Színház- és Filmművészeti Egyetem Doktori Iskola

KÓDBA ZÁRT TÉR

INTERAKTÍV WEB 3D ANIMÁCIÓK TERVEZÉSÉNEK ÉS ALKALMAZÁSÁNAK MÓDSZEREI BROADCAST KÖRNYEZETBEN

DOKTORI ÉRTEKEZÉS

Balogh Áron

2023.

Témavezető: Dr. habil. M. Tóth Géza

Tartalomjegyzék

1. Bevezetés

A háromdimenziós képmegjelenítés folyamatosan megújuló tervező-módszertani és adaptációs mechanizmusokat kínál a 3D animációs fejlesztők és a kreatív alkotók számára. A korszerű munkafolyamatok egyre inkább megkerülhetetlen kérdése a konvergens 3D mozgóképalkotás problematikája, az elkészített médiatartalmak valósidejű transzplantálása eltérő vizuális platformok irányába.

Disszertációmban azon 3D szerzői eszközkészletek újra-definiálásának feltétel- és alkalmazás rendszerét vizsgálom, melyek az animációs eljárásokban jelentenek változást. A térbeli mozgókép tervezésének interaktív módszerei mellett, elemzésem fókuszában a WebGL és WebGPU alapú technológia feltérképezése, a digitális web 3D szkript-nyelveinek szintaktikai elemzése, ismertetése áll.

1.1. Célkitűzés

Doktori disszertációm célkitűzése a valósidejű, interaktívan vezérelhető WebGL - WebGPU 3D animációk művészi és technológiai vonatkozásainak vizsgálata. Értekezésem célja, hogy feltárja az online környezetben alkalmazható 3D vizualizációs adatbázis létrehozásának módszerét, amely lehetővé teszi a 3D tervező- és animációs szoftverekben kialakított CGI jelenetek adaptálását valósidejű broadcast felületekre.

DLA munkásságom fókuszában olyan adatalapú automatizált mozgóképes eljárások kutatása és létrehozása áll, melyek segítségével a hagyományos háromdimenziós tervezői programok objektumai, animációs jelenetei és előre definiált paraméterrendszerei olvasható adatstruktúrákkal továbbíthatók, és interaktívan felhasználhatók háromdimenziós webgrafikai könyvtárak, továbbá broadcast keretrendszerek élőtechnikás adáskörnyezetében. Elemzésem kiemelt célja, hogy bemutassam azokat a ma elérhető mozgóképes eszközöket és vizuális programnyelveket, melyek a legalkalmasabbak az animátor számára a valósidejű 3D tervezéshez és megjelenítéshez. Kutatásom fontos szempontja, hogy az általam leírt metódusok a hazai televíziós technolóigai környezetben, továbbá a virtuális-, a kevert- és a kiterjesztett valóság színtereiben is vezérelhető mozgóképes tartalmat biztosítsanak.

A kutatás kreatív tervezői és fejlesztői szempontból vizsgálja azt a gyorsan változó interdiszciplináris szakterületet, amelyet a platformfüggetlen generatív 3D tartalomgyártás módszerei és az interaktív vizualizációs eszközök határoznak meg. Bemutatom a magyar 3D

filmes technikatörténet legnagyobb releváns eredményeit, a modern valósidejű 3D animációs technológia aktuális pillanatképét, vizsgálom az adásgrafika változékony helyzetét és állapotát, elemzem a jelenleg elérhető technológiai módszereket, valamint bemutatom a megvalósításhoz szükséges módszertanokat.

DLA mestermunkám elkészítésével célom, hogy Magyarországon elsőként hozzak létre saját fejlesztésű nyílt forráskódú 3D WebGPU vizualizációs alkalmazást, mely segítségével a mozgóképes szakemberek, valósidőben, platformfüggetlenül publikálhatják térbeli modelljeiket, mozgóképes alkotásaikat.

Kiemelt célom, hogy kutatásom interdiszciplináris jellege révén elősegítsem a különböző szakterületek közötti együttműködést, és munkásságommal támogassam innovatív technológiák alkalmazását a broadcast és online 3D vizualizációs területeken.

Doktori értekezésemmel és elkészített mestermunkámmal összegzem a legkorszerűbb módszerekről és technológiákról szerzett ismereteimet és tapasztalataimat. Ezeket – folyamatosan fejlesztve – nemcsak alkotói, hanem oktatói munkámban is alkalmazni szeretném, elsősorban a jövő 3D tervezőinek, fejlesztőinek, média design és multimédia szakembereinek képzésében. Ezen túlmenően, a kutatás eredményeit és tapasztalatait szeretném megosztani szélesebb körben is, inspiráció gyanánt a valósidejű 3D vizualizációs technológiák további alkalmazási lehetőségeire a terület iránt érdeklődő alkotók számára.

1.2. A témaválasztás személyes okai

Az elmúlt másfél évtizedben valós idejű 3D broadcast engine-ek fejlesztési környezetében, automata grafikai template-rendszerek tervezésével, továbbá interaktív mozgóképes módszerek kutatásával foglalkoztam regionális és országos televízióknál. Senior fejlesztőként, később Solution Architect-ként lehetőségem volt részt venni a magyar televíziózás legnagyobb adatalapú műsorainak elkészítésében. Grafikus applikációim megjelenhettek a legnézettebb hazai műsorok adásaiban, dolgozhattam a Magyar Televízió interaktív alkalmazás-rendszereinek kidolgozásában, kialakításában. Részt vehettem a világ legnagyobb broadcast vizualizációs konferenciáin, továbbá adásszoftverek, és broadcast rendszerek fejlesztésével kapcsolatos nemzetközi workshopjain. Munkám mellett több éve egyetemi oktatóként is dolgozom, ahol a 3D vizualizációs eszközöket, 3D animációs technológiát és kreatív mozgóképes eljárásokat tanítok. Bár az első 3D animációmat 18 évvel

ezelőtt adta a televízió, a mozgóképes, filmes és animációs technológiák iránt azóta is elkötelezettnek érezem magam.

Rendkívül fontosnak tartom a 3D művészeti- és a fejlesztői diszciplína egyidejű ismertetését, az új digitális lehetőségek alkalmazását, kutatási eredményeim későbbi publikálását, szakmám képviseletét. Disszertációm elkészítésével személyes célom, hogy összegezem a Színház- és Filmművészeti Egyetem Doktori Iskolájában megkezdett kutatásaimat, továbbá, hogy akadémiai szinten képviselni tudjam a 3D fejlesztői szakmát a megújuló valós idejű engine-ek technikai kihívásainak interaktív környezetében.

1.3. A tárgyalt alapfogalmak értelmezése multimédiatervezési aspektusból

A 3D JavaScript könyvtárak többsége platformfüggetlen grafikus-fejlesztői felületet biztosít az interaktív mozgóképes- és animációs tervezők, a broadcast fejlesztők, továbbá a kreatív Front-End szakemberek számára. A technológia megfelelő környezetet nyújt vezérelhető vizualizációk kialakításához, a térbeli- és animációs televíziós műsorelemek fejlesztésénél, továbbá az interaktív CG, CGI rendelésekor.

A weben található 3D WebGL alkalmazások jelenlegi szabványának alapja az OpenGL keretrendszer.

1.3.1. OpenGL

Az Open Graphics Library egy interaktív, 3D képalkotásra alkalmas platform- és hardverfüggetlen rendszer, amely API-ján *(eseménykezelő függvényeken)* keresztül lehetőséget teremt a real-time renderelés, azaz a valósidejű képszámítás időveszteség nélküli grafikai folyamatát biztosítani[1]. A mobil alkalmazás- vagy böngészőalapú 3D megjelenítés-technológiák alapját is ez adja. OpenGL ES (Embedded Systems) változatát beágyazott rendszerekre tervezték, melyek nem rendelkeznek az asztali gépekhez hasonló erőforrással.[2] A Web Graphics Library JavaScript API-ja a GPU-k, azaz a grafikus processzorok erejét használja ki.[3]

---

[1] Khronos Group: Introduction to OpenGL. *OpenGL.org,* 2023. tavasz (https://www.opengl.org/wiki/Introduction_to_OpenGL) Utolsó letöltés: 2022.12.27.
[2] Uo.
[3] Johns, P.: WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. *Addison-Wesley Professional* 2014. tavasz (http://uniguld.dk/wp-content/guld/DTU/webgrafik/0321902920_WebGL.pdf) Utolsó letöltés: 2021.04.04.

