[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.]**

1. **Nyilatokzat**

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**

[**1. Bevezetés**](#_cgzg9wnxn5k) **1**

[1.1 Témaválasztás indoklása](#_ht5770aingsc) 1

[1.2 Megrendelő problémája](#_iw62k5azmmp0) 1

[1.3 Fejlesztő problémája](#_dnaq38lyhwip) 1

[1.4 Környezet kiválasztása](#_qjxporec7zkh) 2

[1.5 Példa probléma](#_97qb6khpvas) 6

[**2. Szerkesztő bemutatása**](#_m4n5yksiwj3g) **7**

[2.1 Telepítés](#_gwp0mgjvyx21) 7

[2.2 Beállítások és eszközök:](#_ln7d56ja7m61) 8

[2.3 Új Projekt](#_a57087y5d6mn) 9

[2.4 Projekt szerkezete](#_ya3xefsku10z) 9

[2.5 Kezelőfelület beállítása](#_cfw0esac7rv3) 11

[2.6 Példa elkészítése](#_eacerrrqxfp) 13

[2.6.1 Projekt beállítása](#_vu88l3gzef4) 13

[2.6.2 Elrendezés](#_9g6o739kim8u) 13

[2.6.3 Esemény lap](#_47ntf6nv40t1) 16

[2.6.4 Exportálás](#_intp6c9mo38z) 21

[**3. Tömeges email küldő**](#_9j711pr2wm6p) **22**

[**4. Construct SDK**](#_buqz56260rog) **22**

[**5. Kiegészítők elkészítése**](#_5oz68eun4o0t) **22**

[**6. Tömeges email küldő elkészítése**](#_19ggyqyhmpj6) **22**

[**7. Konklúzió**](#_gsszlbn8111u) **23**

[**Irodalomjegyzék**](#_wsikuiwoz7qw) **24**

[**Mellékletek**](#_4cdc7r4caisp) **26**

# 1. Bevezetés

## 1.1 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.

## 1.2 Megrendelő problémája

A szoftverfejlesztés egyik legnagyobb nehézsége, hogy a megrendelőnek egyedi igényei lesznek. A szoftvert a feladatra testre kell szabni és az informatikai környezetbe be kell illeszteni, de a nagy költségeket sem képesek mindig kifizetni. Az egyedi szoftverfejlesztés költségei nagyon magasak már a munka kezdetekor. Külön szakembert/csapatot kell felbérelni és az esetleges további módosításokhoz is a szakemberre/csapatra leszünk utalva. Ha a prototípus elkészítése valamilyen okból nem sikerül, az nagy veszteségeket okozhat, mert a prototípus fejlesztése során is szakembereket kellett fizetnünk és elkészült program se keletkezett.

## 1.3 Fejlesztő problémája

Problémát jelent a fejlesztések költségének féken tartása, illetve szükséges szakemberek keresése, betanítása. A költségek nagy részét a munkaerő bére teszi ki. Gazdaságosabb a megrendelt programot nem teljesen alapjaitól megírni, érdemes kiválasztani egy keretrendszert (sablont), hogy akár kevésbé szakképzett fejlesztő, kevesebb idő alatt készítsen el egy programot. Továbbá a fenntartási költségeken is érdemes spórolni, ha az alkalmazás funkcióit önállóan külső szolgáltatás igénybevétele nélkül is tudjuk üzemeltetni.

## 1.4 Környezet kiválasztása

Keretrendszer kiválasztásakor a legfontosabb szempontok:

* **platformfüggetlen**: Az elkészített 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 fejlesztőkörnyezet is platformfüggetlenül használható.
* **bővíthető**: A keretrendszer funkcióit ki lehessen egészíteni saját tartalmakkal, kódokkal. Előny, ha a kiegészítés teljesen vagy részlegesen kompatibilis más programnyelvekkel, más programokban használt kiegészítőkkel/tartalmakkal vagy támogatást biztosít ezek importálásához.
* **könnyen kezelhető**: A keretrendszerhez legyen támogatás, dokumentáció és frissítés, ami a későbbi munkát segíti. A fejlesztői program kezelése könnyen elsajátítható legyen (pl. hasonló programoktól kevés eltérés). A fejlesztői vagy az elkészített 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 legyen a program, ne igényelje külső SDK letöltés, ne kelljen környezeti változókat, regisztereket beállítani stb.).

