Általános információk

A diplomaterv szerkezete:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletesés pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő Diplomaterv sablon dokumentum tartalma. Ügyeljen a konzulens nevét és a beadás évét jelölő szövegdobozokra, mert azokra külön ki kell adni a frissítést. A mezők tartalma a sablonban a dokumentum adatlapja alapján automatikusan kerül kitöltésre.

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2,5 cm, baloldalon 1 cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le.

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben. Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

* a szakdolgozat készítő/diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás Karunkon, ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
* mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozat készítést, ill. diplomatervezést kívánunk!

FeladatkiÍrás

TODO: pontosan mi idén a policy? Elvileg a tanszékvezető utólag aláírja.

A feladatkiírást a tanszék saját előírása szerint vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.

C:\Users\szarnyasg\Downloads\bme_logo_nagy.eps

**Budapesti Műszaki és Gazdaságtudományi Egyetem**

Villamosmérnöki és Informatikai Kar

Játékfejlesztés Unity környezetben

Készítette

Szlovák Máté ÁkosKonzulens

Dr. Blázovics László

2017

Tartalomjegyzék

[Összefoglaló 6](#_Toc396824923)

[Abstract 7](#_Toc396824924)

[1. Bevezetés 8](#_Toc396824925)

[1.1. Formázási tudnivalók 8](#_Toc396824926)

[1.1.1. Címsorok 8](#_Toc396824927)

[1.1.2. Képek 8](#_Toc396824928)

[1.1.3. Táblázatok 8](#_Toc396824929)

[1.1.4. Kódrészletek 8](#_Toc396824930)

[1.1.5. Irodalomjegyzék 9](#_Toc396824931)

[2. Utolsó simítások 10](#_Toc396824932)

[3. Összefoglalás 11](#_Toc396824933)

[Köszönetnyilvánítás 12](#_Toc396824934)

[Ábrák jegyzéke 13](#_Toc396824935)

[Táblázatok jegyzéke 14](#_Toc396824936)

[Irodalomjegyzék 15](#_Toc396824937)

[Függelék 16](#_Toc396824938)

Hallgatói nyilatkozat

Alulírott Szlovák Máté Ákos, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2017. 11. 15.

Szlovák Máté Ákos

# Összefoglaló

A szórakozás mindenki számára mást jelent, azonban a mai világban nincs olyan ember, aki legalább ne próbálta volna ki a számítógépes játékokat. Többek között ez is egy fő oka annak, hogy a játékfejlesztés ekkora népszerűségnek örvend. Játékok fejlesztésére számtalan mód ismert. Lehetséges különböző játékmotorok használata, vagy akár azok kikerülésével, csupán kód írásával is létrehozhatunk játékokat. A játékmotor tulajdonképpen egy szoftveres keretrendszer, amelyet játékok készítésére használnak. Lehetőséget ad mindenhez, ami csak szükséges lehet egy játékhoz. Egyik leghasznosabb eszközük, hogy nincs szükség külön a fizika megvalósítására, de fontos megemlíteni még például, hogy nagyon egyszerűen készíthetünk animációkat, valamint szkriptek segítségével testreszabhatjuk a játékunkat. A két legismertebb játékmotor kétségkívül az Unreal Engine, valamint a Unity. Mind a két eszköznek megvannak a maga sajátosságai, előnyei, illetve hátrányai, azonban rengeteg segítséget adnak a használójuknak.

Ezen szakdolgozat keretein belül a Unity játékmotor sajátosságait fogom bemutatni, és lépésekre lebontva, egy játék fejlesztésének módját feltárni. Ezután egy konkrét, általam elkészített játékra is bemutatom ezeket az általános lépéseket, és kitérek pár bonyolultabb részletre, megvalósításra(módszerre), ami nem feltétlenül szükséges minden játék megvalósításához. Szintén szerepelni fog a szakdolgozatomban az általam implementált mesterséges intelligencia, aminek segítségével a játékban található ellenfelek nem csak egy egyszerű gépi viselkedést mutatnak, hanem „okosan” cselekednek.

# Abstract

English translation of the abstract of the thesis work. This summarises the content of the thesis in 0.5–1 pages and is uploaded to the Thesis Work Portal as well.

# Bevezetés

A számítógépes játékok megjelenése szinte egészen a számítógépek megjelenéséhez vezethető vissza. Eleinte a játékoknak nem volt akkora érdekeltségi köre, mára azonban rengetegen elsősorban játékra használják a számítógépüket.

# Játékfejlesztő-környezetek

Az eleinte csupán nyers kód segítségével íródott játékokat hamar felváltották a játékfejlesztő-környezetek által készített játékok. Nem véletlen, hiszen exponenciálisan több munka az előbbi módon elkészíteni egy igényes programot, mint a második esetben.

Az első legelső belső nézetű lövöldözős játékok egyike, a Doom is egy játékmotorral készült 1993-ban, amelynek a neve a Doom-motor. Ez a játék annak idején forradalminak számított, és ezt a példát követve, rengeteg más játékhoz is használták ezt a motort, igaz, ez nem volt egy igazi 3D-s motor(nem lehetett le, illetve fel nézni), viszont a működése nagyon gyors volt. A Doom-motor idejében a másik legelterjedtebb játékmotor még talán az volt, amit a Wolfenstein 3D-hez használtak.

Ezek után a következő korszakalkotó változtatás 1998-ban történt, amikor megjelent az Unreal Engine 1. Ennek az elkészítését a Doom-engine készítője inspirálta, és három éven keresztül fejlesztették. Ezt főleg FPS(First Person Shooter) játékok készítésére fejlesztették ki, azonban hihetetlen széleskörű játékmotorra sikeredett, így az FPS játékokon kívül rengeteg lopakodós, RPG, illetve MMORPG-hez is tökéletesen használható. Nem véletlen,ha valaki nem nagyon ismeri a játékmotorokat, de az Unreal Engine ismerősen hangzik neki.

Az Unreal Engine 1998 óta rengeteg változtatáson ment keresztül, és sok, az adott időben legismertebb játékoknak volt alappillére a fejlesztésük során. Az Unreal Engine 1 által készített játék, amit mindenképpen szükséges megemlíteni, az az Unreal nevű FPS túlélőhorror. Számos más játék készítését követte a 2002-ben megjelent Unreal Engine 2, amely az America’s Army megjelenésével debütált. Az UE második verziójának egyik fő változtatása az elsőhöz képest az új fizikarendszer bevezetése volt.

Az Unreal Engine 2-t körülbelül két év fejlesztés után követte utódja, az Unreal Engine 3, 2004-ben. Ez a játékmotor rengeteg teljesítmény-javító változtatással érkezett, és ha eleinte nem is, később támogatta az Android, illetve iOS mobilplatformokra való fejlesztést.

Az Unreal Engine 4-et kilenc éven keresztül fejlesztették(2003-2012), viszont kétségkívül a leglátványosabb játékmotorra sikeredett. Igaz, a látvány mellé társul a hatalmas erőforrás igény, ezért vannak olyan fejlesztők, akik óvatosan használják.

Az Unreal Engine 3 megjelenésével egy évben megjelent a Crytek által fejlesztett CryEngine 1. Ez szintén egy nagyon neves játéknak, a Far Crynak volt az alapja. A CryEngine egyik sajátossága, hogy megjelenésének idejében hihetetlen grafikát kínált, aminek megjelenítéséhez nem volt szükség egetrengető hardverre. 2004 óta a CryEngine rengeteg változáson ment keresztül, a mai napig folyamatosan fejlesztik, hogy kompatibilis legyen az új hardverekkel.

Ezalatt a durva 10 év alatt egyébként számtalan más játékmotor is készült, azonban az Unreal Engine sorozat számított a legkorszakalkotóbbnak. Más, ismert játékmotorok közé tartoznak: Quake, Serious, GoldSource, Source, id Tech 1-2-3-4.