Az OpenGL segítségével szabványos felületen keresztül érhető el a grafikus vezérlő[4]. Nem önálló grafikai modellező- és térbeli rajzoló alkalmazás, hanem egy rendszer, mely erre -*függvénykönyvtárai segítségével*- lehetőséget teremt[5]. Az OpenGL utasítások hatékonyan lefordíthatók olyan parancsokká, amelyeket a GPU képes megérteni és végrehajtani. A videókártyát vezérelve *(GPU API)*, a legegyszerűbb rajzeszközök, geometriai-primitívek *(például pontok, vonalak, háromszögek)* segítségével képes hatékonyan, és rendkívüli sebességgel a 3D alakzatok kirajzolására, megjelenítésére. Az OpenGL API végrehajtandó programokat is tud küldeni a GPU-nak.[6] A csúcs koordináták geometriai transzformációs parancsait, és színkalkulációs folyamatait a *Vertex shader* parancs végzi, míg a grafikai modellek pixelenkénti számításait pedig az ún. *Fragment shader*. Az OpenGL shadereket GLSL-ben *(OpenGL Shading Language)* írják.

OpenGL-t alkalmazzák a virtuális valóság megjelenítő rendszeriénél, broadcast vizualizációs környezetben, valósidejű filmes- és animációs programokban. A technológiát felhasználják a játéktervezés során, a virtuális valóság, valamint a kiterjesztett valóság eszközein, CAD alkalmazásokban, tudományos- és adatvizualizációs környezetben. Az OpenGL segítségével olyan komplexitású 3D modellek, animációk jeleníthetők meg, melyekben valósidőben állíthatók a digitális tér alkotóelemei: a 3D fény, és árnyéktulajdonságok, a részecskedinamika szimulált paraméterei, vagy épp a valósidejű textúrák.

A OpenGL technológiát a real-time szoftverek mellett a web is alkalmazza, így egyes böngészőalapú 3D animációs és interaktív vizualizációs technikáknak is ez az alapja.

4 Dr. Fekete Róbert Tamás - Dr. Tamás Péter - Dr. Antal Ákos - Décsei-Paróczi Annamária: 3D megjelenítési technikák. *BME-MOGI,* 2014. (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2023.11.10.

5 Juhász Imre: OpenGL. *mobiDIÁK Könyvtár,* 2003. tavasz (http://193.6.8.43/segedlet/dokumentumok/OpenGL/OpenGL.php). Utolsó letöltés: 2021.12.06.

6 Khronos Group: Introduction to OpenGL. *OpenGL.org,* 2023. tavasz (https://www.opengl.org/wiki/Introduction_to_OpenGL) Utolsó letöltés: 2022.12.27

1.3.2. **WebGL 1** *(OpenGL ES 2.0),* **WebGL 2** *(OpenGL ES 3.0),*

A WebGL *(Web Graphics Library)* olyan OpenGL-en alapuló JavaScript API[7, 8], amely nagy teljesítményű, platformfüggetlen, interaktív 2D és 3D grafikák megjelenítését teszi lehetővé beépülő modulok használata nélkül, bármely kompatibilis böngészőben[9,10]. A JavaScript nyelv WebGL API funkcióin keresztül képes kezelni a 3D grafikai elemeket, továbbá olyan számításigényes feladatokat, mint a 3D animációk, vagy a térbeli fizikai szimulációk. Ezek az alkalmazások, felületek nagy teljesítményű vektor- és mátrix matematikai folyamatokat igényelnek, melyet a JavaScript GL könyvtárak *(glMatrix)* [11] segítségével biztosít.

OpenGL-en keresztül, a számítógép grafikus feldolgozó egységét *(GPU: Graphics Processing Unit)* kihasználva a WebGL képes valós idejű, vezérelhető tartalmat megjeleníteni a DOM *(Document Object Mode)* interfészen, felhasználva a HTML5 *canvas* elemet. A vertexeket összekötő élek és felületek optimalizált rajzolatát, rederelt képpont-párjait a shaderek segítségével definiálja. A shaderek programozható leíróállománya tartalmazza a végső megjelenítéshez szükséges információkat *(pl.: térbeli transzformációs adatok, pozíció- és színadatok, kamera koordináták).*

Az OpenGL első verziója 1990-es évek elején készült. Az azóta eltelt időben a grafikai igények lényegesen megváltoztak, komplexebbek lettek, és a grafikus illesztőprogramoknak pedig egyre bonyolultabb feladatokat kell elvégeznie. A Khronos Group 2016-aban publikálta a *Vulkan*-t[12], mely teljesen új API lett. Emellett további technológiák is fejlődtek, melyek a Vulkan-nal együtt már a modern hardveres technológiák működési modelljéhez illeszthetők. Az Apple a Metal-lal *(iOS, macOS rendszerekhez),* a

[7] Khronos Group: Introduction to OpenGL. *OpenGL.org*, 2023. tavasz (https://www.opengl.org/wiki/Introduction_to_OpenGL) Utolsó letöltés: 2022.12.27.
[8] Juhász Imre: OpenGL. *mobiDIÁK Könyvtár,* 2003. tavasz (http://193.6.8.43/segedlet/dokumentumok/OpenGL/OpenGL.php) Utolsó letöltés: 2023.03.02.
[9] Johns, P. (2014). WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. Addison-Wesley Professional. (http://uniguld.dk/wp-content/guld/DTU/webgrafik/0321902920_WebGL.pdf) Utolsó letöltés: 2021.04.04.
[10] Leggyakrabban használt modern böngésző: Google Chrome, Edge, Safari, Opera, Android Browser, UC Browser, Samsung Internet. Aktuális verziószámok az alábbi linken ellenőrizhetők: https://caniuse.com/WebGL
[11] Brandon Jones, Colin MacKenzie: *glMatrix API* (https://glmatrix.net/). Utolsó letöltés: 2021.12.06.
[12] Khronos Group Inc.: Vulkan. *Khronos*, 2023. (https://www.vulkan.org/) Utolsó letöltés: 2023.03.21.

Microsoft a *DirectX 12-vel* állt elő.[13] A WebGL azonban csak részben tudott alkalmazkodni a megújuló technológiákhoz, így szükség volt *gépközeli* webes API-ra, amely ténylegesen platformfüggetlen *(Vulkan, Metal, vagy DirectX 12)*. Bár WebGL 2014 *(teljes böngészőtámogatottsága)* óta a 3D web origójaként tekinthető[14], az új technológia webes térhódítása így rövidtávon garantálható lett.

1.3.3. WebGPU

A WebGPU a legújabb webes 3D technológia, amely a webes grafikai tartalmak létrehozására és megjelenítésére szolgál. Az API fejlesztését 2021-ben kezdték, így még rendkívül újnak számít az online 3D technológiák között. A WebGPU teljes támogatást nyújt a GPU-n történő számítások végrehajtásához, így a 3D internet vizualizációs eszközkészlete jelentős változás előtt áll. A WebGPU annyira újnak számít, hogy a dolgozat írásának utolsó pillanatáig piaci bevezetése előtt állt. A WebGPU jelenleg egy rendkívül frissen publikált, aktív fejlesztés alatt álló technológia, mely a jövő 3D internetének egyedülálló kulcsszereplője lesz.

A web 3D technológiákra épülő segédkönyvtárak kepések még tovább egyszerűsíteni a tervezői- és ábrázolói feladatokat. Ezek a 3D könyvárak optimális választást, illetve munkafelületet jelentenek a mozgóképes szakemberek számára is.

1.3.4. Segédkönyvtárak

A 3D Web segédkönyvtárak *(pl. Three.js, Babylon.js, A-Frame, PlayCanvas)* kényelmi-réteget, vagy újabb API-t biztosítanak a fejlesztőknek. Ennek hiányában ugyanis a natív GPU-ra épülő grafikus könyvtárak alacsonyszintű fejlesztői környezete rendkívüli kihívásokat jelentene a mozgóképes alkotók vagy animációs tervezők *többségének*[15] *[Script.WebGLCube]*. OpenGL használata esetén rendkívüli fejlesztési komplexitással kell számolni; gépközeli kódokat kell írni, és meg kell érteni a renderelés *(valamint a tesszelláció)* bonyolult folyamatát.

---

[13] Gullen, Ashley: A brief history of graphics on the web and WebGPU. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/brief-history-graphics-web-1517) Utolsó letöltés: 2022-07-01.

[14] 2011 óta létező technika, azonban böngészőtámogatottsága még részleges volt.

[15] Natív WebGL környezetben egyetlen háromszög kirajzolása 100 sornál hosszabb kódsort jelenthet. Bonyolult geometriai alakzatok kialakítása több ezer sor lenne, amely mérnöki, matematikai, geometriai és fejlesztői szaktudást igényel. Példakód a dolgozat függelékében látható. *[Script.WebGLCube]*

Ezek a segédkönyvtárak lényegében a WebGPU API értelmezését könnyítik tehát azzal, hogy leegyszerűsítik annak funkcióhasználatát, függvényhívásait. A térbeli animáció és grafika ezen segédkönyvtárak által válik elérhetővé a 3D tervezőművészek számára is.

