[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 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, 2022.09.30 …………………………….

hallgató

# Kivonat

Webalkalmazás fejlesztése visual scripting segítségével

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 játékhoz, 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

Web application development with visual scripting language

The Construct 3 is a web based visual scripting framework which is developed for games. Nowadays most web applications are similar to computer games in graphics, response time and user input. The question is whether this framework can be used to build general purpose web applications with custom made plugins. The starting overwork can pay off if less professional workers can work on complex projects.

# Tartalomjegyzék

[**Nyilatkozat**](#_k1j7xw5xto0p) **4**

[**Kivonat**](#_o3pg2olkdu69) **5**

[**Abstract**](#_b2vvf2yvez3l) **6**

[**Tartalomjegyzék**](#_9aoapsjwh6p5) **1**

[**Bevezetés**](#_cgzg9wnxn5k) **1**

[Témaválasztás indoklása:](#_ht5770aingsc) 1

[Megrendelő problémája](#_iw62k5azmmp0) 1

[Fejlesztő problémája](#_dnaq38lyhwip) 1

[Környezet kiválasztása](#_qjxporec7zkh) 2

[Példa probléma](#_97qb6khpvas) 5

[**Szerkesztő bemutatása**](#_m4n5yksiwj3g) **6**

[Telepítés](#_gwp0mgjvyx21) 6

[Példa bemutató](#_ue25a6ef6yy6) 7

[Kezelőfelület:](#_5g69jyu8g6mp) 7

[Kiegészítők:](#_ovtpuuib1016) 7

[Bővítmények:](#_3hz53fffuxtr) 8

[Viselkedések:](#_yt4nayqp0nha) 9

[Effektek:](#_xo6f5tbfgfhd) 9

[Az elkészített program felépítése](#_atbfl5zg4h50) 9

[Szerkesztő SDK:](#_2vsfz4yoslfj) 10

[Fejlesztői mód:](#_vs39b1pir0i3) 10

[Környezet:](#_9miai889ddxc) 11

[Közös mappa struktúra:](#_ju62kyi6p6g1) 11

[Bővítmény mappa struktúra:](#_l9o8ws369ej5) 12

[Viselkedés mappa struktúra:](#_gguka9obqal8) 12

[Effekt mappa struktúra:](#_y4s0g5mg33h1) 12

[**Tömeges email küldő**](#_oaslzsxa1hi4) **13**

[**Bővítmény készítés**](#_u8w6542dzn10) **13**

[Tömörítés:](#_29zzmwkoxsap) 13

[Nincs:](#_fkubjubmwmy5) 13

[Egyszerű:](#_vjchx2dlun3u) 13

[Haladó:](#_fvnzdyn1arh) 13

[SDK:](#_h5qr8yx8nqln) 14

[Környezet:](#_nknxr270rkfa) 14

[Közös mappa struktúra:](#_ebg6th5hejn3) 15

[Bővítmény mappa struktúra:](#_3eo7wo9s0tqr) 15

[Viselkedés mappa struktúra:](#_s8ph3pv4y2tb) 16

[Hiányzó funkciók:](#_jfk7is53oemk) 16

[**Kiegészítők elkészítése**](#_11pug8uiw87e) **17**

[Általános szabályok:](#_eg1zn96pq4mi) 17

[Keretrendszer:](#_tes0jnwm4jl) 17

[Kódolás:](#_99j2dc9whdhj) 17

[Text Manager](#_lwy8yu6ehhk2) 17

[Bővítmény:](#_9nq7cz7r2y6k) 17

[Szerkesztő rész:](#_rg0snyn5gn9k) 18

[Futtatási rész:](#_lq7dkynd4dj4) 19

[Viselkedés:](#_qjj48cg5ob2y) 19

[Szerkesztő rész:](#_8gf2fib8la20) 19

[Crypto](#_7sx5mccihfzk) 19

[Szerkesztő rész:](#_kiysuf28fwlu) 19

[Futtatási rész:](#_s271o6okffi5) 20

[Desktop](#_tmtmdcgewd3b) 20

[Email](#_27f0cantvks) 20

[File Manager](#_kcom6nas414u) 20

[Notification](#_caxzddfwlcsi) 20

[**Tömeges email küldő elkészítése**](#_600ec8rjagjk) **20**

[**Irodalomjegyzék**](#_1ac8w2pcohjb) **21**

[**Mellékletek**](#_4cdc7r4caisp) **23**

[**Jegyzet**](#_km65kmkwnxxe) **23**

# Bevezetés

## Témaválasztás indoklása:

Munkám során a megrendelők egyedi programokat rendelnek, azonban szinte közvetlenül azok átadását követően 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 akkor is, ha nem jártas az adott programozási nyelvben. A szakdolgozat egy problémán keresztül mutatja be az elképzelésem megvalósítását. A cél, hogy az általam létrehozott szerkesztő környezet támogatni tudja a felhasználók kreativitását. Eszközt szeretnék adni az embereknek, ahhoz, hogy saját programokat tudjanak létrehozni és a megírt programot hatékonyan tudják csapatban, másokkal együtt továbbfejleszteni.