2005-ben megjelent a Unity Technologies által fejlesztett Unity. Akkor ezt csak OS X-re adták ki az Apple WWDC(Worldwide Developers Conference) konferenciáján, de azóta 27 platformra lehet vele játékokat készíteni. Unityvel Windows vagy OS X operációs rendszer alatt tudunk játékot készíteni, majd az ott elkészített programot számtalan különböző platformra át tudjuk vinni, és futtatni. A Unity egyébként nem csak 2, illetve 3D-s játékok fejlesztésére használható. Készíthetőek vele különböző építészethez használt tervek, illetve real-time látványos animációk.

## Unity vs. Unreal Engine 4

Az, hogy a Unityt válasszuk-e ha játékot akarunk fejleszteni, rengeteg dologtól függ. Mint minden más játékmotornak a Unitynek is megvannak a maga sajátosságai, hátrányai, illetve előnyei. Összehasonlításképpen most az Unreal Engine 4-et fogom használni, hiszen manapság ez a két játékfejlesztő-környezet a legelterjedtebb, és legszéleskörűbben használt.

Először is nagyon fontos kérdés, hogy milyen típusú játékot akarunk készíteni. Ha a célunk egy 2D platformer játék készítése, akkor ahhoz a Unity-nek nagyon jó 2D eszközei vannak, de az Unreal Engine-ben is kezdenek megjelenni ehhez hasonló eszközök. Szintén nagyon fontos kérdés, hogy milyen grafikával szeretnénk dolgozni. Az Unreal Engine 4 kétségkívül magasabb szintű a Unityhez képest ebből a szempontból, viszont ha nem célunk az abszolút elsőosztályú grafikájú játék készítése, a Unity is épp olyan tökéletes lehet nekünk. Felmerül a kérdés, ha megtehetjük hogy egy jobb grafikájú játékot készítsünk, miért ne tennénk? Erre a válasz egyszerű: A jobb grafikai játékoknak sokkal nagyobb az erőforrásigényük.

Azontúl, hogy milyen típusú játékot akarunk készíteni, szintén van egy nagyon fontos szempont, a pénz. A Unitynek több verziója is van, az ingyenes mellett egy Unity Plus, valamint egy Unity Pro. Abban az esetben, ha a célunk csak a játékok készítésének tanulása, megismerkedés a játékfejlesztés világával, akkor a Unity Personal, ami ingyenes, tökéletes választás lehet. Igaz, ez nem tartalmazza azokat a tulajdonságokat, amelyeket a Unity Pro használ, attól még nagyon jó játékokat készíthetünk vele. Ha azonban mégis szeretnénk frissíteni Unity Pro-ra, az évente 1500 dollárba, durván 400.000 forintba kerül, ami lássuk be, nem egy olcsó mulatság, pláne annak, aki csak el akar indulni a Unity világában. Ha azonban valaha is ki akarjuk adni a kész programunkat, nem kell a Unitynek szerzői díjat fizetni, az egészet zsebre tehetjük.

Ezzel szemben az Unreal Engine 4-nek nincsenek Pro, vagy Free verziói, minden ingyenes. Az Unreal Engine 4 összes eleméhez teljesen ingyenesen hozzáférhetünk. Nagy különbség még a Unityhez képest az, hogy amikor az itt készített játékunkat ki akarjuk adni, akkor 5%-os szerzői díjat kell fizetnünk az Epic Games-nek (az Unreal Engine kiadójának).

Ha a fő célunk a játékok kiadása és a bevétel, alaptőke nélkül, talán jobb választás lehet az Unreal Engine 4, ha azonban a tanulás a fő cél, akkor a Unity ingyenes verziója is egy nagyon erős játékmotornak számít.

A következő szempont amit érdemes figyelembe venni, az a programozás a játékmotoron belül. Nyilván ahhoz, hogy egy játékot elkészítsünk, ami nem csak a legalapvetőbb funkciókat tartalmazza, a nagy mennyiségű kód írása elkerülhetetlen. Az, hogy ki melyik programozási nyelvet szereti jobban, az egyéni preferencia, így ha valaki nagyon hajlik a C++, vagy a C#/JavaScript felé, annak a döntés elég egyszerű lehet, még a fent leírtakkal együtt is. A Unity C# és JavaScript programozhatóságot kínál, míg az Unreal Engine 4 a C++-t használja.

Mind a két fejlesztőkörnyezet rendelkezik egy Asset Store-val. Ez tulajdonképpen egy online áruház, amelyben a programukhoz vásárolhatunk elemeket. A Unity egyértelműen sokkal nagyobb palettával rendelkezik az itt elérhető eszközök terén, a hatalmas mennyiségű modellektől kezdve a különböző effekteken át, szinte minden elképzelhető animációig.

Mint már említettem, a grafikai eszközök, amiket az Unreal Engine 4 kínál, abszolút páratlanok. Komplex részecskeszimulációk, fejlett dinamikus megvilágítás, ezek mind nagyon fontos grafikai tulajdonságai az UE4-nek. A Unity is természetesen fejlődik a grafika terén, és egyre jobb és jobb kinézetű dolgokat valósíthatunk meg benne. Persze, a grafika nem minden, nem mindenkinek az a célja, hogy az épp idei játékokhoz méltó grafikai játékokat készítsen, sokan beérik egy kicsit “egyszerűbb” grafikával.

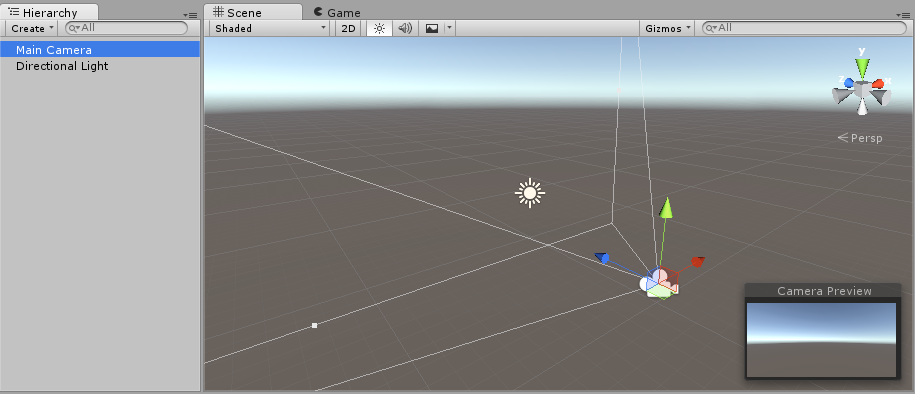
A használhatósági mérce a két játékmotorral kapcsolatban igazából az egyik felé se billen. Általában a Unityt tartják egy nagyon egyszerű környezetnek, amelyet könnyen meg lehet tanulni, és használni, de az Unreal Engine 4 nagyon fejlett és felhasználóbarát UI-al rendelkezik, ahhoz is nagyon könnyű adaptálódni.

A fent említett tulajdonságok, előnyök, hátrányok után mindenki maga döntse el, hogy szerinte melyik az a játékfejlesztő-környezet amit szeretne használni. Nem lehet egyértelműen kimondani, hogy az Unreal Engine 4 jobb a Unitynél, vagy fordítva, mindkettőnek megvannak a sajátosságai, előnyei, hátrányai. Ebben a szakdolgozatban a Unity-t fogom részletesebben bemutatni.

# Játékfejlesztés Unity-ben

Az előző fejezetben bemutattam a játékfejlesztő-környezeteket egészen visszamenőleg a legelsőig, aztán összehasonlítottam a Unity-t az Unreal Engine 4-el kitérve azok előnyeire, hátrányaira, illetve hogy mikor melyiket érdemes használni, milyen típusú játékok fejlesztéséhez.

## Scene – jelenet

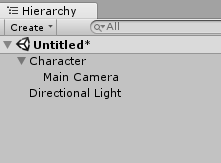
A Unity játékfejlesztés kapcsán az első fontos dolog amit érdemes megemlíteni, az a Scene, vagy Játékszín. Ezt tulajdonképpen úgy kell elképzelni, mint egy filmnek egy jelenete. Amikor Unityben létrehozunk egy projektet, először egy üres Játékszín fogad minket. Alapértelmezetten, a Játékszín tartalmaz egy kamerát, illetve egy fényforrást. A kamerakép azt mutatja, amit mi fogunk látni a játékból amikor elindítjuk, a fényforrás pedig értelemszerűen az a pont, ahonnan szeretnénk, hogy a Játékszín meg legyen világítva.