1.4. A technológia aktualitása

Doktori tanulmányaim kezdetén, első kutatási beszámolóim megírásánál a WebGPU technológia még nem létezett. Csupán tudományos kutatói munkásságom harmadik évére indult a reformtechnológia nemzetközi fejlesztésének bevezetőidőszaka. Alacsony böngészőtámogatottsága miatt az elmúlt három év során csak részlegesen volt elérhető, sokáig például csak a Safari[16] böngészőben. Csupán 2023. január végétől vált publikussá a fejlesztők számára is[17], akkor még kizárólag regisztrált, tesztelői jelleggel.

A WebGPU kezdeti, kísérleti szakaszaiban egyedileg megigényelt token[18] segítségével volt lehetőségem a tesztelésre, melyet a forráskód meta elemei között kellett elhelyeznem *(a http-equiv="origin-trial" jelölésű attribútummal)*. A 3D megjelenítés a jóváhagyott és érvényes egyedi azonosítóval, továbbá a releváns böngésző tulajdonság *(Chrome flag)* engedélyezésével[19] volt kezdeményezhető. A kutatási, tesztelői fázisban lévő technológia elérése a dolgozat megírásának utolsó pillanatáig nem volt mindenki számára biztosított. Elmondhatom, hogy doktori tanulmányaim időszaka végigkísérhette ennek a rendkívül új 3D technológiának a születését, a létrejöttétől egészen a nemzetközi publikálásig.

Egészen egyedülálló, hogy doktori disszertációm leadása előtt négy nappal, 2023. 04. 06-án vált publikussá[20], és mindenki számára elérhető natív eszközzé a Chrome böngészőben.

Rendkívüli egybeesés az is, hogy a Google az első nyilvános WebGPU oktatóanyagában John Conway 1970-es művének, az Életjátéknak az újraprogramozására

---

[16] A Mozilla Firefox (Beta) és a Micorsoft Edge (Canary build) csupán részleges támogatást biztosít a dolgozat megírásának idején. A WebGPU böngészőtámogatottságát ezen a hivatkozáson lehet ellenőrizni: https://caniuse.com/webgpu.

[17] Corentin Wallez, a WebGPU fejlesztőjének Github profilján ellenőrizhető: https://github.com/Kangz

[18] Korábban a WebGPU token igénylése a Chrome fejlesztőknek szánt oldalán volt lehetséges: https://developer.chrome.com/origintrials/#/trials/active

[19] A Chrome flag elérése: about://flags, a korábban engedélyezni kívánt flag pedig a #enable-unsafe-webgpu volt, Android alapú eszközökön a --enable-features=Vulkan flag engedélyezése szükséges. Firefox esetén about:config url-en a dom.webgpu.enabled opció aktiválása szükséges.

[20] Beaufort François - Wallez Corentin: Chrome ships WebGPU. Google Developer, 2023. tavasz (https://developer.chrome.com/blog/webgpu-release/) Utolsó letöltés: 2023.04.06.

ösztönzi a kísérletező fejlesztőket.[21] Különös aktualitást ad, hogy a WebGPU oktatóanyagot a publikus védésre szánt dolgozat leadási határideje előtt két nappal, 2023. 08. 29. tették mindenki számára elérhetővé. *[Fig.WebGPU app 1.]*

1.4.1. A kutatás jelentősége

Az azonosított és jóváhagyott tesztelői oldalakon létrejövő első kísérleti térbeli vizualizációk és kezdeti 3D böngészőmodellek rendkívül fontos visszajelzést biztosítottak a WebGPU API és a WebGPU shading nyelv fejlesztőinek. A doktori kutatás során létrejött DLA prezentációs WebGPU adatbázis 3D modelljei így segíthettek az első elemzések, és felhasználói tesztek megalkotásához, így hozzájárulhattak egy éppen születő interaktív web 3D technológia kialakításához és optimalizálásához. *[Fig.Token1, Fig.Token2] [Script.2]*

1.5. Az értekezés szerkezete

Az értekezés bevezető része a témaválasztásról, a célok kitűzéséről, és a disszertáció struktúrájának ismertetéséről szól. Bemutatom az OpenGL, a WebGL és a WebGPU fogalmát, melyek átívelik a dolgozat alapvető értelmezési rétegeit, és melyekkel a vizsgált témakör központi tematikájának alapvető definíciós adatbázisa kirajzolható. Az elsődleges fogalomkörök tisztázása mellett ismertetem doktori kutatásom szerepét, és azt, hogy milyen jelentőséggel bír a DLA munkához kapcsolódó tervezési és megvalósítási folyamatokban.

A második fejezet a broadcast vizualizáció változó szerepéről, a platformfüggetlenség igényének megjelenéséről, valamint az adatalapú, és generatív modellek integrálásának problematikájáról szól. A dolgozat ezen szakazának célja, hogy feltárja, milyen kihívásokkal kell szembenéznie a broadcast alkotóknak a digitális képalkotás megújuló területein, és hogyan változnak a megszokott munkafolyamatok. Értékelésem során arra törekszem, hogy kiindulópontot adjak kutatási kérdéseim megválaszolásához.

A magyar 3D technológia-történetének rövid áttekintése során azokat a rendkívüli alkotásokat és kísérleti időszakokat elemezem, mely disszertációm 3D vizualizációs eszközeinek szempontjából fontos és releváns. Bemutatom az első vertikálisan építkező sztereoszkóp képsorokat, melyeket az ötvenes években megnyíló első 3D filmszínház tűzött

---

[21] Brandon Jones, François Beaufort: Your first WebGPU app. *Google Developers*, 2023. nyár (https://codelabs.developers.google.com/your-first-webgpu-app#0) Utolsó letöltés: 2023.08.30.

moziműsorára. A fejezet további kiemelt témaköre az első hazai adatalapú filmművészeti alkotás, melynek dinamikus történetét maga a filmvetítés alkalma formálja, cselekménycszálai teljesen automatikusan íródtak, és amely interaktív szereplői forgatókönyv nélkül is rátaláltak filmbeli szerepükre. Ebben a részben igyekszem összegzést adni az adatalapú komputeranimációk nemzetközi eredményeiről is. A fejezet a holografikus térbeli kép megalkotásának történetével is foglalkozik, valamint kitér az első magyar CGI alkotás televízióban megjelenő képsoraira, melyek már a legmodernebb vizualizációs eljárások felé nyitották meg az utat.

A Web 3D bemutatásával a negyedik fejezetben foglalkozom. Ebben a dokumentumegységben a technológiák részletesebb ismertetése során kiemelem a WebGL és a WebGPU közös tulajdonságait, elemzem a térbeli vizualizáció szempontjából legfontosabb különbségeket, és megvizsgálom mely webes 3D könyvtár architektúrájaként integrálhatók broadcast vizualizációs megoldásként.

A web 3D fejlesztői környezet ismertetése során a DLA munkásság gyakorlati megvalósításhoz szükséges 3D Full-Stack környezet technológiai összefüggéseit, a Front-End és Back-End módszertanokat részletezem. Azokat a kliens- és szerveroldali architektúrákat veszem sorra, melyek segítségével a 3D animációs jelenetek élő adás környezetébe transzplantálhatók. A dokumentáció során olyan osztályok és eljárásgyűjtemények rögzítésére törekszem, melyek mestermunkám megvalósításához elengedhetetlenül hozzájárultak, és követik a modern Full-Stack fejlesztés rutinjait.

Az értekezés következő lépésében feltárom a webes 3D jelenetintegráció publikációs módszertanát. A fejlesztési módszertanok mellett azokkal 3D tervezői eszközökkel foglalkozom, melyek a modern, interaktív képalkotás forrásaként értelmezhetők. Nyitásként olyan 3D alkotói szempontrendszereket vizsgálok, melyek az interaktív adaptáció megvalósításához szükségesek. Azokat a formátumokat elemezem, melyek a platformfüggetlen 3D jelenet-transzplantáció kiindulópontjaként definiálhatók. Értelmezem a típusok közötti karakterisztikákat, és feltárom hogyan optimalizálhatók a 3D vizualizáció szempontjából fontos exportálási paraméterek. Ezután értékelem azokat a JavaScript alapú 3D könyvtárakat, melyek optimalizált fejlesztési réteget biztosítanak az alkotó kreatív szakemberek számára. A fejezetben részletesen kitérek a 3D állományok ábrázolási folyamatának problematikájára, a jelenettervezés lépéseire és körülményeire, továbbá a

könyvtárakon keresztül elérhető mozgóképes funkcionalitás, és grafikus ábrázolóeszközök alkalmazói metódusaira.