## Megrendelő problémája

A szoftverfejlesztés egyik legnagyobb nehézsége, hogy a megrendelőnek egyedi igényei lesznek. A szoftvert a feladatra testre kell szabni és az informatika környezetbe be kell illeszteni de nagy költségeket se lehet mindig kifizetni (főleg ha csak prototípus). Egyedi szoftverfejlesztés költségei nagyon magasak már a kezdetekkor. Külön szakembert kell felbérelni és az esetleges további módosításokra is a szakemberre leszünk utalva. Ha prototípus elkészítése valamilyen módon nem sikerül az nagy veszteségeket okozhat mert a prototípus gyártás során is szakembereket kellett fizetnünk és elkészült program se keletkezett.

## Fejlesztő problémája

Problémát jelent a fejlesztések költségének fékentartása. Költségek nagy részét a munkaerő bére teszi ki. Gazdaságosabb a megrendelt programot nem teljesen alapjaitól megírni, érdemes kiválasztani egy keretrendszert (sablont), hogy a program készítő ezáltal kevesebb idő alatt készítsen el egy programot. 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, internetes app áruház integráció stb.). További előny ha a szerkesztő is platformfüggetlenül használható.
* **bővíthető**: A szerkesztő program alapvető funkcióit ki lehessen egészíteni saját tartalmakkal, kódokkal. 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 későbbi munkát 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 összetett konfigurációt (Pl.: önállóan futtatható legyen, hordozható program, ne legyen külső SDK letöltés, ne kelljen környezeti változókat, regisztereket beállítani stb.).

## Környezet kiválasztása

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 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 választok amelyek nem igényelnek különleges 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, hogy 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. Nagy hangsúly van a generált kód optimalizálásán, modern API-okat, illetve technológiákat használ.

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

„Game Maker: Studio” fejlettebb C++-ban írt 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 a 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 hangsúlyososabb1. 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:

A Gdevelop programot a Construct 3 ingyenes kis testvérének nevezhetném. Mindkettő JavaScript kódot generál, visual scripting alapú a programlogika. A Gdevelop ingyenességéből azonban több hátrány is fakad: funkciók implementáláshoz költséghatékonyabb és rosszabb minőségű programokkal rendelkezik (motor sebessége is elmarad a Construct 3-hoz képest). A 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.

## Példa probléma

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 megrendelők egy olyan szoftveres megoldást szeretnének amivel 1000-2000 email küldhető. Külön szervert és informatikust nem tudnak biztosítani.

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 megrendelők inkább egy számítógépén futó programot képzelnek el mint az Office, Chrome, VLC stb. 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: tömeges levélküldés, csak „3rd party” (külsős) kiegészítővel lehetséges, ennek nehézkes a beüzemelése (kiegészítők megbízhatósága kérdéses)
* Sendblaster:

Delivery Tech Corp. által fejlesztett Windows-os szoftver (kinézetre .NET, WPF applikáció), a fejlesztő által „befejezett” szoftvernek látszik. 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)

Megrendelő a “Sendblaster” szoftverhez hasonló programot szeretne.

Cél hogy a szerkesztővel készített program a megrendelő egyedi igényeit megvalósítsa és a fejlesztő gazdaságosan, fenntarthatóan tudja elkészíteni a programot.