1. ábra: Scene - Játékszín

### Kamera

Amikor egy játékot készítünk, nagyon fontos az, hogy a játékos honnan látja a játékot, milyen szemszögből. Ha például FPS, azaz belső nézetű lövöldözős játékot akarunk csinálni, akkor a kamerának ott kell elhelyezkednie, ahol a karakterünk feje van.

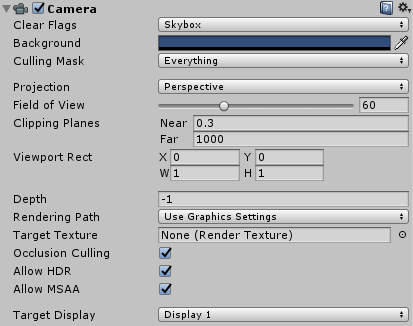
Ehhez először szükség van arra, hogy legyen egy karakter objektumunk, ami a játékost reprezentálja. Ha ez megvan, a kamerát be kell állítanunk, hogy jó helyen legyen a karakterhez képest, például a kamerapozicíó a fejénél legyen. Itt azonban nem elég, ha megvan a pozíció, hiszen ha mozog a játékos, a kamera nem fog vele mozogni, az ott fog maradni, ahova a kamerát beállítottuk, azaz a karakter kiindulópontjában. Erre a megoldás nagyon egyszerű, ha a hierarchiában a kamerát a karakterünk gyerekévé tesszük, akkor a kamera koordinátái a karakterhez képest fognak kiszámítódni.



2. ábra: Kamera elhelyezése

Természetesen nem muszáj mindig a karakterünkhöz képest mozgatni a kamerát, rengeteg más módszer is van. Például egy olyan játéknál, amely egy arénában játszódik felülnézetből, ideális lehet, hogy a kamerát úgy helyezzük el, hogy belássuk az egész játékteret, és akkor nincs szükség a kamera dinamikus mozgatására.

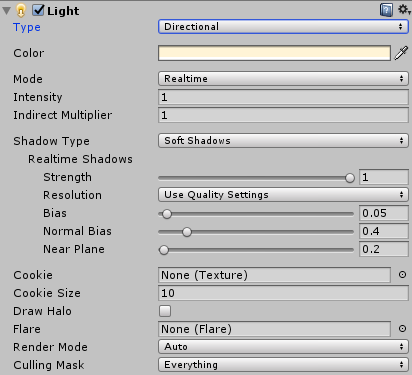
Magához a kamerához egyébként rengeteg minden mást is be tudunk állítani a pozícióján kívül, egyik nagyon hasznos tulajdonsága például a Field of View, azaz a látószög. Ezt fokban adhatjuk meg a kamera Inspector – Camera füle alatt, egyéb beállításokkal együtt.



3. kép: Kamera beállításai

### Fényforrás

Minden játékban, amely nem egy teljesen sötét kamrában játszódik, van fény. Fényforrásból, tetszőlegesen sokat létrehozhatunk, és elhelyezhetünk a Játékszínen belül. Minden, ami fényt áraszt az egy külön fényforrás objektum. Csakúgy, mint a kameránál, a fényforrásunkhoz is rengeteg tulajdonságot állíthatunk.



4. kép: Fényforrás beállításai

Három fő típusú fényforrást adhatunk meg:

* Directional: Ha természetes fényforrást szeretnénk létrehozni, akkor célszerű ennek a használata. Úgy kell elképzelni, mintha végtelen távolságban lenne, és onnan egy irányban szórná a fényt.
* Spot: A Spot típusú fényforrás az, amelynek megadhatunk egy távolságot, ameddig világít, valamint egy szöget, amely megadja, hogy mennyire szórja a fényt. Ez tulajdonképpen egy tölcsér, amelynek az aljából érkezik a fény, és a tölcsér szájáig tart.
* Point: Mint a neve is mutatja, ez egy pont típusú fényforrás. Egy pontból szórja a fényt, 360 fokban mindenfelé, és megadható hozzá egy távolság, ameddig világít

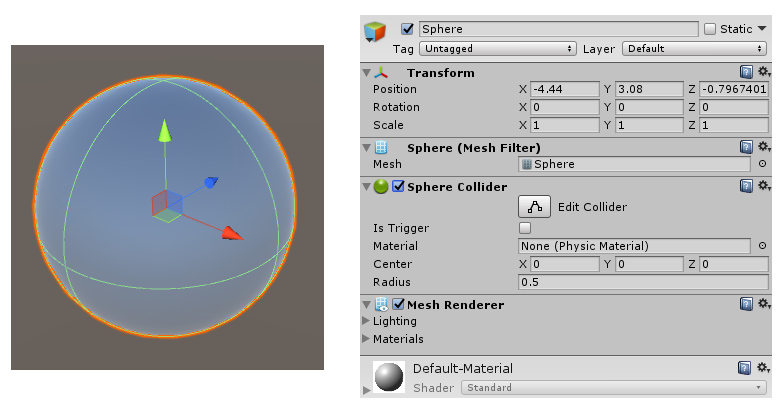
Szintén megadható egy Area típusú fényforrás, amely egy téglalapnyi terület azonos intenzitással világít meg, azonban ez messze nem annyira használatos mint a fentiek.

A fényforrás típusán kívül megadható a fénynek a színe, intenzitása, és egyéb, nem annyira gyakran használt paraméterei.

## GameObject – Játékelem

A Játékszínen belül a kamera és fényforráson kívül létrehozhatunk egyéb Játékelemeket, amelyek a játékunk magját fogják alkotni. Ilyen Játékelem volt például a karakter is, amit létrehoztunk a kamera beállításához. A Játékelemek azok az alapobjektumok, amelyek Unityben a karaktereket, és különböző tárgyakat reprezentálják. Igaz, magukban még nem csinálnak sokat, de hordozóként szolgálnak különböző komponensek számára, amelyek megvalósítják a valódi funkcióját a programnak. Egy komponensük van ami kötelezően szerepel, a Transform, amely megadja, hogy hol helyezkedjen el, mely koordinátákon, majd ezenkívül megadhatóak az X, Y és Z tengely mentén a szögei, és az elem mérete.

Az 5. képen látható egy Gömb Játékelem. Az Inspector menüben találhatóak az adott Játékelemhez tartozó komponensek. Legelsőként látható a kötelező Transform komponens. Ehhez a Játékelemhez azonban tartoznak még további komponensek. Vegyük először a Sphere Collider-t. Ez a komponens mint a neve is mutatja egyfajta ütközőként szolgál. Ha egy másik objektummal érintkezik, akkor ez a komponens segít abban, hogy ne menjenek át egymáson, hanem ütközzenek. Rengeteg más típusú collider is van, és ezeknek a collidereknek tetszőlegesen változtathatjuk a méretüket, helyüket.



5. kép: Gömb Játékelem

Az, hogy milyen típusú collidert érdemes használni, azt nagyban befolyásolja az általunk használt Játékelem alakja, hiszen nagyon ritka, hogy egy gömb alakú objektumnak kocka formájú Collider-je legyen, és fordítva.

### Fizika

A játékfejlesztés egy nagyon fontos eleme a fizika megvalósítása. Minden játékban létezik a fizika valamely módon, nem feltétlenül úgy, mint normálisan.

<https://www.pluralsight.com/blog/film-games/unreal-engine-4-vs-unity-game-engine-best>

<https://docs.unity3d.com/Manual/class-GameObject.html>

<https://docs.unity3d.com/Manual/CreatingScenes.html>

# Bevezetés

Jelen sablon célja, hogy a Mérnök informatikus BSc/MSc, a villamosmérnök BSc/MSc, az gészségügyi mérnök MSc, az gazdaságinformatikus MSc képzések szakdolgozat és diplomaterv munkáihoz sablont nyújtson. A dokumentumot korábbi sablonokból kiindulva Kővári Bence és Szárnyas Gábor állította össze.