A következő kutatási fejezetben a web 3D lehetséges publikálási stratégiáit vizsgálom. Azokat a gyakorlati integrációs módszereket elemzem, melyek során az elkészített 3D jelenet dinamikus, paraméterezhető formátumban valósidejű adáskörnyezetbe illeszthető. Vizsgálom a televíziós ábrázolástechnika jelenlegi helyzetét és trendjeit, valamint bemutatom a web 3D által nyújtott további adaptációs lehetőségeket.

A Web 3D jövőképének részében olyan modern platformokat elemezek röviden, melyek a disszertáció során feltárt broadcast transzplantációs munkamódszer további integrációs megoldásaként értelmezhetők. Bemutatom azokat az előremutató törekvéseket, melyek automatizált, generatív módszerekkel támogatják a jövő mozgóképes szakemberinek fejlesztői és tartalomgyártási eszközkészleteit.

1.6. Tézisek

A WebGPU technológia hozzájárul a broadcast televízió és online streaming 3D adatvizualizációs eszköztárának bővítéséhez, lehetővé téve a platformfüggetlen interaktív tartalmak előállítását és adaptációját.

A 3D webgrafikai könyvtárak és a WebGPU technológia új transzplantációs módszereket kínál broadcast rendszerek számára, lehetővé téve a hagyományos 3D tervezőszoftverekből származó jelenetek hatékony adaptációját.

A WebGL és WebGPU technológiák integrációja a broadcast adatvizualizációs sablonrendszerekkel és a webes interaktív alkalmazás-elemekkel új, rugalmasabb és változatosabb vizualizációs paradigmát teremt a televíziós- és online streaming csatornák számára.

Az mesterséges intelligencia és gépi tanulás technológiák alkalmazása a WebGL és WebGPU rendszerekben új kreatív 3D vizualizációs lehetőségeket teremt a broadcast és online alkotók számra.

1.7. Theses

The WebGPU technology contributes to the expansion of the 3D data visualization toolkit for broadcast television and online streaming, enabling the creation and adaptation of platform-independent interactive content.

3D web graphics libraries and WebGPU technology offer new transplantation methods for broadcast systems, enabling efficient adaptation of scenes from traditional 3D design software.

The integration of WebGL and WebGPU technologies with broadcast data visualization template systems and web-based interactive application elements creates a new, more flexible, and diverse visualization paradigm for the television and online streaming channels.

The application of artificial intelligence and machine learning technologies in WebGL and WebGPU systems creates new creative 3D visualization opportunities for broadcast and online creators.

2. Problémafelvetés - technológiai paradigmaváltás a 3D tervezésben és a konvergens mozgóképalkotás problematikája

A háromdimenziós képalkotó eljárások módszertana és felhasználási lehetőségrendszere olyan technológiai paradigmaváltáson megy keresztül, mely jelentősen újra-definiálja, kiszélesíti a hagyományos 3D szerzői eszközkészletet és fogalomrendszert. Ahogyan a film is hibrid műfajjá változott[22], úgy a térbeli alkotás gyakorlata is jelentősen átalakult; több *–mint három-* dimenziós, generatív, multiplatformos[23], adat- és adatbázis alapú[24], szkript-vezérelt, szenzoros[25], immerzív adaptációs folyamattá nőtt[26], melyben egyre inkább jelentésttartalmat szerez a szoftveres-, böngészőművészet [27], a gépi tanulás és a mesterséges intelligencia alapú mozgóképgyártás fogalma is.

A 3D tervezők által elkészített művészeti alkotások a broadcast engine-ek, a WebGL és WebGPU grafikus motorok, a sztereoképek, virtuális-, kiterjesztett- és kevert valóság generált illúzióterei irányába egyformán adaptálhatók lettek. A webalapú környezetbe illesztett 3D tartalom, és az adatbázis-alapú mozgóképes interakciók mellett, a valósidejű térbeli grafikai elemeket ma már hibriditással jellemezhetük, és bármilyen kompatibilis felület irányába integrálhatjuk.

A fenti folyamatokon túl a gépi tanulás, az automatizált feldolgozás és az okos rendszerek térnyerése rendkívül jelentős hatást gyakorol a mozgóképes és kreatív szakmákra, különösen azokra, amelyekben a mesterséges intelligencia alkalmazható. Az eltérő

---

[22] Dragon Zoltán: A film a digitalizáció korában – bevezető. *Apertúra*, 2011. tavasz (https://www.apertura.hu/2011/tavasz/dragon-film-digitalizacio-koraban/) Utolsó letöltés: 2022.06.03.

[23] Gerencsér Péter: Bevezetés a web 2.0 definícióiba és ideológiáiba. Aperatúra, 2019. tél (https://www.apertura.hu/2019/tel/gerencser-bevezetes-a-web-2-0-definicioiba-es-ideologiaiba/) Utolsó letöltés 2023.01.06.

[24] Dragon Zoltán: Újmozi, avagy adatbázis az egész világ. Válasz Sághy Miklós A film jövője: adatbázis és/vagy (interaktív) narratíva? című reflexiójára. *Apertúra,* 2006. tél (http://uj.apertura.hu/2011/nyar/dragon-ujmozi-avagy-adatbazis-az-egesz-vilag/) Utolsó letöltés: 2019.05.10.

[25] Füzi Izabella: A képernyő és a filmvászon hibridizációja: a videóchat és a videószelfi játékfilmes idézése (Remélem, legközelebb sikerül meghalnod, FOMO – Megosztod és uralkodsz, Szép csendben). Aperatúra, 2021. nyár (https://www.apertura.hu/2021/nyar/fuzi-a-kepernyo-es-a-filmvaszon-hibridizacioja-a-videochat-es-a-videoszelfi-jatekfilmes-idezese/) Utolsó letöltés 2021.10.21.

[26] Kelemen Zsolt: Arcade Fire 2.0 és az adatbázis-logika. *Apertúra,* 2011. tavasz (http://uj.apertura.hu/2018/tavasz/fuzi-nezo-es-kozonseg-mozitorteneti-vazlat/) Utolsó letöltés: 2019.05.10.

[27] Gerencsér Péter: A net art az net art az net art. A médiumspecifikusság és a posztmédia dichotómiája az internetes művészetben. *Aperatúra*, 2016. nyár (https://www.apertura.hu/2019/tel/simons_az-iphone-es-a-youtube-kozott-a-filmek-mozgasban/) Utolsó letöltés 2022.05.10.

szakterületek ismeretanyagának egyidejű vizsgálata és alkalmazása elengedhetetlen a mozgóképművészeti és fejlesztői diszciplínák összeolvadása miatt. Az alkotóknak a dinamikus, platformfüggetlen tartalomelőállítás problematikájával, és a mesterséges intelligencia eszközkészletének integrálásával kell szembenéznie.

2.1. A platformfüggetlenség igényének megjelenése broadcast környezetben

A konvergens televíziózás a 3D mozgóképgyártás folyamatának transzdiszciplináris megközelítését teszi szükségessé. Broadcast környezetben olyan integrációs hibrid rendszerek kialakítása szükséges, amelyek a térvizualizációs és a 3D videografikai eljárásokból származó mozgóképes objektumok transzformációs és adaptálási folyamatát eltérő médiafelületek irányába képesek biztosítani. A digitális képtartalom előállításának jelentős változása és a valósidejű vizualizáció igényének megkérdőjelezhetetlen térhódítása elkerülhetetlenül magában rejti a tervezői munkafolyamatok újradefiniálásának szükségességét is.

A platformfüggetlenség nemcsak a szoftverek felhasználói rétegére, hanem az alkalmazások architektúrájára is jelentős hatással van. A meglévő valósidejű 3D programok már elkezdték integrálni azokat a formátumokat, és modulokat, melyekkel a tervezés utáni webes publikációs eljárást is natívan támogatni tudják. Az alkalmazásfüggetlen formátumok azonban csak részlegesen képesek megörökíteni az eredeti művészeti koncepciót. Különösen igaz ez a hagyományos 3D szoftverekre. Számos korlátozó jellemző határolja a kreatív alkotói folyamat szabadságát. A magas minőségű modellek kialakítása szigorú kötöttségekkel valósítható csak meg. A nagy felületszámú alkotásokon éppúgy elkerülhetetlen a topológiai újra-strukturálás, vagy az anyagparaméterek utólagos módosítása, mint a jelenet fény- és árnyéktulajdonságainak teljes újratervezése. Az ismételt munkafolyamatok különösen összetettek, és a megszokott grafikus interfész *(GUI)* helyett sok esetben csak programozási nyelvek, szkriptek segítségével végezhetők el. Ez a munkamódszer pedig további követelmény elé állítja a szakembereket, hiszen az új technológiák alkalmazása és az ezekhez kapcsolódó tudásanyag bővítése elengedhetetlen. A platformfüggetlen és mozgóképes kompetencia-anyagok rendkívül dinamikus fejlődése pedig tovább nehezíti a technológiai változások nyomon követését.