# Szerkesztő bemutatása

## Bemutató

### Telepítés:

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 felhőbe történő exportáláshoz és a licenckulcs frissítéséhez heti egyszer internetkapcsolat szükséges.

Frissítések heti rendszerességgel érkeznek: stabil verzió, gyors javítások amik alverziók 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 elindulhasson a szerkesztő ha valamelyik 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 menüből nyolc különböző ablakot lehet előhívni, ezekben az ablakokban különböző információkat találhatunk szerkesztés közben2.



Mindent ablakot törölve egy középső elem marad. Az különböző ablakok lebegve szabadon rendezhetőek, azonban dokkolhatóak is egy másik ablak oldalához vagy összeilleszthetőek. Összeillesztéskor kis lapfülek segítségével válthatunk az egymáson lévő ablakok között.



Áthelyezés helyét jelzi a program.



### Példa program:

Téglalap területét kiszámító projekt keretében mutatom be egy program készítésének fontosabb fázisait.

Új projekt létrehozásához a Menu (Menü) > Project (Projekt) > New (Új) gombot kell használni. Projekt létrehozásakor kezdeti beállításokat adhatunk meg amik teljes mértékben módosíthatóak később.

Projekt ablakba láthatjuk a projekt felépítését.

* Layouts (Elrendezések) mappa tartalmazza a különböző felületeket ez lesz maga a “pálya” amit lát a felhasználó.
* Event sheets (esemény lapok) az elrendezés logikáját/viselkedést tartalmazza
* Object types (Objektum típusok) mappába kerülnek a kiegészítők.

Meg szeretnénk jeleníteni három szövegfoboztamit hozzáadhatunk: Object Types (Objektum típusok) > jobb kattintás > Add new object type (Új objektum hozzáadása) > Text Input (Szöveg bevitel) > Insert (Beillesztés)

Ha három szövegbevitelt beillesztettük ezzel a módszerrel, majd ezeket behúzással elhelyezhetjük az elrendezésen. Rendezzük el tetszés szerint a szövegdobozokat.

Ha rákattintunk akkor kijelöli a program és a Properties (Tulajdonságok) ablakon megjelenik a kijelölt elem tulajdonságai.

Lefelé görgetve a Properties (Tulajdonságok) almenüben a Placeholder (Helyőrző) értékét szerkesszük, hogy legyen egy “Hosszúság”, “Szélesség”, “Eredmény” beviteli mezőnk.

Nyissuk meg a Event sheet 1 (Esemény lap 1) esemény lapot. Most elkészítjük, hogy beírt szöveg esetén számolja ki a program a területet és jelenítse is meg.

Egyik szerkesztés figyelése: add event (esemény hozzáadása) > TextInput > On text change > Next

Vagy blokk készítése azért ha az egyik VAGY másik szövegdoboz szerkesztésekor is számoljon területet a program: Előzőleg hozzáadott eseményen bal szélére jobb klikk > Make OR blokk

Másik szerkesztés figyeléséhez előzőleg hozzáadott eseményen bal szélére jobb klikk > Add > Add another condition > TextInput2 > On text change > Next

Számolás elvégzése előtt elvégzése során ellenőrizzük a bevitelt hogy szám-e. Jelöljük ki a feltétel szélét > jobb klikk > Add (Hozzáadás) > Add blank sub-event (Üres al-esemény hozzáadása)

Megjelenő üres mezőbe kattintsunk duplán > System (Rendszer) > Is value type (Ha érték típus)

A megpróbáljuk számmá konvertálni a TextInput értékét, ezt a float(TextInput.Text) függvénnyel tudjuk megtenni. A Type (Típus) értéket állítsuk Number (Szám) értékre. Így ha sikerült a konverzió akkor a feltétel is igaz lesz.