A dokumentum a szakdolgozat/diplomaterv szerkesztési elveit követve, bemutatja, hogyan érdemes felépíteni és kidolgozni egy ilyen dolgozatot. Fontos, hogy az itt leírtak nem képviselnek hivatalos egyetemi, sem tanszéki álláspontot, céljuk csupán, hogy mankóként szolgáljanak a nagy munkához.

Az útmutató két nagy egységre bomlik, a 2. fejezet a dolgozattal szemben támasztott tartalmi, míg a 3. fejezet a formai elvárásokat foglalja össze. A leírás feltételezi a magyar nyelvű Word használatát, de természetesen bármilyen más szövegszerkesztő is használható a dolgozat elkészítéséhez.

## Frissítsd a dokumentumot

Ha ezt a dokumentumot nem

* a <http://pub.ext.aut.bme.hu/dip> vagy
* a <https://github.com/FTSRG/thesis-template-word>

címről töltötted le, elképzelhető, hogy nem a legfrissebb változat van nálad. Mielőtt továbbmész, érdemes letöltened a legfrissebb verziót.

## Szakdolgozat vagy diplomaterv

Bizonyára észrevetted, hogy oktatóid és társaid felváltva használják a szakdolgozat és a diplomaterv kifejezéseket. Ennek magyarázata egyszerű. A BSc képzés végén leadandó dokumentumot szakdolgozatnak, az MSc képzés végén leadandót diplomatervnek hívják. Míg a szakdolgozat elkészítésére egy, addig a diplomaterv elkészítésére két félév áll rendelkezésedre, ezért a diplomaterv általában mélyebb/átfogóbb, és kb. 30%-kal hosszabb mű, de ezt leszámítva érdemi különbség a kettő között nincs.

## Témaválasztás

A szakdolgozat/diplomaterv alapvetően arról szól, hogy választasz magadnak egy informatikai kihívást (pl. egy szoftver megtervezését és megvalósítását), ezzel konzulensed segítségével megküzdesz, és az eredményeidet, illetve az azokhoz vezető folyamatot leírod egy 50-70 oldalas dokumentumba.

A témád lehet olyan kihívás is, amit úttörőnek gondolsz, de akár egy rendes „iparos munka” is, amikor is megvalósítasz egy 146. könyvtári nyilvántartó rendszert. Az úttörő, újszerű munka **nem elvárás**, bármit is választasz témának, a te feladatod csak annyi, hogy a tanulmányaid során elsajátított mérnöki eszköztáraddal (illetve szükség esetén annak bővítésével) elemezd a problémafelvetést és adj rá egy elfogadható megoldást. Természetesen fontos, hogy a téma téged is motiváljon, s azt tapasztaltuk, hogy erre az újszerű technológiák alkalmazása, illetve megoldások készítése általában jobban alkalmas.

# A dolgozat szerkezete

Nagyon nehéz általános szabályokat felállítani, vagy kőbe vésett arányokat, oldalszámokat mondani, hiszen minden munka egy kicsit más. Itt mégis megpróbáljuk felvázolni nektek egy általános diplomaterv/szakdolgozat szerkezetét, amitől természetesen el lehet térni, amennyiben a téma ezt megkívánja.

## Fejezetek

A dolgozat számos fejezetből épül fel. Fontos látni, hogy ezeket egyáltalán nem szükséges sorrendben feltölteni. A bevált gyakorlat az, hogy e sablonból kiindulva kitörlöd az ismertető részeket, címsor 1-2-3-mal összerakod a főbb fejezeteket, amikről írni akarsz, kb. olyan terjedelemben, hogy a tartalomjegyzék 1-1,5 oldal hosszú legyen, majd hangulattól és az aktuális munkádtól függően tetszőleges sorrendben elkezded őket tartalommal feltölteni. De nézzük meg, mik is a nélkülözhetetlen szerkezeti elemek.

### Feladatkiírás

Nem sorszámozott oldal, megelőzi még a címet is a dolgozatban. Az elektronikusan beadott változatban ez az oldal kihagyható. A nyomtatott változatban ennek az oldalnak a helyére a diplomaterv portálról letöltött, jóváhagyott feladatkiírást kell befűzni.

### Címoldal

A tanszék, a munka címe, a neved a konzulensed neve és a védés éve automatikus generálódnak a dokumentum metaadataiból. Ezt a **Fájl** menü alatt tudod szerkeszteni, Word 2016-ban a **Minden tulajdonság szerkesztése** gombra kattintva.

### Tartalomjegyzék

A tartalomjegyzéket automatikusan generáld a címsorokból. A tartalomjegyzékben elég az első 3 címsor szintet kezelni (ez a sablon eleve így van beállítva). Figyelj rá, hogy a tartalomjegyzéket (mint minden mezőt) kézzel kell frissíteni, ha változott az elrendezés, vagy új címsorok kerültek a dokumentumba.

### Nyilatkozat

A törlendő részt húzd ki. A nevedet és a dátumot tartalamzó mezők automatikusan generálódnak, nyomtatás előtt frissítsd ezeket. Nyomtatás után ne felejtsd el aláírni.

### Tartalmi összefoglaló

A dolgozat következő eleme egy legfeljebb 1 oldalas magyar nyelvű „Összefoglaló”, illetve az ennek angol fordítását tartalmazó „Abstract”.

A tartalmi összefoglaló készítésének szigorú műfaji szabályai vannak. Bár sokban hasonlít a könyvek hátoldalán látható ajánlókhoz (az ún. blurbökhöz), valójában sokkal több, mint kedvcsináló. Ez alapján az olvasónak el kell tudnia majd dönteni, hogy érdemes-e elolvasnia a dolgozatot, tartalmaz-e számára értékes információkat. Ennek megfelelően nem csak a kedvcsinálás a cél, hanem az is, hogy az eredmények és a kontextus is bemutatásra kerüljön. A tökéletesen összeszedett ajánlásokat [1] foglalja össze. Referenciaként használhatod ennek az útmutatónak az absztraktját is.

Tipp: a tartalmi összefoglalót célszerű a dolgozat végén megírni, hiszen ekkorra áll össze az összes információ, ami szükséges hozzá.

### Bevezetés

Itt kezdődik a dolgozat érdemi része. A bevezetés hossza szakdolgozatban 3-5, diplomatervben 3-7 oldal. Célja a feladat értelmezése, a motiváció leírása, a kontextus megteremtése. Ami a stílust illet, képzeld azt, hogy egy ELTE bölcsésznek próbálod elmagyarázni, mit csináltál és miért. Indíts messziről, pl. beszélj a széles értelemben vett területed (web, mobil stb.) fontosságáról, majd fokozatosan közelíts rá arra a szűkebb problémára amit látsz, s amit megpróbálsz megoldani a dolgozatodban.

Tipikus hiba szokott lenni, hogy motivációként azt írod le, hogy „a konzulensemtől ezt a feladatot kaptam”. Ez nagyon bénán hangzik. Próbáld inkább egy tágabb perspektívából nézni a dolgot és ragadd meg, mitől teszi a te megoldásod/munkád jobbá a világot és erre fókuszálj a bevezetésben.

Kitérhetsz a kapcsolódó technológiák megemlítésére is, de vigyázz, ne ess a 2.2.4. fejezetben bemutatott csapdákba.

A bevezetés praktikus zárása egy olyan alfejezet, ami a dolgozat további szerkezetét ismerteti.