Az egyedi alkalmazásokat támogató szkript- és kódbázisok kialakítása, a 3D jelenetoptimalizációs rutinok fejlesztése, az egyedi vizualizációs alkalmazások platformfüggetlen publikációs eljárásait támogató broadcast eszközkészletek megtervezése, a

3D adatbázisok fejlesztése, és a mozgóképes jelenetek interaktív elemeinek definiálása, jelentős kihívások elé állítják az alkotókat és a televíziós csatornákat. Ezen kihívások mellett továbbá figyelembe kell venni a mesterséges intelligencia térhódítását is, melynek részeként az alkotóknak a generatív állományok integrálásával is számolniuk kell.

2.2. Az MI és a generatív modellek integrálásának problematikája

A mesterséges intelligencia egyedi generatív működési modelljének elterjedése rendkívüli változást okoz a képalkotó rutinokban. A gépi tanulási- és adatbányászati technológiákra épülő algoritmusok képesek az előre bevitt adatokból új tartalmat előállítani, és azokat egyedi döntések mentén feldolgozni. A generatív modellek önállóan, grafikus tervezői beavatkozás nélkül is alkalmasak kreatív, korábban nem létező műalkotások megteremtésére. A módszer lehetővé teszi az azonnali interaktív vizualizációk előállítását, a néző számára pedig biztosítja a teljesen egyedi tartalomfogyasztás élményét. Bár az MI és a generatív modellek használata számos előnyt biztosít broadcast felhasználási területen is, elkerülhetetlenül magával vonzza egyes szakterületek teljes átalakulását, munkafolyamatok újjászervezését, egyes munkakörök megszűnését.

Az elmúlt évtizedekben a mozgókép-technológia jelentős átalakuláson ment keresztül, és ezek a folyamatok mindig kreatív lehetőségeket hoztak magukkal. A generatív modellek alkalmazása és az MI térhódítása is kétségtelenül újabb fontos mérföldkő lesz a képalkotás történetében.

Amikor távolabbról vizsgáljuk a filmtechnika-történet alakulását, felfedezhetjük benne azokat a kiemelkedő pillanatokat, melyek a jelenkorihoz hasonló rendkívüli változások ígéretét, vagy garanciáját hordozhatták magukban. Az adatvezérelt filmalkotások, a holografikus kép, a sztereoszkóp felvétel- és vetítéstechnológia problémái éppúgy részei voltak a korábbi filmtechnikai kísérletezéseknek, mint amelyekkel napjainkban szembesülünk.

Értekezésem következő fejezetében azokat a kreatív alkotókat és módszereket ismertetem, akik innovatív kísérleteikkel kiemelkedő technológiai eredményeket értek el a térbeli mozgóképgyártás hazai történetében. Az ő munkásságuk segíthet megérteni a modern technológiák értékét, továbbá a kísérletező szándékú hozzáállás jelentőségét napjainkban is.

Disszertációmban azokat az egyedi immerzív eljárásokat helyezen fókuszba, melyek a modernkorban is ismert és használt 3D technológiák korai előfutárai.

3. Magyar 3D technológiatörténeti visszatekintés

A disszertáció aktuális fejezete olyan technikai újítások történetét fogja át, amelyek, ha csak rövid időre is, de forradalmasították a képalkotást Magyarországon, és lehetővé tették a korábban elképzelhetetlen tartalmak térbeli megjelenítését a mozivásznon vagy a képernyőkön. A filmtechnika-történelem és a magyar mozgóképgyártás azon unikális pillanatait helyezem fókuszba, melyek során hazai alkotóink képesek voltak felülírni a nemzetközi gyakorlatot és újradefiniálni a világ jelentős mozgóképművészeinek hagyományos eszközkészletét. Az egyik ilyen áttörést a térhatású plasztikus filmek megjelenése jelentette.

A plasztikus filmek történetének áttekintését követően a dolgozat a magyar automatizált filmművészet egyik rendkívüli alkotását, a holografikus képek megteremtésének körülményeit, és az első képernyőn megjelenő CGI jelenetek leírását foglalja össze.

3.1. Plasztikus film

A sztereo-képek térhatású élményt hoztak az ötvenes évek filmszínházainak vásznaira. A nézők átélhették a sztereoszkóp látványvilág immerzív mozgóképes élményét, mely újdonságként hatott a korábbi anagliptikus technológia látványvilágához képest. A magyar plasztikus térbeli alkotóművészek a nemzetközi gyártási technológiák tudományos eredményeit alapul véve dolgozhatták ki a hazai filmgyártás történetében egyedülálló sztereo-rögzítési és filmvetítési mechanizmusukat.

3.1.1. Nemzetközi kontextus: a sztereofilm fejlődése és hatása a magyar plasztikus filmre

A sztereoszkóp képalkotás kialakulásához vezető úttörő munka már a XIX. század második felében elkezdődött. Wheatstone és Brewster az optikai illúziók és a térbeliség vizsgálatával járultak hozzá a sztereoszkópia alapjainak megteremtéséhez.

A sztereo képvisszaadás mechanizmusát Charles Wheatstone nevéhez köthető, aki 1831-ben sztereoszkóp nevű eszközével megteremtette az első térhatású illúziót. Ennek

lényege az volt, hogy két különböző nézőpontból készült sík képet helyezett egymás mellé, amelyek 45 fokos szöget zártak a néző szemeihez képest, így létrehozva a térhatás illúzióját.[28]

1849-ben David Brewster és Jules Duboscq találta fel a lencsés sztereoszkópot.[29] A népszerű technológia a duális térfényképek megtekintését egyszerűsítette egy hordozható eszköz segítségével. A sztereó-nézőn, sztereó vetítőn keresztül megelevenedő másolatok megnyitották az utat a sztereo-kamerák elterjedése előtt. 1850-ben készült el az első sztereodagerrotípia Magyarországon, mely Strelisky Lipót nevéhez köthető.

Thomas Edison 1855 találta fel az animációs sztereo kamerát, a kinematoszkópot, mely a kamerával rögzített képeket azonos sebességgel tudta visszajátszani. A népszerű kézi sztereoszkóp kifejlesztése Oliver Wendell nevéhez köthető.

Bodrossy Félix plasztikus filmrendező, A plasztikus film c. tanulmányának történeti bevezetőjében további fontos mérföldköveket is megemlít. Kiemelte a *„biedermeier szalonok”* egyszerű illúziójátékainak jelentőségét, a stroboszkópok szerepét, Eadweard J. Muybridge fénykép-technikai újítását.[30]

A XX. század folyamán a technológia fejlődésével párhuzamosan vált lehetővé a sztereoszkópia alkalmazása a filmgyártásban is. Ezek az alkotások az addig ismert technológiákat próbálták forradalmasítani és a mozgóképes térhatású technika elterjesztésére törekedtek.

Az 1893-ban William Friese-Greene által létrehozott komikus jeleneteket bemutató szekvenciák[31] és az Auguste Lumiere által készített sztereoszkópos film, A vonat érkezése *(L'arrivee du train, 1903.)* után számos hasonló alkotás látott napvilágot világszerte az 1950-es évekig.

---

[28] Dr. Fekete Róbert Tamás, Dr. Tamás Péter, Dr. Antal Ákos, Décsei-Paróczi Annamária. „3D megjelenítési technikák” *BME-MOGI*, 2014. tavasz (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2021.11.10.

[29] Gorácz Anikó: 3D revolúció. A 3D múltja, jelene és jövője. Térélmény. *Filmvilág*, 2009 ősz (https://www.filmvilag.hu/xista_frame.php?cikk_id=9878) Utolsó letöltés: 2024.08.23.

[30] Bodrossy Félix, „Volt egyszer egy plasztikus film”, in *Az Élet és Tudomány Kalendáriuma* 1981, szerk. Németh Ferenc, Ludas M. László, 264. (Budapest: Hírlapkiadó Vállalat, 1980.), 3.

[31] Schedeen, Jesse. „The history of 3D movie tech.” *IGN*, 2010. tavasz (http://www.ign.com/articles/2010/04/23/the-history-of-3d-movie-tech) Utolsó letöltés: 2017. 01.14.

[Toldi Plasztikus Filmszínház - vetítés közben.
Forrás: Bodrossy Félix: A plasztikus film. Budapest, Művelt Nép Könyvkiadó, 1953.]

Bár a legelső anagliptikus *(térhatású)* filmek pillanatképeit a nézők valósághűnek találták, a filmesek azon dolgoztak, hogy még erőteljesebb hatást érjenek el. Lumiere „fényszűrőt, zselatinnal bevont vagy naftolzöld, eozin és tartracin keverékkel festett üveglapot”[32] alkalmazott a hatás fokozása érdekében. A technológiai eljárások száma is folyamatosan nőtt.