Ugyan így adjuk hozzá a blokkon belül egy másik ilyen feltételt. Bal szélére jobb klikk > Add > Add another condition > System (Rendszer) > Is value type (Ha érték típus). Hasonlóan a float függvényt kell meghívni csak a TextInpu2.Text-re. (float(TextInput2.Text))

Maga a számítást egyből kiírjuk a TextInput3 mezőbe. Legutóbb létrehozott feltételek mellett Add action > TextInput3 > Set text (Szöveg beállítása)

A TextInput és TexInput2 értékét átkonvertáljuk összeszorozzuk: float(TextInput.Text) \* float(TextInput2.Text)

Elkészült projekt kipróbálható a Menu (Menü) melletti lejátszás gombbal vagy Menu (Menü) > Project (Projekt) > Preview (Előnézet)

Asztali exportáláshoz menjünk a: Menu (Menü) > Project (Projekt) > Export (Exportálás) > NW.js

A felugró ablakon hagyhatjuk az alapértelmezett beállítást de kisebb méret érdekében a Minify mode (Tömörítési módot) állíthatjuk advance (haladó) értékre.

Következő ablakban az operációs rendszert választhatjuk ki.

Ezután a programot elkészíti a szerkesztő. Exportálás után letölthető és futtatható a program bármilyen más program nélkül.

## Kiegészítők:

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 **szerkesztőre** (edit time) és **futtatókörnyezet**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.

Futtatókörnyezet:

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.)

Ez a szerkesztő önmagában nem rendelkezik sok beépített funkcióval. A legtöbb dolgot kiegészítők segítségével lehet elkészíteni. A menüből elérhető 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 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.

## Az elkészített program felépítése

Minden construct 3-as projekt ugyanolyan elemekből épül fel, ami nagyban megkönnyíti, hogy a különböző részeken, más szakképesítéssel rendelkező felhasználók gyorsan átlássák egymás munkáját is. Minden program “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). Végrehajtási mód szerint a következő lehetőségeink vannak:
  + 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 (általában másodpercenként 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 webhelyekről is. A biztonsági szabályok 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 fejlécekben 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
* Típus 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 mappa 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 egyedi funkciók kezelésére szolgál. Ezek a bővítmény egyedi szerkesztői funkcióihoz kell, amit a szerkesztő alapból nem támogat. Például: példány 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 egyedi funkciók kezelésére szolgál. Ezek a viselkedés egyedi szerkesztői funkcióihoz kell, amit a szerkesztő alapból nem támogat. Például: példány mozgatása.
* **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 videófeldolgozó résznek kerülnek átadásra így jobban fog a program futni. Effekt programkódja a szerkesztőben is lefut így a felhasználó szerkesztés közben is látja az eredményt. (<https://www.construct.net/en/make-games/manuals/addon-sdk/guide/configuring-effects>)

# Tömeges email küldő

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 nehézségbe 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.

# Bővítmény készítés

## Tömörítés:

A szerkesztő a 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 a szintaktikai hibákat, változó referenciákat, típusokat és az 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))

## SDK:

### Környezet:

Ez a környezet már a program futásakor fog életbe lépni. Nem azonos a szerkesztői környezettel. Más SDK interfészek érhetőek el. Itt már dinamikusan lehet csak lekérni az objektumokat, self.C3 objektumban érthetőek el az örökölhető osztályok. Az bővítmények hatóköre csak saját magukra, illetve az átadott más bővítmények interfészeire terjed ki.



A futási környezet nem éri el a hagyományos (DOM) JavaScript interfészeket mert egy külön szálon (worker-en) fut ezzel simább program futást ér el. Ha az eszköz a külön szálon történő futtatást nem támogatja, akkor a futtatási környezet is hozzáfér a hagyományos interfészekhez de ennek kihasználása kompatibilitási problémát okoz (milyen módba, hogyan fusson), helyette az SDK által támogatott kommunikációt lehet használni. Ablakhoz való (DOM) hozzáférést csak a bővítmények támogatnak.

