek el.

* Alap osztályok
  + Bővítményhez: IPluginBase
  + Viselkedéshez: IBehaviorBase
* Példány osztályok
  + Bővítményhez: IInstanceBase, IWorldInstanceBase
  + Viselkedéshez: IBehaviorInstanceBase
* Typus osztályok:
  + Bővítményhez: ITypeBase
  + Viselkedéshez: IBehaviorTypeBase
* Szerkesztőhöz vagy más objektumokhoz köthető metódusok, függvények és objektumok. Ezeket az objektumokat különböző függvények és esmények eredménye lehet:
  + Geometria: Color, Quad, Rect
  + Grafika: IDrawParams, IWebGLRenderer, IWebGLText, IWebGLTexture
  + Általános: ILang, IZipFile, IZipFileEntry
  + Model: IEventBlock, IEventParentRow, IEventSheet, ILayer, ILayout, IProject, IProjectFile
  + Objektum: IAnimation, IAnimationFrame, IBehaviorInstance, ICollisionPoly, IContainer, IFamily, IImagePoint, IObjectClass, IObjectInstance, IObjectType, IWorldInstance
  + UI: ILayoutView, Utilities
  + Kiegészítő keresés: C3SDK\_ListACEIDs

### Közös struktúra:

* **addon.json**: Ez írja le a kiegészítő alap adatait és struktúráját. Köztük szerepel a kiegészítő azonosítója, neve, leírása, típusa, készítője, verziója, elérhetőségek, és a hozzá tartozó fájlok listája, valamint a szerkesztő fájlok listája. (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/addon-metadata>)
* **lang/en-US.json**: A bővítmény nyelvi adatait tartalmazza mert bővítmény beállítások nem tartalmaznak nyelvi adatokat csak azonosítókat. A fájlokat a társítani kívánt nyelv ISO nyelv és ország kódjával kell elnevezni (pl.: hu-HU.json) (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/language-file>)

### Bővítmény mappa struktúra:

* **aces.json**: Ebben a fájlban van definiálva a szerkesztőben létező “Műveletek”-et (“Actions”), “Feltételek”-et (“Conditions”), “Függvények”-et (“Expressions”) (angol szavak rövidítve: “ACEs”) és minden kapcsolatos beállítást. (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/defining-aces>)
* **plugin.js**: bővítmény globális beállításai találhatóak. “IPluginBase” class-t kell örököltetni és a this.\_info-n keresztül “IPluginInfo” interfészt ahol a bővítmény tulajdonságait lehet beállítani (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/configuring-plugins>)
* **instance.js**: létrehozott bővítmény példány(ok)hoz tartozó szerkesztői programlogika és szerkesztő által végrehaktott események kezelésére szolgál. Ezek a bővítmény egyedi szerkesztői viselkedéséhez kell, amit a szerkesztő alapból nem támogat. Például: alakzat egyedi átméretezése.
* **type.js**: Típus azonosításához szükséges. Általában minden bővítménynél hasonló a felépítése.

### Viselkedés mappa struktúra:

* **aces.json**: ugyan olyan szerepe van mint fentebb kifejtett aces.json fájlnak.
* **behavior..js**: bővítmény globális beállításai találhatóak. “IBehaviorBase” class-t kell örököltetni és a this.\_info-n keresztül “IBehaviorInfo” interfészt ahol a viselkedés tulajdonságait lehet beállítani https://www.construct.net/en/make-games/manuals/addon-sdk/guide/configuring-behaviors)
* **instance.js**: létrehozott viselkedés példány(ok)hoz tartozó szerkesztői programlogika és szerkesztő által végrehaktott események kezelésére szolgál. Ezek a viselkedés egyedi szerkesztői logikájához kell, amit a szerkesztő alapból nem támogat. Például: alakzat egyedi átméretezése.
* **type.js**: Típus azonosításához szükséges. Általában minden viselkedésnél hasonló a felépítése.

### Effekt mappa struktúra:

* **addon.json**: az alap kiegészítő beállítás mellett további beállításokat is definiálni kell (animáció, mélység kezelés, rajzolás stb), mivel további effekttel kapcsolatos JavaScript kód nem kerül betöltésre
* **effect.fx**: Az effektek GLSL (OpenGL Shading Language) nyelvben történnek. Mivel kizárólag rajzolási feladatot látnak el ezért ezt az erre specializálódott nyelvben kell megírni. Ezek a kódok már videófeldolgónak kerülnek átadásra így jobban fog a program futni. Effekt programkódja a szerkesztőben is lefut hogy a felhasználó szerkesztés közben is lássa az eredményt. (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/configuring-effects>)

# Futási környezet

## Tömörítés:

Szerkesztő végleges programkód készítésekor (exportáláskor) a Google Closure Compiler könyvtárat használja. Ez a program a futtatható kész program forráskódját elemzi, eltávolítja a felesleges nem használt kód részleteket és újraírja a maradék kódot rövidebb, gyorsabb formába. Továbbá elemzi szintaxis hibákat, változó referenciákat, típusokat és alapvető JavaScript hibákat is feltárja (<https://developers.google.com/closure/compiler>). A tömörítésnek három szintje lehetséges: Nincs, Egyszerű, Haladó

### Nincs:

Ilyenkor a programkódokat nem tömöríti a program. Nem okoz plusz problémát a kiegészítők működésében.