Az Metro Goldwyn Mayer egy Audioskope nevű plasztikus képalkotási rendszert alkalmazott, mely később Magyarországon is megjelent. A hasonló eljárások kissé eltérő elnevezéssel *(Pasztigram, Metroszkopix és Parallax Sztereogram)* jelölték a hasonlatos technológiákat.[33] A plaszticitás elméleti lehetőségeit az Yves-testvérek és Estanave vizsgálták[34], míg a Televiev System már képes volt felváltva vetíteni a képszekvenciákat[35]. Az anaglif technikával készült Szerelem ereje *(The power of love)* 1922-ben nagy sikert aratott, és megalapozta a technológia továbbfejlődését. Később az anaglif technikát a színes

[32] Kispéter Miklós. A győzelmes film – Film, tudomány, művészet. Budapest: Királyi Magyar Egyetemi Nyomda, 1938.
[33] Nagy Eszter, Politzer Péter, „Variációk a harmadik dimenzióra”, *Filmvilág* 40, 1. sz., (1997): 48–50. (http://filmvilag.hu/xereses_frame.php?cikk_id=1379) Utolsó letöltés: 2023.01.14.
[34] Kispéter Miklós. A győzelmes film – Film, tudomány, művészet. Budapest: Királyi Magyar Egyetemi Nyomda, 1938.
[35] Dr. Fekete Róbert Tamás, Dr. Tamás Péter, Dr. Antal Ákos, Décsei-Paróczi Annamária. „3D megjelenítési technikák” *BME-MOGI*, 2014. tavasz (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2021.11.10.

képvisszaadásra is alkalmassá tették. 1935-ben készült el az első színes, háromdimenziós mozgókép, és Edwin H. Land a polarizációs eljárás kidolgozásával további fontos eredményeket ért el a képminőség javításában.

A Szovjetunióban 1941-ben a mutatják be a Paralax sztereogram technológiát. A nézőknek mozdulatlanul kellett ülniük, a nézőtér meghatározott pozícióiban, hogy részesei lehessenek a térbeli filmélménynek. 1947-ben készült el az első orosz 3D-s mozifilm, a Robinson Crusoe is, mely hasonló technológiával készült, így a nézőnek nem kellett szemüveget viselnie.

A II. világháború utáni sajtó a megújuló 3D reformokról írt: „A Szovjetunió kinematográfiái minisztériumának rendeletére Moszkvában, a Szverdlov-téren »Vosztok-Kinó« néven plasztikus filmszínház nyitotta meg kapuit [1947] december elsején. A nagyobb méretű vetítővásznakkal felszerelt plasztikus filmszínházak létesítésével egyidejűleg megindult a kisméretű fényerős sztereo-vetítővásznak készítése is. Ezeket a kultúrházakban és a munkásklubbokban szerelik majd fel." [36] Egy későbbi lapszámban egészen sajátosan összegezték a 3D térhatású filmek igényét. „Más államokban, úgy látszik, nem éri meg a nagy költséget az, hogy a moziátogatóknak *(sic!)* valóban művészi élményt adjanak. Hosszú idő kell még ahhoz, hogy a tér-szerű, színes hangosfilm elterjedjen és a mozitulajdonosok ellenállását kell előbb leküzdeni. De a néhány – vagyonát féltő – mozis nem állhat a fejlődés útjába s remélhetőleg egy-két év vagy évtized múlva mindennapos lesz a háromdimenziós film használata." [37]

Az 1950-es években a háromdimenziós filmkészítés új lendületet kapott, és az évtized végére már széles körben elterjedt világszerte. Az 1952-ben bemutatott Cinerama utána a Bwana Devil című alkotás is újra felhívta a figyelmet a térhatású filmszínházakra az egyedi polarizációs szűrőtechnikának köszönhetően *(Natural Vision)*[38].

Egy évvel később mutatták be a Viaszbabák háza *(House of Wax)* című filmet, mely a korszak kiemelkedő minőségű és költésvetésű alkotása volt, ráadásul magyar vonatkozású is,

[36] Sz. Oszipov Tjurin, „Plasztikus Filmszínház született a Szovjetunióban", *Élet és tudomány* 2, 2. sz. (1947): 61.
[37] Z. Kiss Endre. „A mozi-vászon mélységei". *Élet és tudomány* 2, 26. sz. (1947): 61.
[38] Dr. Fekete Róbert Tamás, Dr. Tamás Péter, Dr. Antal Ákos, Décsei-Paróczi Annamária. „3D megjelenítési technikák" *BME-MOGI*, 2014. tavasz (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2021.11.10.

hiszen Tóth Endre rendezte[39]. Ugyanebben az évben vetítették először George Pal magyar származású amerikai filmproducer és rendező *Világok háborúja (War of the Worlds, 1953)* című alkotását is.

A XX. század második felében tapasztalható nemzetközi sztereofilmgyártási trendek inspirációt jelentettek a magyar művészeti és tudományos közösség számára. Ez az ösztönző hatás nem csupán a saját térhatású filmek létrehozására sarkallta a hazai alkotókat, hanem innovatív technológiai alkalmazások és újszerű módszertanok implementálására is. Az első magyar sztereofilm művészek jelentős kutatási tevékenységet folytattak a térhatású mozgóképalkotás sajátos eszközkészletének és technológiai protokolljainak fejlesztéséért, ami nem csupán a kreatív kibontakozást segítette elő, hanem a filmgyártás hazai szintű innovációját is.

3.1.2. A plasztikus film és a magyar sztereoszkópia keletkezésnek történeti áttekintése

A nézők Bodrossy Félix munkásságának köszönhetően, 1952. június 25-én láthattak először sztereo-plasztikus képsorokat Magyarországon.[40] A budapesti Toldi mozi áprilisi bezárását követően jelentős átalakításokat végeztek az épületben annak érdekében, hogy a magyar filmgyártás első térhatású alkotásainak vetítését megvalósíthassák. A nézőtér és a gépház teljes újjáépítése után, megkezdődhetett a próbaüzem, majd nyárra kinyitotta kapuit a Toldi Plasztikus Filmszínház. A mozi három éven át vetítette a magyar gyártású térhatású filmeket.

Az első hazai sztereoszkópikus mű az Állatkerti séta című film volt, melyet 1951-ben mutattak be. A film készítői, Bodrossy Félix és Győrffy József addigra már évek óta dolgoztak azon, hogy meghonosítsák a sztereoszkópikus filmkészítést Magyarországon. A Fekete-fehér Artista vizsga *(1951)* és a Sztereó híradó *(riportfilmek, 1952. május 1.)* című filmek állandóan szerepeltek a moziműsoron. Bodrossy visszaemlékezése jól mutatja, hogy alkotásaik nagy sikert arattak: "Hat hónapon át a mozipénztár előtt hosszú sorok

[39] Michael Barson, „André De Toth", in Encyclopædia Britannica. (https://www.britannica.com/biography/Andre-De-Toth) Utolsó letöltés: 2020. 10.28.
[40] Kurutz Márton. „Budapest V., Bajcsy-Zsilinszky út 36-38. City filmszínház, Szittya-filmszínház, Úttörő mozi, Toldi Plasztikus Filmszínház, Toldi mozi". *Hangosfilm,* 2011. (http://www.hangosfilm.hu/mozilexikon/budapest-v-city-szittya-uttoro-toldi) Utolsó letöltés: 2018.01.14.

kígyóztak."[41]. A Téli regének *(zenés jégbalett, 1953)*, a Színes szőttestnek *(1954)* és a második Plasztikus filmhíradónak is nagyon sokan látták később.

[Bajcsy-Zsilinszky út, Toldi mozi. A felvétel Bodrossy Félix: Állatkerti séta és Artista vizsga című 1952-ben forgatott plasztikus (3D) filmjeinek felújításakor készült. Forrás: Fortepan]
[Újsághirdetés 1952 december. Forrás: Élet és tudomány folyóirat – 1947. / VII. évf. 50. p. 762.]