A piacon rengeteg rendelkezésre álló programnyelv és szoftver van. Elsősorban felhasználó számítógépén közvetlenül futtatható programokat készítő keretrendszerek között keresek, mert így a fenntartási költségeket is lehet csökkenteni. Backend rendszerekkel a kliens program nem találkozik közvetlenül és ezek a rendszerek külön környezetben működnek, ezért backend rendszereket most nem keresek, de szempont, hogy a kliens programot külső backend rendszerhez könnyen lehessen illeszteni.

Programozási nyelv tekintetében a mai rendszerek elég vegyesen épülnek fel. A keretrendszer különböző eszközei/funkciói valószínűleg különböző nyelveken íródtak. Nincs programnyelv béli megkötés, de böngészőben történő futtatáshoz valamilyen webes technológia használata (részbe vagy egészben) elengedhetetlen lesz a keretrendszer részéről.

* **Qt**: Platformfüggetlen C++ alkalmazás keretrendszer grafikus és nem grafikus alkalmazások készítéséhez. Kezelése nagyobb szaktudást igényel, kisebb projekteknél nem térül meg a keretrendszer ára.
* **Flutter**: Modern és reszponzív felhasználó felület építésére összpontosító keretrendszer, azonban itt a közös kódot a fordító platformspecifikus natív kóddá alakítja, így a platformok között eltérés tapasztalható.
* **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. Megjelenítés HTML elemeken keresztül történik, az különböző elemeket interaktívan kezeli.

Számtalan további keretrendszer létezik, de ezek a keretrendszerek speciális programozói tudást igényelnek, telepítésük, a kód karbantartása, továbbadása és verziókezelése nehézkes lehet egy kezdő számára, így egyszerűbb keretrendszereket kerestem, CMS (Content Management System) rendszerek a tartalom kezelésére összpontosít, általában kényelmes grafikus felületet biztosít a működés, tartalom kezelésére.

* **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.

Számtalan további tartalomkezelő rendszer létezik különböző célokra optimalizálva. Például Joomola[2], Drupal[3], Wix[4]. Bővíthetőség, könnyen kezelhetőség adott, de kizárólag webszerverre telepítve, webes tartalmakat generál, offline futtatható alkalmazás készítésére nem alkalmas és a kiegészítők összességében drágává tehetik.

Programok készítésére no code/low code keretrendszerek is rendelkezésre állnak. Ezek a rendszerek a programkészítés során a hagyományos kódolási feladatokat csökkentik és helyezik át egy kezdők számára könnyen átlátható grafikus felületre. Hasonló grafikus felületet biztosítanak, mint a CMS rendszerek, de ezeknél maga a program készítés nagyobb hangsúlyt kap.

* **Appsheet**: Google üzleti alkalmazás készítő szolgáltatása. Többi Google szolgáltatás összekötésével egyedi munkamenetünkhöz igazodó alkalmazást hozhatunk létre.
  + Előnyök: Jól támogatott, egyszerű a használata
  + Hátrányok: Csak szolgáltatás formában elérhető, nem lehet teljes mértékben testre szabni
* **Appery.io**: Webes és platformspecifikus applikáció készítő szolgáltatás. Vizuális szerkesztő mellett backend szolgáltatásokat is biztosít. Különböző bővítményekkel bővíthető az alkalmazás
  + Előnyök: Sokféle funkció és kiegészítés, nagyfokú testre szabási lehetőség
  + Hátrányok: Csak szolgáltatás formában elérhető, kisebb projektekhez drága

Legtöbb ilyen rendszer csak szolgáltatásként érhető el és a teljeskörű bővítése nem lehetséges. Rengeteg további no code[5] low code[6] alkalmazás van a piacon.