### Egyszerű:

A programkódból minden egyértelműen felesleges dolgot eltávolít például szóközöket, tabulátorokat, sortöréseket és függvényeken belül az átnevezhető változókat átnevezi rövidebb formátumra. Általában nincs hatással a bővítményre.

### Haladó:

A programkódot teljesen analizálj, nem használt kódokat eltávolítja, rövidebb érték megadásokat, egyszerűsítéseket készít, illetve minden változó nevét lecseréli egy rövidebb névre. Ha a programkód nem megfelelően van felkészítve a tömörítésre, akkor a külső üzenetekre, keretrendszerek utasításaira való hivatkozás törlődik vagy megváltozik így a kész program futása során a bővítmény hibásan vagy nem megfelelően fog működni.

Tömörítés előtt:

const obj = {

apples: 1,

oranges: 2

};

console.log(obj.apples, obj.oranges);

⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟

const obj = {

a: 1,

b: 2

};

console.log(obj.a, obj.b);

Fő szabály hogy kerülni kell tulajdonság megadás módjának a keverését mert a string stílusú értékmegadást a tömörítő figyelmen kívül fogja hagyni:

const obj = {

"apples": 1

};

console.log(obj.apples);

⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟⮟

const obj = {

"apples": 1

};

console.log(obj.a);

Látszik hogy az “obj.a” nem létezik a tömörítés miatt.

Külső függvényeknél, üzeneteknél a string meghívást alkalmazzuk:

externalLibrary["doSomethingUseful"]();

Globális eléréshez használjunk “globalThis” kulcsszót:

globalThis["myGlobalFunction"]();

[(https://www.construct.net/en/make-games/manuals/construct-3/scripting/guides/advanced-minification](https://www.construct.net/en/make-games/manuals/construct-3/scripting/guides/advanced-minification))

## Futási SDK:

### Környezet:

### Mappa struktúra:

# Kiegészítők

## Szükséges kiegészítők:

# Tömeges email küldő elkészítése

Lorem Ipsum, dolor sit amet

# Irodalomjegyzék

[A felhasznált szakirodalom megadása]

[A szerzők nevét mindenütt “Családnév, X.” formában kell megadni, ahol X. a szerző keresztnevének (keresztneveinek) kezdőbetűje. Magyar cikk esetén a vessző a családnév és a keresztnév kezdőbetűje közt elhagyható. Ha az egyértelműség megkívánja, a keresztnév kiírható teljesen is.

A dolgozat szerzője szabadon választhat, az A vagy a B típust használja.]

**[A-típus:**

A cikkekre való hivatkozás egy []-be írt sorszámmal történik. A sorszámozást folytonosan kell megtenni, a sorba rendezés alapja az első szerző családneve.]

[1] Szerző1 (, Szerző2 ...): *Cikk címe*

Folyóirat neve, sorszám, kezdőoldal-végoldal, év.

[2] Szerző1 (, Szerző2 ...): *Konferencia-kiadvány-beli cikk címe*

„Konferenciakiadvány:” Konferencia neve, hely, kezdőoldal-végoldal, év.

[3] Szerző1 (, Szerző2 ...): *Könyvcím*

„Könyv:” Kiadó, hely, oldalszám, év.

[4] Szerző1 (, Szerző2 ...): *Kutatási jelentés címe (csak publikus elérhető jelentés!)*

„Kutatási jelentés”: Kutatási projekt neve, intézet, oldal, év.

[5] Szerző: *Disszertáció címe*

„PhD/kandidátusi/stb. disszertáció”: Egyetem, kar neve, év.

[6] *Internetes oldal elnevezése:*

*URL, letöltés ideje (csak konkrét dokumentumra mutató URL adható meg!)*

**[B-típus:**

A hivatkozás a szerzők családi nevéből és a kiadás évéből képezett azonosítóval történik. Kettőnél több szerző esetén az „et.al.” rövidítés használható. Pl. „[Vijayasundaram, 1986.]”, „[Meister and Sonar, 1998.]”, „[Felcman et.al., 1994.]”

Abban a ritka esetben, amikor több cikknek is azonos azonosító jutna (megegyeznek a szerzők és a kiadás éve) az év után az „a”, „b”, „c”, stb. betűk csatolandók, pl. „[Stone and Norman, 1993a.]”.

A sorba rendezés alapja a szerzők családneve, végül az év.

Ezt leszámítva a formátum ugyanaz, mint az A-típus esetén, de ekkor a hivatkozási sorszám lehagyható.]

# Mellékletek

1. [A dolgozat mellékletei, ha vannak]