Bodrossy Félix volt az első, aki összefoglalta a magyar sztereoszkópikus, 3D-s filmkészítés elméleti módszertanát és technikai dokumentációit. Tanulmányában összegezte a sztereopszis *(vagyis a térlátás)* kiemelten fontos szerepét a filmtechnikában. A biológiai térlátás vizsgálata és az emberi szem, valamint a filmtechnikai eszközök hasonlóságainak és korlátainak összevetése egyedi és jelentős mozgóképtörténeti leírást eredményezett. „A nyugtalan, folyton újat kereső alkotó ars poétikája: a fantasztikus, a vízió. Ezért keresi fel szívesen alkotásaiban a mese, a mitológia, a sci-fi világát. S hogy ezt a világot mind eredetiben, ötletesebben fejezze ki — folyamatosan megmegújítja művészi eszközeit. Ő készítette például az első színes cinemascope-filmet is Magyarországon. Már az ötvenesnek elején az ő nevéhez fűződött egy másik magyar kísérlet: saját eljárással megalkotta az első magyar három- dimenziós plasztikus hatású sztereó-filmeket. Ezután szinte természetes, hogy elsőként alkotott sci-fi rövidfilmet is a magyarok közül." [42]

[41] Bodrossy Félix, „Volt egyszer egy plasztikus film”, in *Az Élet és Tudomány Kalendáriuma* 1981, szerk. Németh Ferenc, Ludas M. László, 264. (Budapest: Hírlapkiadó Vállalat, 1980.), 270–272.
[42] László Ibolya. „Magyar világszabadalom – Új lehetőség a színes rajzfilmgyártásban”. *Tolna Megyei Népújság*, 1972. február 15., 4. (https://library.hungaricana.hu/hu/view/TolnaMegyeiNepujsag_1972_02/?query=SZO%3D(h%C3%A1rf%C3%A1s)&pg=125&layout=s) Utolsó letöltés: 2018.01.14.

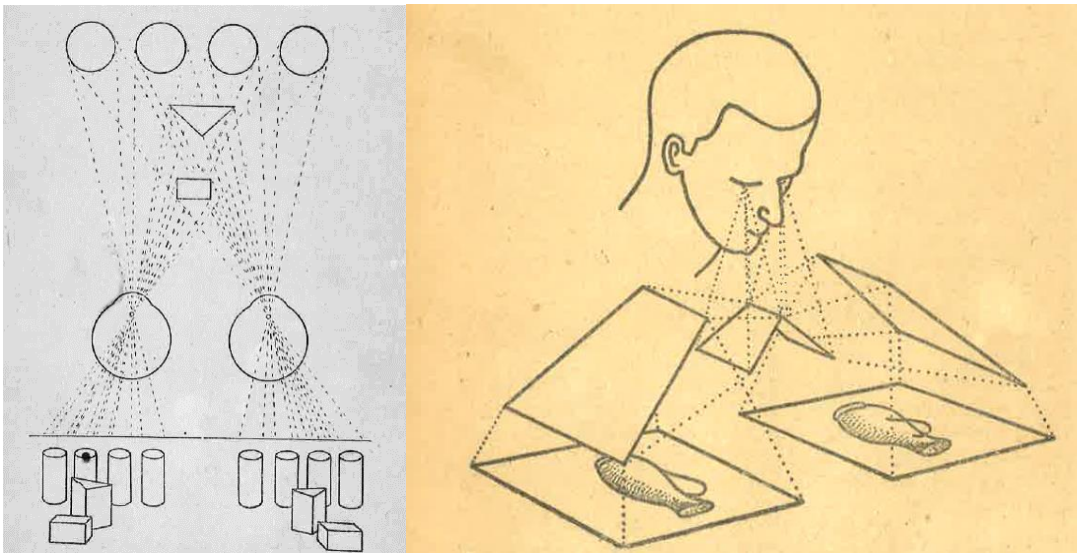
[Toldi Plasztikus Filmszínház - vetítés közben.
Forrás: Bodrossy Félix: A plasztikus film. Budapest, Művelt Nép Könyvkiadó, 1953.]

Bodrossy a magyar sztereofilmezés úttörője volt. Korán észrevette, hogy a korszak térhatású filmjeire jelentős nézi igény mutatkozik. „Napjainkban az érdeklődés világszerte a plasztikus *(sztereoszkópikus, térhatású, háromdimenziós)* filmre összpontosul. A filmszakma hatalmas átalakulás előtt áll, ugyanúgy, mint a hangosfilm feltalálásakor. […] Most, hogy jelen sorok szerzőjének és munkatársának, Barabás János főmérnöknek eljárása alapján hazánkban is megindult a plasztikus film gyártása, a moziba járók érdeklődése erősen megnövekedett a kérdés iránt.”[43]

Bodrossy elemzése a sztereopszis, azaz a vizuális érzékelés folyamatának és módszertanának tanulmányozásával indult. Korábbi írásom a Bodrossy féle térérékeléssel kapcsolatos biológiai alapfogalmakat összegezi. A térérzékelés problémájának origójaként a binokuláris diszparitás[44] szerepelt. „Ennek lényege, hogy a fejen elhelyezkedő szemek különböző helyzetükből adódóan eltérő vetületű látványt érzékelnek. A két szem retináján keletkező eltérő képek agyunk segítségével válhatnak összesített eredményképpé. Bodrossy e binokuláris fúzió elemzésekor azt a biológiai folyamatot írta le részletesen, mely során az agy a két szem által látott képet egybeolvasztja. Kiemelte, hogy a két szem eltérő pozíciója miatt létrejövő parallaxis teszi lehetővé a pontos mélységészlelést. A binokuláris érzékelés lényege

[43] Bodrossy Félix: *A plasztikus film.* Budapest, Művelt Nép Könyvkiadó, 1953. 5.
[44] Dr. Fekete Róbert Tamás - Dr. Tamás Péter - Dr. Antal Ákos - Décsei-Paróczi Annamária: 3D megjelenítési technikák. *BME-MOGI*, 2014. tavasz (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2023.11.10.

tehát, hogy szemünk két eltérő képének térbeli egyesítését agyunk végzi el, s így sztereoszkóp képet kapunk.” [45]

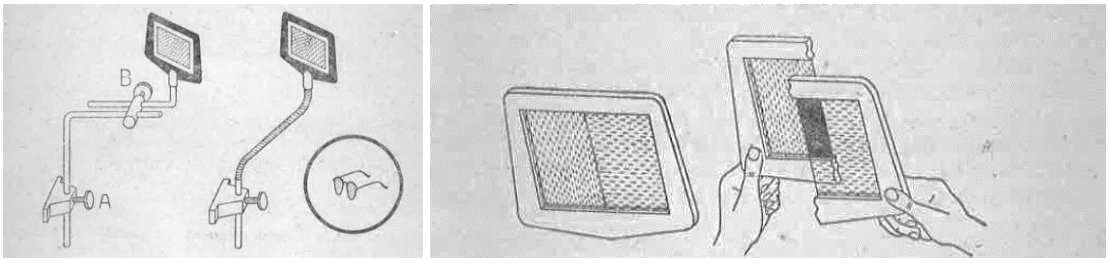


[Térbeli elhelyezkedés 'mögélátás' révén.
Forrás: Bodrossy Félix: A plasztikus film. Budapest, Művelt Nép Könyvkiadó, 1953.]
[Tükrös sztereoszkóp. Forrás: Bodrossy Félix: A plasztikus film. Budapest, Művelt Nép Könyvkiadó, 1953.]

A szerzők egyedi eljárással készítették a felvételeket. A kamerát sztereo előtéttel szerelték fel, melyek a látott kép Y-tengelyű eltolását és feldolgozását tettette lehetővé. A felvételeket 2:1 képarányú, vertikálisan egymás alatt elhelyezett rendszerben rögzítették, majd a Toldi mozi speciális vetítőgépének polarizált fényű prizmarendszerével egyesítették. A nézők statikus *(polárszűrős)* megtekintőablakon keresztül figyelhették a filmet, és tapasztalhatták meg az egyedülálló 3D hatást. A Toldi moziban plasztikus vetítőelőtéttel dolgoztak, amelyben két ék alakú prizma kapott helyet. A prizmák polarizáló szűrőn keresztül felfelé és lefelé vetítették a képet, és a két irányt síkba hozták. A nézők előtt egyedi nézőkészülékeket, nézőablakokat alakítottak ki. A térhatású filmeket ezeken a pontosan beállított nézőablakokon keresztül élvezete a néző. Az eszközök állandó pozícióban voltak rögzítve, így a két polárszűrő eltérő módon állt egymáshoz a nézők távolságától függően. A vetítések összességében nagy sikert arattak, azonban akadtak olyanok, akik a távolságfüggő nézőablakok átkalibrálása, vétlen átállítása miatt kevésbé voltak megelégedve. Bodrossy így jegyezte fel ezeket: „Persze megfigyelhetjük „határozott jellemeket”, akik nem hagyták magukat egykönnyen rábeszélni valamire, és csak azért is maguk felé fordították a szürke oldalt. Ezek aztán ún. „inverz” képet láttak, azaz mindegyik szemük a másik szemük számára vetített képet szemlélte és előadás után, szemüket dörzsölve, mozit és feltalálót szidva

[45] Balogh Áron (Színház- és Filmművészeti Egyetem, Budapest): A plasztikus film. *in: Theatron*, 2020/14,2.

távoztak. […] Nótórius későn jövők, akik a minden előadás előtt felhangzó figyelmeztetés után érkeznek, könnyen ugyanígy járhatnak."[46]



[Polár nézőkészülékek. Forrás: Bodrossy Félix: A plasztikus film. Budapest, Művelt Nép Könyvkiadó, 1953.]