### Irodalomkutatás, technológiák, hasonló alkotások bemutatása

A fejezet terjedelme szakdolgozat esetében kb. 5-10, diplomaterv esetében 7-13 oldal. Ennek a fejezetnek a lénye, hogy bemutassa azt az alapot, amire a munkádat építetted. Ha tudományosabb munkát végzel, itt bemutathatod a szakirodalmat, mások munkáját, eredményeit, amikhez képest a sajátodat majd később meg tudod határozni. Egy tipikus szoftverfejlesztés esetében pedig itt van lehetőséged arra, hogy röviden bemutasd azokat a technológiákat, amikre a munkád során építettél. Fontos, hogy mindig, a témád szempontjából releváns részekről írj, ha például egy webshopot készítesz .NET-ben, teljesen felesleges a reflexióról írnod, annál fontosabbak lehet viszont megemlíteni pl. a jQuery technológiát.

Feltétlenül érdemes kitérned arra, miként álltak a hasonló problémákhoz mások, s bemutatni 2-3 a területen népszerű megoldást, kiemelve azokat az ötleteket, amiket esetleg átvettél belőlük.

### A feladatkiírás pontosítása és részletes elemzése

Tipikus fejezetcím lehet az „Architektúra”, vagy „Tervezés”, a terjedelme 3-5 oldal szakdolgozat és diplomaterv esetén is. A feladat itt már egyáltalán nem mesélgetés, hanem egy objektív elemzésbemutatása arról, hogy mi is a konkrét feladat, amit meg kell valósítani, milyen funkcionális és nem funkcionális követelmények vannak, hogy néz ki a rendszer architektúrája amit terveztél, milyen használati esetek (nem use-case-ek, lásd. 3.6.3.) vannak, s esetleg itt kaphat helyet az adatbázis séma tervezete is. Ez a fejezet viszonylag közel van a „nagykönyv” szerinti követelményspecifikációhoz.

### Önálló munka bemutatása

Ez a blokk egy, vagy több nagyobb fejezetet tartalmaz, mely bemutatja az elkészült megoldás műszaki felépítését, kiemelve az érdekesebb/bonyolultabb megoldásokat és a nem egyértelmű technológiai döntések indoklásait. Terjedelme szakdolgozatban kb. 30, diplomatervben kb. 40 oldal.

Nagyon fontos, hogy szemben az előző blokkal itt nem cél, hogy egy száraz dokumentációt készítsél. Tipikus hiba tud lenni, hogy minden apró részletet és függvényt be akarsz mutatni a programodból. Ennek az eredménye általában csak számos hosszú száraz felsorolás, amit minden jóérzésű bíráló már a második elem után átugrik…

Képzeld el, hogy egy barátodnak kell bemutatnod, a szoftvert amit készítettél. Tervezz meg egy gondolati szálat, ami mentén be akarod mutatni neki. Például kezdheted az adatbázistól, melyen keresztül bemutatod az adatmodellt, majd az erre épülő üzleti logikai réteget, végül pedig a felhasználói felületet. De haladhatsz vízszintesen is, modulról, modulra a rendszer bemutatása során. A lényeg, hogy legyen valamilyen rendező elv, s ezen belül se ***mindent*** próbálj bemutatni, hanem koncentrálj az érdekesebb, nem triviálisabb megoldásokra.

Rossz példa:

*A WordProcessor osztály a következő funkciókat valósítja meg:*

* ***AddWord****: felvesz egy új szót*
* ***RemoveWord****: törli a megadott szót*
* ***ChangeWord****: megváltoztatja a kijelölt szót*
* ***CreateDocument****: létrehoz egy új dokumentumot.*

Ugyanez helyesen:

*Megoldásomban a szavak kezelését a* WordProcessor *osztály végzi. Első használatkor a* CreateDocument *függvényével tudunk egy új dokumentumot létrehozni, majd különböző függvényekkel manipulálni annak tartalmát.*

A fenti példából az is látszik, hogy a programozási elemek (osztálynevek, függvények) elkülönítését hogyan segítheti egy megkülönböztető betűtípus/formázás (következetes!) használata.

### Önálló munka értékelése, mérések, eredmények bemutatása

E fejezet szakdolgozatban 4-5, diplomamunkában 6-7 oldal lehet. Célja, hogy összegezve bemutassa az eredményeidet. Például ha valamilyen algoritmust fejlesztettél, itt mutathatod be mérésekkel, diagramokkal, hogy milyen teljesítményt produkál a különböző megvalósításaihoz, vagy éppen mások implementációihoz képest. Ha egy weboldalt, vagy mobil alkalmazást készítettél, és készültek hozzá automatizált felülettesztek, vagy egységtesztek, akkor azokat itt bemutathatod. Itt írhatsz a szoftver utóéletéről, pl. kikerült-e a Store-ba, elkezdték-e élesben használni, mik vele a tapasztalatok? Előfordulhat az is, hogy nincs külön mondanivalód ezekről, vagy ezeket logikusabb elmondani az előző fejezetben. Ilyen esetekben ez és az előző fejezet összevonható és összemosható egymással.

### Összefoglaló

Dolgozattípustól függetlenül 1 oldalban foglald össze az eredményeidet. E/1-ben és múlt időben. Megterveztem, megvalósítottam, eldöntöttem, leteszteltem… stb.

Itt kaphatnak helyet további max. 1 oldalban a továbbfejlesztési lehetőségek. Milyen hiányosságait látod a rendszernek, illetve milyen lehetőségeket látsz a továbbfejlesztésére?

A dolgozat zárásánál sokan zavarba jönnek, miről lehetne írni. Praktikus tanács itt, hogy foglald össze a személyes tapasztalataidat (amik itt végre bátran lehetnek akár szubjektívak is ☺). Mit tanultál ebből a projektből? Mi a véleményed a technológiákról, amiket alkalmaztál? Beleszerettél az Android fejlesztésbe, vagy egy életre megutáltad? Ilyen és hasonló gondolatok kellemes oldott hangulatú végszót tudnak adni a dolgozatodnak.

### Köszönetnyilvánítások

Őszintén szólva, a legtöbb esetben felesleges blokk, pusztán a konzulensednek megköszönni a segítségét nem szükséges. Neki az a dolga, hogy segítsen. Akkor érdemes ezzel foglalkozni, ha például valamilyen harmadik fél (külső cég, külső konzulens, családod stb.) is említésre méltóan segítette a munkádat és úgy érzed, fontos lenne ebben a formában is kinyilvánítani a háládat.

### Részletes és pontos irodalomjegyzék

Ezt a Word generálja neked. Részletek a 3.10. fejezetben.

### Ábrajegyzék, táblázatjegyzék

Hacsak nem állsz szánalmasan rosszul az oldalszámmal, akkor ilyenekre nincs szükséged.

### Függelék

A függelékek a törzstartalmon kívüli kiegészítések a dolgozathoz. Ide kerülhetnek a nagyobb ábrák, hosszabb példakódok, vagy részletes algoritmus bemutatások, amelyek csak lazábban kapcsolódnak a témához, ezért a dolgozatod törzsében nem mutattad be részletesen. A függelék nem kötelező tartalmi elem, sok esetben teljesen elhagyható.

## Egyéb tartalmi elemek

### Stílus

A szakdolgozat/diplomaterv műfaját tekintve félúton van egy szoftverspecifikáció és egy irodalmi esszé között. Fontos, hogy a leírtak (a nyilvánvalóan szubjektív részeket leszámítva) általában objektív elemzés eredményei legyenek, mérnöki szemléletmódot tükrözzenek. Ugyanakkor kerüld a szoftverspecifikációk száraz stílusát, az értelmetlen felsorolásokat, s próbáld inkább (az objektivitás megtartása mellett) elbeszélő stílusban bemutatni a munkádat.

### E/1

A bíráló számára lényeges, hogy határozottan szét tudja választani az önálló munkádat mások munkájától. Ezért rendkívül fontos, hogy következetesen és sűrűn alkalmazd az egyes szám, első személyű mondatokat és kerüld a passzív mondatok használatát. Ne azt írd, hogy „Ez itt az osztálydiagram”, mivel ez simán jelentheti azt is, hogy az ábrán egy osztálydiagramot látunk, amit a netről letöltöttél, hanem írd azt, hogy „A 4. ábrán bemutat**om** az osztálydiagramot, amit az alkalmazás**om**hoz tervezt**em**”

### Rövidítések

A dolgozatodban alapszabály, hogy **minden** rövidítést legalább az első használatkor ki kell fejteni, és vagy röviden elmagyarázni mit jelent, vagy egy referenciát beszúrni, mely a magyarázatot tartalmazza. Ezt követően a rövidítés már szabadon használható.