### Közös mappa struktúra:

* **c3runtime/actions.js**: A műveletek függvényeit (azok a függvények amik végrehajtják a konkrét műveletet egy hívással) állítja be. Itt a szkript célja mindig ugyan az: egy objektumba összegyűjteni a függvényeket és “C3.Plugins.<kiegészítő azonosítója>.Acts” helyre beilleszteni. Fontos hogy a függvények neveinek egyeznie kell az aces.json-ban megadott függvény nevekkel.
* **c3runtime/conditions.js**: A feltételek függvényeit (azon a függvények amik végrehajtják a konkrét feltételt egy hívással) állítja be. Itt a szkript célja mindig ugyan az: egy objektumba összegyűjteni a függvényeket és “C3.Plugins.<kiegészítő azonosítója>.Cnds” helyre beilleszteni. Fontos hogy a függvények neveinek egyeznie kell az aces.json-ban megadott függvény nevekkel. Az aktiválás (Trigger) típusú függvények futás közben egy külön függvénnyel (this.Trigger) kerülnek meghívásra és csak igaz visszatérés esetén fognak lefutni a további logikai műveletek.
* **c3runtime/expressions.js**: A függvények függvényeit (azon a függvények amik végrehajtják a konkrét műveletet egy hívással) állítja be. Itt a szkript célja mindig ugyan az: egy objektumba összegyűjteni a függvények függvényeit és “C3.Plugins.<kiegészítő azonosítója>.Exps” helyre beilleszteni. Fontos hogy a függvények neveinek egyeznie kell az aces.json-ban megadott függvény nevekkel.

### Bővítmény mappa struktúra:

* **c3runtime/plugin.js**: Bővítmény alap osztályát hozzuk itt létre. Mindenképpen örököltetni kell az SDKPluginBase vagy SDKDOMPluginBase osztályt és a “C3.Plugins.<saját bővítmény azonosítója>” helyre kell beilleszteni az így létrejött új osztályt.
* **c3runtime/type.js**: Bővítmény típus osztályának megadását kell itt elvégezni. örököltetni kell az SDKTypeBaseés a “C3.Plugins.<saját bővítmény azonosítója>.Type” helyre kell beilleszteni az így létrejött új osztályt.
* **c3runtime/instance.js**: Bővítmény példányainak kódja. SDKInstanceBase, SDKWorldInstanceBase és SDKDOMInstanceBase osztályokból kell örököltetni a példányosztályt a C3.Plugins.<saját bővítmény azonosítója>.Instance helyre.
* **c3runtime/domSide.js**: Ez kód már a ablak (DOM) oldalon fog lefutni. self.RuntimeInterface.AddDOMHandlerClass(HANDLER\_CLASS); függvényben kell átadni az DOMHandler és DOMElementHandler osztályokból örököltetett új osztályt. Az kommunikáció a tényleges futtatási környezettel üzenetekkel és üzenet figyelőkkel történik.

### Viselkedés mappa struktúra:

* **c3runtime/behavior.js**: Viselkedés alap osztályát hozzuk itt létre. Mindenképpen örököltetni kell az SDKBehaviorBase osztályt és a C3.Behaviors.<saját bővítmény azonosítója> helyre kell beilleszteni az így létrejött új osztályt.
* **c3runtime/type.js**: Viselkedés típus osztályának megadását kell itt elvégezni. örököltetni kell az SDKBehaviorTypeBase és a C3.Behaviors.<saját bővítmény azonosítója>.Type helyre kell beilleszteni az így létrejött új osztályt.
* **c3runtime/instance.js**: Bővítmény példányainak kódja. SDKBehaviorInstanceBase osztályból kell örököltetni a példányosztályt a C3.Behaviors.<saját bővítmény azonosítója>.Instance helyre.

## Hiányzó funkciók:

* Nyelvek kezelése: különböző nyelvek közötti váltás eléggé nehézkesen megoldott a szerkesztőben. Megkerülő megoldások vagy hiányos és fiztős kiegészítőkkel kezelhető a probléma. Célom hogy a nyelvek kezelése könnyen kezelhető legyen.
* Titkosítás: A titkosítás különösen fontos ha jelszavakat vagy érzékeny adatokat kezelünk. A construct nem ad beépített lehetőséget az adatok titkosítására vagy ellenőrző összeg generálására.
* Asztali környezet: Az asztali exportálás NW.js környezetbe történik, azonban ennek a keretrendszernek jobb alternatívái is léteznek. Az Electron jobban karbantartott, több fejlesztő által fejlesztett, modernebb és gyorsabb keretrendszer amit a Construct 3 nem támogat.
* Levélküldés: Email levelek küldésére nem található bővítmény, az elkészített program egyik fő feladata pedig ez.
* Fájlkezelés: A szerkesztő csak az NW.js bővítményen keresztül biztosít fájlkezelést az is csak asztali exportáláshoz. Célom egy platformfüggetlen fájlkezelő létrehozása. A különböző eszközökön biztonsági vagy más okokból eltérő fájlkezelést is kezelné a bővítmény a lehető legegyszerűbb használat érdekében.
* Értesítések: Construct-ban nincs lehetőség távoli vagy helyi értesítések küldésére, ami értesíti a felhasználót a háttérben futó program változásáról. A bővítmény különböző platformok eltérő értesítés kezeléséhez is igazodna.