Tovább bővítettem a keresést játékmotorokra is, mivel ezekkel a rendszerekkel olyan programokat lehet írni, amik külső szolgáltatás nélkül, gyorsan tudnak reagálni a felhasználói bevitelre, animációkat, rendszer funkciókat tudnak 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 a motorról.
  + Előnyök: Nagy közösség, kiterjedt funkcionalitás, fejlett szerkesztő
  + Hátrányok: Használata speciális tudást igényel (programozási nyelv, függvények/motor ismerete)
* **Construct 3**, **Game Maker: Studio**, **Gdevelop**: Ezen programok az előzőekben bemutatott programok ötvözete. Gyors tanulási lehetőséget biztosítanak vizuális programozási lehetőséggel (visual scripting), azonban lehetőség van a motort bővíteni hagyományos programozási nyelvekkel, így amik bonyolultabb projektek is készíthetőek.
  + Előnyök: Bárki számára könnyen elsajátítható, bővíthető
  + Hátrányok: kevésbé professzionális megoldások, esetleges sebesség béli 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 GML fordítót használ a kész program C++ 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 kissé 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. A Gdevelop viszont ingyenes és beüzemelése is egyszerűbb. „Game Maker: Studio” bővíthetősége jobb, mint a Gdevelop programnak. Az egyszerűbb beüzemelés és használat miatt, én mégis a Gdevelopot részesítem előnyben.

*1. táblázat: Game Maker: Studio, Gdevelop összehasonlítás*

|  | 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 hangsúlyosabb [7]. A Construct 3 egyszerűbben elérhető, könnyebb kezelhetősége, bővíthetősége rugalmasabb. Dokumentáció és SDK jobban elérhető Construct 3-hoz.

*2. táblázat: Construct, Game Maker: Studio összehasonlítás*

|  | 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.

3. táblázat: *Construct, Gdevelop összehasonlítás*

|  | 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.

## 1.5 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 (Outlook, Thunderbird, stb.):
  + 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.

# 2. Szerkesztő bemutatása

A szerkesztő bemutatását egy alap példa programon keresztül fogom bemutatni. A példa feladat egy téglalap kiszámító programot takar. A program a téglalap két oldalának hosszát fogja kérni és kiírja a kiszámolt téglalap területét.

## 2.1 Telepítés

Az szerkesztő bármely verziójának megnyitásakor a böngésző elkezdi letölteni és frissíteni a különböző forrásokat. A letöltés végeztével a program internet nélkül is használható, azonban a felhő mentéshez, felhőben történő exportáláshoz és a licenckulcs frissítéséhez internetkapcsolat szükséges.

Frissítések heti rendszerességgel érkeznek. Szakdolgozat írásakor a r308.2 a legfrissebb stabil verzió. 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 tartalmazzák.
* Béta verzió: <https://editor.construct.net/beta>. Új funkciók, komplexebb javítások első megjelenése. Kevésbé tesztelt, néha hibásan működhet.
* 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 frissítések kihagyását ez általában hibamentesebb fejlesztést eredményez.
* Biztonságos mód: <https://editor.construct.net/?safe-mode>. Minden „3rd party” (külső fél) kiegészítőjét kikapcsolja, hogy elindulhasson a szerkesztő, ha valamelyik kiegészítő hibásan működött.

## 2.2 Beállítások és eszközök:

Bármely verzió indításkor egy előre beállított kezdőképernyő fogadja a felhasználót, ahol hasznos linkeket, ajánlott minta projekteket és a legutóbb megnyitott projekteket ér el.



*1. ábra: Construct szerkesztő kezdőoldala*

A felső sávon balra lévő *Menu* (*Menü*) gomb állandó elem. Ennek segítségével hívható elő a szerkesztő fontosabb beállításai.

*Menu* (*Menü*) > *Settings* (*Beállítások*) menüpont tartalmazza a szerkesztő beállításait. Itt állíthatóak a nyelv, felületi beállítások, mentési helyek és szerkesztői konfigurációk.

Szerkesztőt kiegészítőkkel lehet bővíteni a *Menu* (*Menü*) > *View* (*Nézet*) > *Addon manager* (*Kiegészítő kezelő*) menüpontban. Itt lehet a telepített kiegészítőket lehet megtekinteni, törölni, illetve új kiegészítő telepítésére is lehetőség van.