Nézd meg a lenti példát! Figyeld meg azt is, hogy forrásként magát a szabványt, és nem a Wikipedia oldalt adtam meg.

Rossz: A weboldalt HTML-ben írtam le.

Jó: A weboldal leírására HyperText Markup Language-t (HTML) használtam [2].

### Technológia megválasztása

Számos esetben előfordul, hogy egy-egy probléma megoldására több különböző technológia áll rendelkezésedre. Amennyiben a feladatkiírásod eleve kikötötte valamelyik technológia (pl. NET, Java stb.) használatát, akkor semmilyen magyarázattal nem tartozol emiatt, használod azt, ami a feladatod volt.

Ha egyébként a feladatkiírás nem tett a technológiai választásra utalást, akkor sincs gond, amennyiben valamely mainstream technológiára esik a választásod, valószínűleg nincs szükség magyarázkodásra (a .NET vs. Java példa ebbe a kategóriába esik).

Amennyiben egy feladat megvalósítására többféle, különböző előnyökkel és hátrányokkal járó technológia áll rendelkezésedre (pl. WinForms vs. WPF) és nem a nyilvánvalóan korszerűbbet választod (pl. WinFormsban akarsz megoldani egy feladatot) akkor mindenképpen szükséges, hogy ezt indokold is (pl. a banki környezetben, ahol dolgozom, ez jelenleg a támogatott fejlesztési nyelv).

Mindenképp kerüld viszont el az amatőr magyarázkodást és a szubjektív érvelést: „Azért választottam az iOS platformot, mert sokkal jobbnak tartom az Androidnál” – az ilyen mondatokat előszeretettel szedik cafatokra a bírálók. Inkább kezeld tényként a technológia választást: „Megoldásomat iOS platformon valósítottam meg”.

# Formázási tudnivalók

Ez a fejezet összeszedi azokat a nélkülözhetetlen elemeket, amelyeket a dolgozat készítése során használnod kell a Word eszköztárából. Feltétlenül olvasd egyszer végig, hátha akad köztük újdonság.

## Általános tudnivalók

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az alapértelmezett betűkészlet a 12 pontos Times New Roman. Ennek használata nem kötelező, de mindenképpen javasolt talpas (serif) betűtípus használata. A dolgozatot 1,15-ös sorközzel célszerű készíteni. A másfeles sorköz még elfogadható, ennél nagyobb használata azonban tilos.

### Margók

A margók mérete mindenhol 2,5 cm. A kötésmargót az **Oldalbeállítás** | **Kötésmargó** menü alatt tudjuk beállítani. Egyoldalas nyomtatás esetén állítsunk be a baloldalon 1 cm-es kötést. Amennyiben a dolgozat kétoldalas nyomtatással készül, a **Több oldal** beállításnál válasszuk a **Margók tükrözése** opciót.

### Számozás

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak. A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni.

### Irodalomjegyzék

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben.

Ez a sablon úgy van kialakítva, hogy a fenti előírásokat eleve kikényszerítse, így általában külön nem kell foglalkoznod velük.

## Stílusok

A Word szövegek egységességét stílusok segítéségével lehet a legegyszerűbben garantálni. Azt javaslom, rögzítsd a **Stílusok** eszköztárt jobb oldalra, így folyamatosan nyomon követheted, hogy milyen stílussal dolgozol, illetve a stílusok közti váltást is sokszor kényelmesebb itt kezelni, mint a fenti eszköztáron.

A dokumentum folyószövegéhez használjuk a **Normál** (angol Word esetén Normal) stílust.



1. ábra: Stílus eszköztár rögzítése

## Címsorok

A fejezetcímek esetén a **Címsor 1–4** (Heading 1–4) stílusokat használjuk. Címsor 4-nél mélyebb fejezetszintre egy ilyen terjedelmű munkában ritkán van szükség, ha ez mégis felmerülne, sokszor inkább a fejezetszerkezetet érdemes átgondolni újból

Tipp: a fejezetcímek aktuális szintjét gyorsan módosíthatod az Alt + Shift + [bal/jobb] billentyűkombinációkkal.

## Másolás, beillesztés

A copy-paste a szép formázás legnagyobb ellensége. Amennyiben a dolgozatodon belül helyezel át részleteket, és a stílusokat szépen következetesen használtad, nem nagyon ronthatsz el semmit. Ha viszont külső forrásból másolsz be szöveget, érdemes minden esetben a beillesztést formázások nélkül végezni. Ezt az opciót a Word mindig felkínálja, a beillesztést követően.



2. ábra: Beillesztés formázás nélkül

Amennyiben olyan szövegrészt mozgatsz kivágás/beillesztés párossal, melyben automatikus sorszámozás volt (irodalomhivatkozás, ábra, táblázat), akkor figyelj rá, hogy a sorszámok nem frissülnek automatikusan.

## Mezőfrissítés

A dolgozatodban számos automatikusan kalkulált mező szerepel (ábra sorszámok, irodalomhivatkozások, tartalomjegyzék stb. Ezeket a Word nem tartja automatikusan karban. A mezők értékét érdemes rendszeresen (de a beadás/nyomtatás/PDF generálás előtt feltétlenül) frissíteni. Ehhez jelöld ki a teljes dokumentumot (Ctrl + A), majd az előugró menüből válaszd a „mezőfrissítés” opciót.

## Helyesírás

A rossz helyesírásra nincs mentség. E fejezetben összeszedem a leggyakrabban látott hibákat, amiknek elkerülésére érdemes odafigyelni. Ettől függetlenül melegen ajánlom, hogy a kész dolgozatod olvastasd át egy barátoddal/családtagoddal, hogy az apróbb, megbúvó hibákat is kiszűrd.

### Elgépelések

Ez mindenkivel megesik. Egy jó helyesírás ellenőrző az elgépelések nagy részét megfogja.

### Egyeztetés hiánya

Az elírások egyik leggyakoribb formája az egyes szám/többes szám egyeztetésének hiánya mondatrészek között, mint például itt: „Petike és a barátnője elment**ek** a boltba és hoz**ott** egy kiló kenyeret”. Ezek a mondatok főleg az utólagos átfogalmazások, belejavítások során keletkeznek, legjobb védelem ellenük az utólagos átolvasás.

### Külföldi szavak, kifejezések

Az idegen szavakkal csak a baj van, nehéz őket leírni, ragozni, kiolvasni, vagyis úgy általában használni. Az általános jó tanácsom, hogy amennyiben csak lehetséges, akkor **magyar vagy magyarosított írásmódú szakkifejezéseket használj**, és a könnyebb olvashatóság érdekében mindig **kerüld az idegen szavak ragozását**.

Ez sokszor nem intuitív szabály, hiszen a beszélt informatikai szlengben előszeretettel használunk olyan fordulatokat, mint „property-vel”, „compile-olom” stb., ezek helyett a „tulajdonsággal”, illetve a „lefordítom” százszor jobban néznek ki papíron, könnyítik az olvasást és csökkentik a hibázás esélyét. Néha (például a tulajdonneveknél) nehezen elkerülhető a ragozás, de kis leleményességgel, ügyes fogalmazással itt is megoldható a probléma. Pl. „Apache-csal” helyett írjuk azt, hogy „Apache webszerverrel”.

Néhány tipp a szoftverfejlesztőknek: property = tulajdonság, event = esemény, method = metódus/függvény, debug = hibakeresés, file = fájl.

Ha mégis belevágsz az angol kifejezések használatába, lelked rajta, de kérlek, legalább ezt a minimális szabályhalmazt olvasd át előtte [3].

### Stb.

Az „stb.” rövidítés azt jelenti „és a többi”, mivel „és” előtt felsorolásoknál nem teszünk veszőt, az „stb.” elé sem teszünk soha. Pl.: asztal, szék, lámpa stb.