# Kiegészítők elkészítése

## Általános szabályok:

### Keretrendszer:

* Tömörítés támogatása
* Worker mód támogatása
* Minden platform támogatása (legalább hiba nélküli indítás)
* Felhő exportálás támogatása
* Lehető legkevesebb nem dokumentált funkció használata (használat esetén kiemelendő).
* WebGL 1 támogatás

### Kódolás:

* Szigorú mód bekapcsolása: “use strict”
* Hozzáférés (scope) korlátozása kapcsos zárójelekkel: {}
* Korlátozott hozzáférés: const, let használata (függvények is változóba)
* Osztály metódusok elnevezése szóközök nélkül, szavak nagy kezdőbetűvel: OsztalyMetodus
* Belső, saját használatú osztály metódusok alsó kötőjel kezdéssel: \_OsztalyMetodus
* Javascript metódusok nevénél első betű kicsi, egybe, szavak nagy kezdőbetűvel: peldaValtozo
* Hordozható objektumok kulcsnevei kisbetűs szavanként, kötőjellel elválasztva: pelda-valtozo
* Üznetek objektumok, nem szabványos metódusok stringként való meghívása: uzenet[“pelda-valtozo”], fuggveny[“kulsoFuggveny”]
* Nem tisztázott esetekhez Google Stílus Útmutató-ja az irányadó (<https://google.github.io/styleguide/jsguide.html>)

## Text Manager

Különböző nyelvek betöltéséhez a legalapvetőbb módszer, hogy a szöveget egy kulcs-érték párban tároljuk és megjelenítéskor az adatok csak kiolvassuk. Ez sok szövegnél nehézkesen kezelhető, a különböző programrészek nehezen kapcsolhatóak össze. Részleges megoldást kínálnak a külső felek által készített kiegészítők (<https://www.construct.net/en/game-assets/addons/i18n-translation-plugin-1931>, <https://www.construct.net/en/game-assets/addons/pro-ui-38>). Ezek kiegészítők kevés rugalmasságot biztosítanak és nyelv felismerés és kezelés nincs implementálva (kizárólag a betöltést könnyítik meg). Saját kiegészítőm szöveget, nyelvet és paraméterezést is kezelni fog, továbbá a formátumok közötti konvertálást is lehetővé teszi.

### Bővítmény:

Bővítmény szerepe az program és az eszköz számára elérhető nyelvek kezelése, nyelv automatikus választása, beállítás mentése és nyelvi adatok dinamikus kezelése. Adatok következő formátumba lehetnek:

* JSON (single): JSON fájl ahol végső a érték a fordítás szövege. Betöltéskor meg kell adni a forrás nyelvét.
* JSON (multiple): JSON fájl ahol a végső érték egy olyan objektum ahol nyelvenként fel van sorolva a különböző fordítások.
* CSV: szabványos vesszővel elválasztott értékek fájl ahol a fejléc (első sor) azonosítja a különböző nyelvek fordításait.
* Construct 3 szótár: szerkesztőben létrehozott szótár fájl. Betöltéskor meg kell adni a forrás nyelvét.
* Construct 3 tömb: szerkesztőben létrehozott tömb fájl ahol a fejléc (első sor) azonosítja a különböző nyelvek fordításait.