## 2.3 Új Projekt

Új projekt létrehozásához a *Menu* (*Menü*) > *Project* (*Projekt*) > *New* (*Új*) elemet kell használni. Projekt létrehozásakor egy felugró ablakban kezdeti konfigurációkat adhatunk meg, ezek a beállítások később is módosíthatóak. Jelenleg semmilyen beállítást nem adunk meg csak a *Create* (*Létrehozás*) gombra kattintsunk.

Létrehozás után plusz ablakok jelennek meg a kezelőfelületen. Ezek testreszabásáról későbbiekben lesz szó.

## 2.4 Projekt szerkezete

Projekt létrehozása után alapértelmezett beállítások használatakor a jobb oldalon a *Project* (*Projekt*) ablakban láthatóak a projekt szerkezete:



*2. ábra: Szerkesztő projekt ablaka*

Projekt nevével ellátott mappa a főkönyvtár. Ezen belül almappák, az almappák-on belül elemek vagy további almappák találhatóak. Mappák és elemek átnevezését, törlését, másolását és egyéb funkciókat tartalmazó menü a jobb egérgombbal hívhatóak elő. Mappák funkciók szerint vannak kategorizálva:

* **Layout (Elrendezések)** mappa tartalmazza az elrendezéseket, vagyis azokat a felületeket, amit a felhasználó a kész programban látni fog. Elrendezésen belül egymás alatt elhelyezkedő lévő rétegek találhatóak. Egyszere egy elrendezés jeleníthető meg de a rétegeket másik elrendezés módosíthatja. Elrendezésekhez egy eseménylap is társítható.
* **Event sheets (Esemény lapok)** mappa tartalmazza az esemény lapokat, amik az elrendezések vezérlését végzik. Eseménylap fogja tartalmazni a program konkrét működési logikáját. Eseménylapok egymásba is beágyazhatóak.
* **Scripts (Szkriptek)** mappa JavaScript kódok tárolására szolgál. Itt tárolt programkódoknak ugyanaz az a feladata, mint az esemény lapoknak csak nem vizuálisan történik a programlogika elkészítése, hanem JavaScript nyelven egy szöveges fájlban.
* **Object types (Objektum típusok)** mappa tartalmazza a projekt építő elemeit az objektumokat. Minden látható vagy nem látható elemet, funkciót (Gomb, Hang, Videó, Adattárolás) objektumok valósítanak meg, ezeket a projekthez kell adni.
* **Families (Családok)** mappában családok szerepelnek. Ezek olyan csoportosítások, ahol az azonos objektumokat (pl.: Gombokat) egy csoportba (családba) rakhatjuk ezzel egyszerűbben egybe tudjuk őket kezelni (pl.: családban lévő minden szöveg objektum piros színre állítása)
* **Timelines (Idővonalak)**: ebben a mappában idővonal elemek találhatóak. Egy idővonal elemmel a különböző objektumok értékeinek időbeli vezérlését adhatjuk meg. *Eases* (*Átmenetek*) almappában az egyedi átmenetek találhatóak. Az átmenet két értékváltozás közötti interpoláció (pl.: lineáris, köbös stb.) adja meg.
* **Sounds (Hangok)** mappa olyan hang fájlok tárolására szolgál, aminek az időben történő lejátszása kritikusan fontos (pl. kattintás hanga), ezért az ide másolt hangokat a program indítás előtt be fogja tölteni.
* **Music (Zenék)** hasonlóan a hang mappához itt is hangfájlokat lehet tárolni, de az itt található audió fájlok nem töltődnek be a program indulásával, így lehetnek kisebb megakadások a lejátszás során.
* **Videos (Videók)** mappában videó fájlokat lehet tárolni
* **Fonts (Betűtípusok)** különböző betűtípus fájlok (.ttf, .woff, stb.) helye
* **Icons (Ikonok)** a program által használt ikon fájlok (.png). Beállítható, hogy fájl melyik ikon helyhez legyen társítva.
* **Files (Fájlok)** mappában minden fajta fájl beállítható, akár az előzőekben bekategorizált fájltípusok is. Csak az egyéb, nem kategorizálható fájlokat érdemes ide másolni, mert ebben a mappában lévő fájlokat a szerkesztő nem fogja optimalizálnit (pl.: hang esetében tömörítést) végezni.