### Helyesírás-ellenőrző

Személyes ízlés kérdése, hogy milyen eszközzel (LaTeX/Word), illetve ezen belül milyen nyelvű változattal dolgozol, ugyanakkor azt **meg kell oldanod, hogy legyen mellé magyar nyelvű helyesírás-ellenőrződ**. E nélkül dokumentumot szerkeszteni olyan, mint papíron programozni.

Figyelj rá, hogy a Word megengedi, hogy többféle nyelv is legyen egy dokumentumban, illetve a kívülről beillesztett szövegek nyelvét néha a forrásoldal/forrásdokumentum határozza meg. Érdemes ezekben az esetekben kikényszeríteni, hogy a teljes szakasz/dokumentum nyelve magyar legyen. Érdemes néha tesztelni a helyesírás ellenőrzőt. Ha pl. beírod, hogy  és nem húzza alá pirossal, akkor valószínűleg valami nem stimmel a beállításaiddal.



3. ábra: Ellenőrzés nyelvének megadása

## Képek

A dolgozatodban valószínűleg számos ábrára lesz szükséged, ezek használatánál azonban érdemes pár dologra odafigyelned.

### Beszúrás, formázás

A képhez használd a **Kép** stílust.

Képaláírást a képen jobb gombbal kattintva a Képaláírás beszúrása… opcióval adhatod hozzá, így az automatikusan **Képaláírás** (Caption) stílusú lesz és kezeli a sorszámozást is.



4. ábra: Példa képaláírásra

Az ábra sorszáma mellé mindig érdemes rövid magyarázatot is fűzni. Érdemes tudni, hogy Wordben számos beállítási móddal lehet képet beszúrni. Tapasztalni fogod, hogy számodra a legpraktikusabb a „szöveggel egy sorba” elrendezés lesz, mivel így szépen együtt marad az ábra a szöveggel. Oldaltörésekkor az ábrák méretének módosításával tudod megoldani, hogy ne maradjanak nagy üres felületek a dolgozatodban.

### Képminőség

A raszteres (tehát nem vektorgrafikus) képek használata különös körültekintést igényel. Ezek kiválóan néznek ki a 90 dpi-s monitorodon, ám a 600/1200 dpi-s nyomtatókon kinyomtatva rendkívül bénák lesznek a szép, pixelmentes szövegek és vektorgrafikus ábrák mellett. Hogy lásd, mire számíthatsz, nézzük meg az előző ábrát 5-szörös nagyításban.



5. ábra: A raszteres képek nyomtatásban csúúúúnyák lesznek

Ha tehát lehetséges, használjunk vektorgrafikus ábrákat, vagyis a diagramokat, forráskódot stb. ne képenyőképeken keresztül, hanem közvetlen copy-paste megoldással másoljuk át a dokumentumunkba.

Ha elkerülhetetlen a raszteres képek használata, akkor próbáljunk meg minél magasabb felbontású képet berakni. Bár a képernyőn nem feltétlenül látszik majd a különbség, de a nyomtató ki tudja majd használni a magasabb felbontású képet. A 6. ábra például igen magas felbontású, amit láthatsz is, ha felnagyítod.



6. ábra: Ez az ábra 460 × 750 pixel felbontású

Figyelj rá, hogy a PDF-be mentéskor a Word (2013) alapértelmezésben jelentősen lerontja a képek minőségét, sőt, a vektorgrafikus ábrákból is rasztereset csinál. Ha a nyomtatást PDF alapján akarod csinálni, érdemes külső szoftvert használni a Word 🡪 PDF konverzióra, vagy alaposan eljátszani a Word beállításaival.

## Táblázatok

Bizonyos adatokat táblázatos formában érdemes ábrázolni. Például: „A probléma megoldására a két fő megoldási irány az A és a B opció, melyek különböző funkciókat támogatnak (lásd 1.1. táblázat).”

|  |  |  |
| --- | --- | --- |
|  | A opció | B opció |
| 1. funkció | + | + |
| 2. funkció | + | − |
| 3. funkció | − | + |
| 4. funkció | − | − |

1.1. táblázat. Példa táblázat feliratára

## Kereszthivatkozások

Amennyiben szeretnél egy ábrára, vagy korábbi fejezetre hivatkozni, használd a Word „Hivatkozás/Kereszthivatkozás” menüpontját. Kerüld az így beszúrt szövegek közvetlen ragozását, hiszen az könnyen változhat, amennyiben a forrás változik. Fejezetekre, ábrára sose a teljes szövegükkel, csupán a sorszámukkal („Csak címke és szám”) hivatkozz. Ezt szemlélteti a 7. ábra.



7. ábra: Ábrahivatkozás beszúrása

Tipp: érdemes elkerülni az „előző oldalon”, „következő oldalon” fordulatokat, ugyanis az ábrák végső helyzete a tördelés során még megváltozhat.

Amennyiben az „ábra” szót szeretnénk ragozni (pl. „lásd az x. ábrán”), akkor a fenti megoldás nem alkalmazható. Helyette jelöljük ki a képaláírásban a sorszámot (pl. „1.1.”), kattintsunk a **Könyvjelző** gombra, majd hozzunk létre egy könyvjelzőt (pl. „bmelogo” névvel). Ezután a **Kereszthivatkozás** gombra kattintva a **Hivatkozástípus**t állítsuk **Könyvjelző**re és válasszuk ki a **bmelogo** könyvjelzőt. Sajnos azonban ez a megoldás sem tökéletes: a Word ugyanis nem támogatja a megfelelő határozott névelő beszúrását. Egy újabb ábra beszúrása esetén a névelőket egyesével kell ellenőriznünk – például ha a dokumentum elejére beszúrunk egy új ábrát, akkor a korábbi „az 1. ábrán” hivatkozásból „az 2. ábrán” lesz.

## Irodalomhivatkozások

Az irodalomhivatkozások kezelésére a Word egy kényelmes és jól használható funkciót kínál. Amikor dolgozatodban egy külső műre, weboldalra, könyvre, előadásra stb. szeretnél hivatkozni, használd a „hivatkozás/források kezelése” menüpontot. Itt felveheted az egyes műveket, illetve a hozzájuk kapcsolódó szerzői adatokat. Folyóiratcikkeknél a szerzők mellett szerepeljen a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal szerzőjét és címét (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Az egyes forrásmunkákra a dolgozat szövegében lehet hivatkozni a „hivatkozás/idézet beszúrása” menüponttal. Figyelj rá, hogy az IEEE[[1]](#footnote-1) stílust használd.



8. ábra: Idézetek és irodalomjegyzék

### Pozícionálás

Az irodalomhivatkozások a szövegtörzsben, ábrák szövegében és táblázatokban is előfordulhatnak, de **fejezetcímekben soha**. Amennyiben egy adott forrás egy egész bekezdésre/fejezetre vonatkozik, akkor is elég, ha az első mondat/első bekezdés után megemlítjük. Szó szerint idézett szöveg esetén az idézet végén, külső forrásból átvett ábra esetén a képaláírásban hivatkozzunk a forrásra.

### Mikor kell hivatkoznom?

Minden külső forrásból átvett képnél, szövegrésznél, olyan szövegeknél, melyek megfogalmazásához külső forrásokat használtál, illetve olyan állítások/technológiák/algoritmusok megemlítésénél, melyek nem feltétlenül egyértelműek egy átlagos műveltségű olvasó számára. Nagyságrendileg egy szakdolgozatban átlagosan 10-20, egy diplomatervben átlagosan 20-30 külső forrást illik megemlíteni. A források megválasztásánál, ha lehet, törekedjél a nyomtatott források keresésére, ezek lehetnek folyóiratcikkek, könyvek, könyvfejezetek, útmutatók, egyetemi jegyzetek stb., és csak akkor hivatkozz weboldalakra vagy Wikipediára, ha ilyet nem találtál.

### Irodalomjegyzék megvalósítása

Az irodalomjegyzék kezelése többféleképpen is megoldható, az alábbiakban két egyszerű módszert ismertetünk.