#### Szerkesztő rész:

Műveletek, feltételek és függvények csoportosítás:

* Nyelvi elemek:
  + Műveletek:
    - Nyelv hozzáadása
    - Nyelv beállítása
    - Nyelv törlése
  + Feltételek:
    - Támogatott nyelv ellenőrzése
    - Nyelvváltás
    - Nyelvváltás eszközön
  + Függvények:
    - Eszköz nyelv lista
    - Projekt nyelv listája
    - Ajánlott nyelv
    - Nyelvi kódok lekérése
* Nyelvi adat elemek:
  + Műveletek:
    - Adatok betöltése
    - Adatok törlése a memóriából
  + Feltételek:
    - Adatok sikeres betöltése
    - Adatok sikertelen betöltése
  + Függvények:
    - Betöltött adatok listája
    - Betöltött címkék listája
    - Betöltés állapota
    - Betöltött adat lekérése (fordítás)
* Paraméter elemek:
  + Feltételek:
    - Paraméter szintaxis
  + Függvények:
    - Escaping
    - Paraméterek listája
    - Paraméterek száma
    - Paraméterek kezdeti és végpontjai
    - Számozott és szöveges paraméterek behelyettesítése

Szerkesztőben beállítható tulajdonságokat is tartalmaz a bővítmény:

* Nyelvhez kapcsolódó részek: beállítható hogy a program indulásakor milyen nyelvek a támogatottak, melyik legyen az alértelmezett nyelv és a beállítások megmaradjanak-e a program bezárása után. Ezek futási időben is beállíthatóak de a legtöbb program esetében ezek statikus adatok így hasznos hogy szerkesztéskor ezek már beállíthatóak.
* Importáláshoz kapcsolódó részek: Importálni futási időben is lehet de a legtöbb program esetében ez statikus lokális fájl amit már szerkesztéskor megadhatunk.
* Konvertáló rész: Nem befolyásolja a elkészített programot de a nyelvi adatok kezelését megkönnyíti azzal, hogy különböző felépítésű struktúrákba képes konvertálni azokat.

addon.js, type.js fájlokat az alapértelmezett értékek szerint hagyom, csak a saját adatokat módosítom. plugin.js fáljba alapértelmezetteken felül beállítom hogy egypéldányos globális bővítmény legyen.

#### Futtatási rész:

### Viselkedés:

A különböző objektumok fordításához (szöveg, alakzat, gomb stb) a bővítménynek hozzá kell férni az objektumokhoz. Ezt a hozzáférést viselkedésen keresztül valósítom meg. A bővítmény rész a szövegkezelés központja lesz, ha hiányzik a projektből a viselkedés része se fog működni.

#### Szerkesztő rész:

Egy objektumhoz egyszer társítható a viselkedés az ütközések elkerülése miatt. Beállítható hogy be legyen-e kapcsolva a bővítmény és milyen kulcs szerint történjen a fordítás. Opcionálisan beállítható még, hogy milyen nyelvre legyen fordítva az alapértelmezett nyelv helyett.

## Crypto

A kiegészítő feladata különböző titkosítási algoritmusok alapján titkosítani adatokat, továbbá titkosítást kiegészítő műveletek elvégzése (ellenőrző összeg számítása, kulcsgenerálás stb.). Nagyvonalakban a JavaScript kriptográfiai könyvtárát illesztem a Construct 3-hoz (<https://developer.mozilla.org/en-US/docs/Web/API/Crypto>).

### Szerkesztő rész:

### Futtatási rész:

## Desktop

## Email

## File Manager

## Notification

# Tömeges email küldő elkészítése

Lorem Ipsum, dolor sit amet

# Irodalomjegyzék

[1] How JavaScript beats GameMaker Language (GML)

<https://www.construct.net/en/blogs/construct-official-blog-1/javascript-beats-gamemaker-1570>, 2022.10.18 17:22

[2] Construct’s interface

<https://www.construct.net/en/make-games/manuals/construct-3/overview/the-interface>

[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]

# Jegyzet

* Téma igazítása: fókuszban az olcsó munkaerő és az egyedi igények legyenek.
* Szerkesztő bemutatása egy példán keresztül.
* Szerkesztői fogalmak tisztázása.
* Szájbarágós API magyarázás törlése.
* Kiegészítők.
* Feladatkiírási lap szkennelése/beillesztése
* Mit kell berakni az irodalomjegyzékbe