## 2.5 Kezelőfelület beállítása

A *Menu* (*Menü*) > *View* (*Nézet*) > *Bars* (*Sávok*) menüből nyolc különböző ablakot lehet előhívni, ezek az ablakok különböző funkciókat biztosítanak [8]. A projekt elrendezéseit esemény lapjait, egyéb fájljait is külön ablakban lehet megjeleníteni.



*3. ábra: Szerkesztő alapértelmezett felülete*

Mindent ablakot törölve csak felső menü elemek marad.



*4. ábra: Szerkesztő üres felülete*

Ablak megnyitás után szabadon áthelyezhető a felületen a felső szélének egérrel történő húzásával.

Lehetőség van az ablakokat dokkolni is, ezt az ablak kék négyzetekre való ráhelyezésével tehetjük meg. Dokkolni a képernyő vagy egy ablak négy oldalára lehet. Ha ráillesztjük az ablakot egy másik ablakra, akkor megjelenő fülekkel lehet váltani az egymást fedő ablakok között. Dokkolás helyét jelzi a program.



*5. ábra: Szerkesztő felülete átrendezés közben*

Mobilos nézetben az idővonal ablak nem elérhető, illetve az ablakok nem változtathatóak csak oldalról behúzhatóak.

Továbbiakban a példa készítésénél az alapértelmezett elrendezést fogom használni, vagyis bal oldalon Properties (Tulajdonságok) ablak, jobb oldal felső részén Project (Projekt) ablak, jobb alsó részén Layers (Rétegek) ablak lesz.

## 2.6 Példa elkészítése

### 2.6.1 Projekt beállítása

Egész projektre vonatkozó beállítások eléréséhez kattintsunk bal egérgombbal a projekt nevére a *Project* (*Projekt*) ablakban. Ezután a tulajdonság ablak a projekt tulajdonságait fogja mutatni. Properties (Tulajdonságok) ablakban szerkesszük a következő értékeket:

* *Name* (*Név*) mezőbe írjuk be a projekt nevét: „Terület számoló”
* *Description* (*Leírás*) mezőbe adjunk meg: „Téglalap terület számoló program.”
* *ID* mezőbe írjuk be: „com.teglalap.terulet”
* *Author* (*Szerző*) mezőbe írjuk be magunkat.
* *First layout* (*Első elrendezés*) listából válasszuk ki az egyetlen elrendezésünket.



*6. ábra: Projekt tulajdonságainak beállítása*

*Viewport size* (*Nézetablak méret*) mezőben a programunk virtuális felbontását állíthatjuk. Ez nem valódi felbontást jelent (futtatáskor a megjelenéshez igazítja a felbontást is), inkább arányokat jelen, vagyis az elrendezéseket és az objektumokat mekkora nagyságban kell megjeleníteni.

Többi beállítással jelenleg nem foglalkozunk, projekt létrehozásakor a sablonok ezeket a projekt tulajdonságokat állítják be előre.

### 2.6.2 Elrendezés

*Project* (*Projekt*) ablak > *Layouts* (*Elrendezések*) mappa > *Layout 1* (*Elrendezés 1*) elemet dupla bal egér kattintással vagy jobb egérgomb és *Open* (*Megnyitás*) gombbal nyithatjuk meg (ha nem lenne megnyitva). Ilyenkor az elrendezés nevével elnevezett ablakban lesz látható az elrendezés.

Ha kiválasztjuk az elrendezés elemet egyszeri bal egér kattintással a *Properties* (*Tulajdonságok*) ablakban megjelennek az elrendezés tulajdonságai. Szerkesszük a *Size* (*Méret*) mezőket 854x480-as értékre, ezzel az elrendezésünk méretét az projektünk virtuális méretéhez igazítottuk.