#### Kereszthivatkozásokkal

A kereszthivatkozásokkal történő irodalomjegyzék egy megfelelően formázott felsorolás, melynek egyes elemeire (bekezdéseire) mutatnak hivatkozások. Jelen dokumentum ezt a megközelítést alkalmazza.

Az Irodalomjegyzékben szereplő hivatkozásokat **Irodalomjegyzék bejegyzés** stílussal formázzuk, a címüket pedig a **Kiemelés** stílussal emeljük ki.

A szövegbe a hivatkozásokat a **Kereszthivatkozás beszúrása** (Insert cross-reference) funkcióval helyezzük el (példa egy így beszúrt hivatkozásra: [1]), így azok később frissíthetők a hivatkozások átrendezése esetén (lásd 2. fejezet).

#### Források kezelése

A kereszthivatkozások alternatívája, hogy a hivatkozott műveket először felvesszük a szerkesztőprogram adatbázisába. Új műveket a **Hivatkozás** fülön a **Források kezelése** alatt az **Új…** gombbal vehetünk fel. A szerzőket érdemes a **Szerző** mező mellett található **Szerkesztés** gomb használatával felvenni.

Az irodalomjegyzéket az **Irodalomjegyzék** gomb alatt az **Irodalomjegyzék** opcióval szúrhatjuk be a dokumentumba. A hivatkozások stílusa a **Stílus** gomb alatt állítható be, a javasolt stílus az **IEEE**.

## Word tippek és trükkök

### Navigációs ablak

Mivel a fejezeteket nagy valószínűséggel nem sorrendben fogod tartalommal feltölteni, érdemes bekapcsolni a navigációs ablakot (nézet/navigációs ablak). Ez gyors és egyszerű keresési és ugrálási lehetőséget biztosít a fejezetek közt



9. ábra: Navigációs ablak

### Megjegyzések

A véleményezés/megjegyzés menüponttal megjegyzéseket fűzhetsz a szöveg tetszőleges részéhez. Ezt konzulensed előszeretettel fogja majd használni az észrevételei megfogalmazásához, de neked is jól jöhet, hogy megjelöld azokat a részeket, ahova még vissza kell térned. Fontos, hogy nyomtatás előtt az összes megjegyzést töröld, vagy külön állítsd be, hogy a megjegyzések ne jelenjenek meg a nyomtatásban.

### Korrektúra

Konzulensed, vagy mások, akiket a dolgozat átolvasására megkérsz legegyszerűbben a korrektúra funkció használatával tudják úgy javítani a dolgozatod, hogy a javításokat te is lásd. Az átolvasáshoz tehát érdemes bekapcsolni a „változtatások követése” funkciót és „Minden korrektúra” mutatására beállítani a rendszert.

Amikor korrektúrázott változatot kapsz vissza valakitől, használd a „Véleményezés ablak”-ot, így egyetlen javítás sem kerülheti el a figyelmedet, s így mindegyiket elfogadhatod vagy elvetheted, igény szerint.



10. ábra: Ellenőrizd, hogy minden bekarikázott funkciót ismersz-e!

### Gyorsbillentyűk

A Word lehetőséget ad tetszőleges funkció gyorsbillentyűre kötésére. Használd a „billentyűparancsok testreszabása” funkciót, hogy gyorsbillentyűt rendelj a leggyakrabban használt gombokhoz.

Érdemes továbbá gyorsbillentyűt rendelni az alapvető stílusokhoz is. Ebben a dokumentumban hozzárendeltük az Alt + K kombinációt a „Kép” stílushoz.

## Kódrészletek

Érdekesebb és bonyolultabb programozási megoldásainkat bátran illusztrálhatjuk kódrészletek beszúrásával. Fontos, hogy a beillesztett kódrészlet mérete álljon arányban annak a fontosságával, tehát ritkán érdemes egy „bekezdésnyi” kódnál többet beszúrni egyszerre. Amennyiben egy bonyolultabb (akár több oldalas) algoritmust szeretnénk bemutatni, annak a kódját érdemesebb függelékbe rakni.

### Formázás

A kódrészletek formázásánál kerüljük a helypazarlást, illetve próbáljuk megelőzni az olvashatóságot rontó sortördelést, akár a forráskód módosításának árán is. Két praktikus tipp Visual Studióhoz:

Másolás előtt érdemes a behúzások mértékét 4-ről 2 karakterre csökkenteni, majd a másolandó kódrészletet újraformázni (Ctrl + K, Ctrl + D), alternatívaként persze utólag is kitörölgethetjük a felesleges szóközöket a kódból.



11. ábra: A behúzások mértékének csökkentése Visual Studióban

A másolandó kódrészlet kijelölésekor tartsuk lenyomva az Alt billentyűt. Így a valóban releváns részekre korlátozhatjuk a kijelölést.



12. ábra: Kódblokk kijelölése Alt + egérkurzor segítségével

Kódrészletek beillesztésére semmiképpen ne képeket használjunk, hanem a közvetlenül beszúrt kódrészletekkel dolgozzunk. A Visual Studióból kimásolt kódok alapvetően jól néznek ki, további szépítésre használjuk a **Kód** stílust.

static void Main(string[] args)

{

var ci = new CultureInfo("en-us");

ci.NumberFormat.CurrencySymbol = "";

Thread.CurrentThread.CurrentCulture = ci;

Console.WriteLine(ci);

Console.WriteLine("{0:c}", 5.66);

}

### Irodalomjegyzék

Az Irodalomjegyzékben szereplő hivatkozásokat **Irodalomjegyzék sor** stílussal formázzuk, a címüket pedig **Irodalomjegyzék forrás** stílussal emeljük ki.

A szövegbe a hivatkozásokat a Kereszthivatkozás beszúrása (Insert cross-reference) funkcióval helyezzük el, így azok automatikusan frissülnek a hivatkozások átrendezésekor.

## Utolsó simítások

Miután elkészültünk a dokumentációval, ne felejtsük el a következő lépéseket:

* Kereszthivatkozások frissítése: miután kijelöltük a teljes szöveget (Ctrl + A), nyomjuk meg az F9 billentyűt, és a Word frissíti az összes kereszthivatkozást. Ilyenkor ellenőrizzük, hogy nem jelent-e meg valahol a „Hiba! A könyvjelző nem létezik.” szöveg.
* Dokumentum tulajdonságok megadása: a dokumentumhoz tartozó metaadatok kitöltése (szerző, cím, kulcsszavak stb.). Ez Word 2013 alatt a **Fájl** | **Információ** | **Tulajdonságok** | **Dokumentumpanel megjelenítése** gombra kattintva érhető el, míg Word 2016-ban a **Fájl** | **Információ** | **Tulajdonságok** | **Minden tulajdonság megjelenítése** alatt**.**
* Kinézet ellenőrzése PDF-ben: a dokumentum elkészítése után feltétlenül ellenőrizzük a kapott PDF dokumentumot is.

# Köszönetnyilvánítás

A köszönetnyilvánítás nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

# Irodalomjegyzék

1. Jeney Gábor, Hogyan néz ki egy igényes dokumentum? Néhány szóban az alapvető tipográﬁai szabályokról, Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék, Budapest, 2007. május 9., online: <http://mcl.hu/~jeneyg/foliak.pdf>
2. William Strunk Jr., E. B. White, The Elements of Style, Fourth Edition, Longman, 4th edition, 1999.
3. Levendovszky, J., Jereb, L., Elek, Zs., Vesztergombi, Gy., Adaptive statistical algorithms in network reliability analysis, Performance Evaluation – Elsevier, Vol. 48, 2002, pp. 225-236
4. National Instruments, LabVIEW grafikus fejlesztői környezet leírása, <http://www.ni.com/> (2014. aug.)
5. Fowler, M., UML Distilled, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004

# Függelék

A függelék szövege.

1. IEEE (ejtsd „I triple E”): Institute of Electrical and Electronics Engineers [↑](#footnote-ref-1)