*7. ábra: Elrendezés tulajdonságainak beállítása*

Projektünk különböző részeit objektumok fogják alkotni. Ezeket az *Object types* (*Objektum típusok*) mappában tudjuk kezelni. Új objektum hozzáadásához először a *Project* (*Projekt*) ablakban > *Object types* (*Objektum típusok*) mappára > jobb egérgombbal kattintva > *Add new object type* (*Új objektum típus hozzáadása*) gombbal el kell hívni a *Plugins* (*Bővítmények*) listáját.



*8. ábra: Objektum hozzáadása*

A példa programunkhoz két beviteli (hosszúság és szélesség) mező és egy megjelenítési (kiszámolt terület) mező kell. Beviteli mezők *Text Input* (*Beviteli mező*) típusú objektumok lesznek, míg az eredmény megjelenítéséhez egy *Text* (*Szöveg*) típusú objektumot használunk.



*9. ábra: Objektum lista*

Létrehozott három objektumokat egérrel való húzással elhelyezhetőek az elrendezésen. Rendezzük el tetszés szerint a szövegdobozokat.

Bal egér kattintással kijelölt objektumok tulajdonságai meg fognak jelenni a *Properties* (*Tulajdonságok*) ablakban. Állítsuk be a három objektum tulajdonságait:

* Eredmény szöveg
  + *Name* (*Név*) legyen: „Eredmeny”
  + *Text* (*Szöveg*) legyen „Eredmény:”
* Hosszúság beviteli mező:
  + *Name* (*Név*) legyen: „Hosszusag”
  + *Placeholder* (*Helyőrző*) legyen „Hosszúság”
* Szélesség beviteli mező:
  + *Name* (*Név*) legyen: „Szelesseg”
  + *Placeholder* (*Helyőrző*) legyen „Szélesség”



*10. ábra: Elrendezésen lévő objektumot szerkesztése*

Ezzel a felhasználói felület elkészült. *Menu* (*Menü*) > *Project* (*Projekt*) > *Preview* (*Előnézet*) gombbal vagy fenti lejátszás gombbal kipróbálható a projekt.

### 2.6.3 Esemény lap

A programot esemény lapokban lévő utasításokkal vezérelhetjük. Eseménylap ugyan olyan funkciót tölt be, mint egy programkód csak a megírt vezérlés emberközelibb formában jelenik meg pl.: leíró jellegű, külön részekre bontható, nincs szigorú szintaxis, beépített „varázslók” (végigvezet a beállításokon).

*Project* (*Projekt*) ablak > *Event sheets* (*Esemény lapok*) mappa > *Event Sheet 1* (*Esemény lap 1*) elemet dupla bal egér kattintással vagy jobb egérgomb és *Open* (*Megnyitás*) gombbal nyithatjuk meg (ha nem lenne megnyitva). Ilyenkor az esemény lap nevével elnevezett ablakban lesz látható az esemény lap.



*11. ábra: Üres eseménylap szerkesztése*

Egy esemény lap Action (Művelet), Condition (Feltétel), Expression (Függvény) (angol rövidítés: „ACE”) típusú blokkokból épül fel. Blokkok hierarchikusan épülnek fel, egy blokk tartalmazhat al-blokkokat.

* Művelet a program „cselekvése”. A valamilyen konkrét műveletet végez pl.: írás, olvasás, áthelyezés, megjelenítés stb. Két fajta művelet létezik:
  + **Szinkron művelet**, ahol művelet kezdete és befejezése egyebe történik (nem bontható szét időben).
  + **Aszinkron művelet**, ahol a művelet kezdete és befejezése külön történik meg (szét bontható időben). Ilyenkor kezdés és befejezés között hosszabb idő telik el pl.: fájl beolvasást kezdése és fájl adatainak beolvasása között hosszabb idő telik el. Szerkesztő egy óra szimbólummal jelöli.
* Feltételek feladata műveletek lefutásának kezelése. A feltételben lévő blokkok akkor aktiválódnak, ha a feltétel igaz. Feltételeknek 3 fajtája van:
  + **Normál feltétel**, akkor aktiválja az alatta elhelyezkedő blokkokat, ha a feltétel igaz.
  + **Ciklus feltétel** hasonló a normál feltételhez, de a feltétel újra indíthatja magát és a benne lévő blokkokat, ha a feltétel igaz. Szerkesztő egy ismétlés ikonnal jelzi.
  + **Esemény** olyan feltétel, ami egy történés hatására (pl.: egérkattintás, betöltés befejezése stb.) lefuttatja az alatta elhelyezkedő blokkokat. Szerkesztő jobbra mutató zöld nyíllal jelöli.
* **Függvény** segít műveletek és feltételek paramétereinek megadásában. Függvényeknek szöveg vagy szám formájában paramétereket adhatunk át és szöveget vagy szám értéket adnak vissza.

Legkülső blokkok (kivéve az eseményeket) minden képernyő frissítéskor lefutnak (általában 60-szor másodpercenként).

Most a konkrét példát készítjük el, a terv az, hogy ha a felhasználó megadja a hosszúságot és szélességet akkor a program beolvassa a két oldal és összeszorozza, az eredményt pedig kiírja a képernyőre. Példa programunkba egy eseménnyel figyeljük, hogy a beviteli (hosszúság és szélesség) mező tartalma megváltozott-e, vagyis a felhasználó írt-e bele valamit.

1. Add event (Esemény hozzáadása) gombbal megnyitjuk a varázslót. Kiválasztjuk a hosszúság megadására szolgáló korábban megadott szöveg beviteli objektumot (nálam „Hosszusag” névvel rendelkezik).
2. A felsorolt feltételek közül kiválasztjuk az On text changed (Szöveg változásakor) esemény feltételt. Ez további paramétert nem kér, így hozzá is adja a szerkesztő az esemény lapunkhoz.

Mivel a másik beviteli mezőt (hosszúságot) is figyelni kell erre is létrehozunk a fenti módszerrel egy eseményt. A két esemény összefűzhető egy feltétel blokkba, azonban a feltétel blokk alapértelmezetten AND (ÉS) kapcsolatot akar kialakítani a két feltétel között. Ezt meg kell változtatnunk OR (VAGY) kapcsolatra. A blokk szélére jobb egér kattintással előhívható menüből keressük ki Make OR block (VAGY blokk létrehozása) elemet.



*12. ábra: Feltétel blokk VAGY kapcsolatra állítása*

A VAGY blokkba már összehúzható a két feltétel. Ebben a blokkban lévő további blokkok akkor fognak aktiválódni, ha a hosszúság vagy szélesség mezőt megváltoztatja a felhasználó.

Most a meglévő feltételek alá szeretnék új üres blokkot adni. Ezt a feltételünk bal szélére (így az egész feltétel blokkot kijelöli, nem csak a feltétel részeket) történő jobb egér kattintással előhívható menüből tehetjük meg.



*13. ábra: Üres blokk hozzáadássa*

Létrehozott üres blokkhoz adjunk egy változót is, amiben a kiszámolt területet tároljuk, ezt az üres blokkra történő jobb egérkattintással, majd az Add (Hozzáadás) > Add local variable (Lokális változó hozzáadása) gombbal tehetjük meg. A felugró ablakba adjuk meg a változó nevét („Terulet” nevet fogom használni).



*14. ábra: Változó hozzáadása*

Most kiszámoljuk a területet az oldalak szorzásával majd eltároljuk az előbb létrehozott változóba. Üres blokk mellett lévő Add action (Művelet hozzáadása) gomb megnyomásával előugró ablakban válasszuk ki a System (Rendszer) elemet, mert a változókkal történő műveletek rendszer objektumba találhatóak. Set value (Érték beállítása) műveletet kiválasztása után a következő párbeszédablakban a művelet paramétereit állítjuk be. Nem kell más változót megadni (összesen egy változónk van a projektben), az érték részhez pedig a területet kell megadni.

A függvények segítségével dinamikusan tudjuk számolni az oldalakból a területet. A Find Expressions (Kifejezések keresése) gomb segítségével előhívható, hogy az adott objektumoknak milyen függvényei vannak, illetve milyen és mennyi paramétert kell átadni a függvényeknek. Következő függvényeket fogjuk használjuk:

* Beviteli mező Text függvényét, ami a beviteli mezőbe írt aktuális szöveget adja vissza. Hivatkozni a beviteli mező nevével lehet majd ponttal elválasztva a függvényt meghívni pl.: Hosszusag.Text, Szelesseg.Text
* Redszer float függvényét, ami első paraméterben megadott szöveget konvertálja számmá, ha nem konvertálható nullát ad vissza.

A terület számítási képletek a következő lesz: float(Hosszusag.Text) \* float(Szelesseg.Text)



*15. ábra: Területszámítás elkészítése*

A kiszámolt terület után ellenőrizzük, hogy a terület nagyobb-e, mint nulla. Ha igen akkor kiírjuk az eredmény különben kiírjuk, nem jó adatok.

Adjunk a területszámítás alsó szomszédságába egy üres blokkot (fenti üres blokk hozzáadása rész ismétlése).

Az újonnan létrehozott üres blokra duplán bal egérrel kattintva feltételt adhatunk meg. System (Rendszer) objektumon belül Compare variable (Változó hasonlítása) feltételt válasszuk ki. Paramétereket úgy állítsuk be, ha a változó nagyobb mint nulla akkor legyen igaz a feltétel.



*17. ábra: Területszámítás ellenőrzése*

A feltételhez adjunk meg kiírás műveletet. Add action (Művelet hozzáadása) gomb után válasszuk ki az eredményt megjelenítő szöveg objektumot. A szöveg objektum Set text (Szöveg beállítása) művelet segítségével kiírható az eredmény. Statikus szöveget “-jel között lehet megadni, &-jelel lehet összefűzni a szövegeket.

****

*18. ábra: Eredmény kiírása*

Különben feltételt az egész feltételre történő bal egérkattintással, majd Add (Hozzáadás) > Add 'else' ('Különben' hozzáadása) gombbal tehetjük meg. A létrejövő feltételhez készítsünk egy műveletet, ami az eredmény szövegbe kiírja, hogy „hibás eredmény” (est hasonlóan kell megtenni, mint az eredmény kiírást csak „hibás eredmény” szöveggel). Elkészült programunkat tesztelhetjük is a fenti lejátszás ikonnal.



*19. ábra: Végső programlogika*

### 2.6.4 Exportálás

Exportálás során a projektünket önállóan futtatható programmá tesszük. Asztali exportáláshoz menjünk a: *Menu* (*Menü*) > *Project* (*Projekt*) > *Export* (*Exportálás*) > *NW.js* elemre



*20. ábra: Projekt és végső program kapcsolata*



*21. ábra: Exportálás elkezdése*

A felugró ablakon lévő beállításokat maradhatnak alapértelmezett értékeken, de a *Minify mode* (*Tömörítési mód*) elemet *advance* (*haladó*) értékre állíthatjuk gyorsabb kisebb méretű programhoz, azonban ebben a módban lassabb lesz az exportálás.

Következő ablakban az operációs rendszert és egyéb NW.js-re vonatkozó paramétereket választhatjuk ki (alapértelmezett beállítások megfelelőek). Ha valamely rendszerre nem akarunk exportálni, akkor az operációs rendszer neve ne legyen kijelölve a felsorolásban.

Ezután elkezdődik a program exportálása. Exportálás után egy csomagolt fájl fog letölteni a böngészőnk. Ez a fájl a programunk futtatható változatát tartalmazza.

# 3. Tömeges email küldő

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 4. Construct SDK

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 5. Kiegészítők elkészítése

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 6. Tömeges email küldő elkészítése

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 7. Konklúzió

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# Irodalomjegyzék

[2] Joomola

<https://www.joomla.org/about-joomla.html>, 2022.10.25 22:03

[3] Drupal

<https://www.drupal.org/about>, 2022.10.25 22:03

[4] Wix

<https://www.wix.com/about/us>, 2022.10.25 22:03

[5] List of low-code development platforms

<https://en.wikipedia.org/wiki/List_of_low-code_development_platforms>

[6] List of no-code development platforms

<https://en.wikipedia.org/wiki/No-code_development_platform>

[7] 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

[8] 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]