A film dramaturgiai eszközkészlete nem változott jelentősen a plasztikus mozgóképek megjelenésével. A rendezők legfontosabb célja az volt, hogy a lehető legjobban átadják a térhatás élményét a nézőknek. Ezért olyan helyszíneket és eseményeket választottak, amelyekben a plasztikus mozgások és a jelenetek cselekménypályái jól érvényesültek. Az előre beállított térbeli kompozíciók segítségével a szereplők majdnem kiléptek a vászonról. Az ún. *"mögé látás"* elvét is szándékosan alkalmazták. A jeleneteket úgy rendezték, hogy a központi esemény előtérén és háttérén kívül másodlagos események is megjelenhessenek. A tárgyak, amelyek az előtérben látható tárgyak mögött helyezkedtek el, a parallaxisnak köszönhetően váltak érzékelhetővé. A kiemelés célja a távolság hangsúlyozása, azaz a mélységérzet megteremtése és a térbeli elrendezés kiemelése volt.

A magyar mozgóképgyártás utolsó térbeli filmjét Préda Tibor rendezte. A Magyar Híradó és Dokumentum Filmgyár 3D filmkockáit Bodrossy Félix sztereó eljárással fényképezte. Az alkotók mintaszerűen megőrizték a plasztikus filmekre jellemző karakterisztikát. A rendkívüli képsorok annak az állításnak a cáfolatai, melyek az sztereoszkóp képkészítés egyszerűségét feltételezték. A térbeli alkotás ugyanis a Népstadion 1953-as augusztus 20-i megnyitójára készült. Több ezer sportolót és nézőt kellett ezzel a technológiával rögzíteni. „Kicsit tudományos munka a felvétel, mert a legfontosabb eszközök a centiméter és a különböző táblázatok, mivel nem mindegy, ki milyen messze áll a kamerától, és mi van a háttérben." [47] A különlegesen nehéz felvételek, a tudatosan beállított és rögzített képsorok látványos filmtörténeti örökségeink. A Bodrossy féle plasztikus filmeket

[46] Bodrossy Félix: *A plasztikus film*. Budapest, Művelt Nép Könyvkiadó, 1953. 5.
[47] Valuska László: Bodrossy és a 3D - A polárszűrő, az Agy és 2 füstölgő szem. *Filmhu*, 2006. tavasz (http://magyar.film.hu/filmhu/magazin/bodrossy-es-a-3d-beszamolo-szakma) Utolsó letöltés: 2018.06.07.

a vetítések túlzott műszaki igénye és költsége miatt nem lehetett, vagy nem engedték tovább fejleszteni.

A háromdimenziós alkotásokkal kapcsolatos leggyakoribb ellenérvek is ezek voltak. A vetítő elé szerelt szűrő, az egyedi vászon, a polárszűrős nézőablak költséges technikát jelentettek. A térhatású mozi mindvégig a valóságos térérzetet igyekezett reprezentálni, de a nézők egy része így is többet várt az új technológiától. Bodrossy frappánsan válaszolt. „A plasztikus moziban ugyanúgy látjuk a világot, mint a valóságban. Azonban, aki „többet vár", mint a valóság, az persze csalódik."[48]

Az óriási sikerek ellenére, idővel egyre kevesebb nézőt vonzott a különleges technika. Bodrossy így emlékszik vissza: „…szerettünk volna rögtön hozzálátni a következő műsor készítéséhez, mert az átfutás sok hónap, előbb-utóbb új filmre lesz szükség. Mert nincs az a tengernyi néző, ami egyszer el ne fogyna… Ám, azzal nyugtattak bennünket, hogy minek nyüzsgünk; viszik a jegyeket, mint a cukrot! Valóban, szeptember 16-án már az 56 000. nézőnél tartottunk. A hatodik hónap után azonban a sorok kezdtek rövidülni… 1953 februárjában a helyzet kezdett katasztrofális lenni." [49]

A Plasztikus Filmszínház végül 1954 márciusában vettette utoljára a plasztikus filmeket. Bodrossy így összegezte a kísérleti korszak lezárását. „Csakhamar célzások keltek szárnyra a rentabilitásról, a ráfizetésről, s kezdtük sejteni, hogy művünk felett meghúzták a lélekharangot. 1954. március 5-én bezárták a Toldit, a világ második plasztikus filmszínházát." [50]

Az 1950-es években egyre népszerűbbé vált egy másik filmtechnikai megoldás gondolata is, a minden néző szemszögéből rögzített és vetített reál-plasztikus film koncepciója. A szinte kivitelezhetetlen bonyolultságú elképzelés nem jelentett valós alternatívát még a kísérletező alkotók számára sem.

[48] Bodrossy Félix: *A plasztikus film.* Budapest, Művelt Nép Könyvkiadó, 1953. 44.
[49] Bodrossy Félix: „Volt egyszer egy plasztikus film". In Németh Ferenc – Ludas M. László (szerk.): *Az Élet és Tudomány Kalendáriuma 1981,* Budapest, Hírlapkiadó Vállalat, 1980. 270–272.
[50] Uo.

Bodrossy Félix ehelyett a plasztikus film jövőjét holografikus ábrázolásban látta. „És lehet, hogy a plasztikus filmet ötven év múlva már nem is vászonra vetítik, hanem az alakok egyfajta, átlátszó műanyag-tömbben fognak mozogni.” [51]

3.1.3. A plasztikus filmkészítés öröksége

Mire a Sportoló fiatalok utómunkálata elkészült az egyetlen budapesti mozit is bezárták, mely képes lett volna vetíteni.[52] Bodrossy utolsó plasztikus filmjét több mint ötven évvel később a Színház- és Filmművészeti Főiskola egykori hallgatója, a Magyar Televízió későbbi operatőre, Molnár Miklós fedezte fel újra, és vitte az egyetemi közönség elé. Molnár Miklós olyan új trükk-eljárásokkal kísérletezett[53], melyek képesek lennének a magas költségvetésű technikát elérhetővé tenni a nézők számára. 1973-ban 3D-ben rögzítette Szent-Györgyi Albert Nobel-díjas tudós interjúját a Bodrossy féle sztereoszkóp képalkotás modernizált technikájával. Doktori kutatásom során lehetőségem volt tanulmányozni azon műszaki feljegyzéseit, melyek a plasztikus felvéltelkészítés jellegzetességeit elemzik. A híres biokémikust személyesen kérte fel a speciális felvételek elkészítésére.

A tudós örömmel vállalta el az újszerű riportot[54], melyet végül a Margitszigeti Nagyszálló halljában rögzítettek 6 percben[55]. A térhatású felvételeket két, egymástól 25 centiméterre elhelyezett 16 mm-es filmfelvevő géppel rögzítették, fekete-fehér filmre. Tizenöt évvel később készítette el következő 3D interjúját, ezúttal már színesben. Takács Marival 1988-ban rögzített sztereó-riportot, melyben a bemondónő üzen a jövő nézőinek. [56]

A Szent István Egyetem tanára tudta, hogy munkássága eredménye csak jóval később lesz felhasználható. 45 évvel ezelőtt úgy kellett megörökítenie a felvételt, hogy akkor még nem állt rendelkezésére a lejátszáshoz szükséges technika, és nem lehetett biztos abban sem, hogy ez az eljárás mikor válik elérhetővé, mivel a két vetítő asszinkronitása lehetetlenné tette

---

[51] Bodrossy Félix: *A plasztikus film*. Budapest, Művelt Nép Könyvkiadó, 1953. 55.
[52] Schreiber András: 3D idehaza, Magyar dimenzió. *Filmvilág*, 2006. nyár (https://filmvilag.hu/xista_frame.php?cikk_id=8665) Utolsó letöltés: 2022.05.17.
[53] Molnár Miklós egyetemi tanári jegyzetei, sztereoszkópiával kapcsolatos személyes írásai alapján.
[54] Ács Tibor Adrián: Háromdimenziós üzenetek a múltból.
Múlt-kor, 2009. tél (https://mult-kor.hu/20090217_haromdimenzios_uzenetek_a_multbol?print=1) Utolsó letöltés: 2018.04.08.
[55] P. Szabó Dénes: Térhatás most és régen.
Filmvilág, 2012. nyár (http://filmvilag.blog.hu/2012/07/16/terhatas_most_es_regen) Utolsó letöltés: 2018.05.10.
[56] Csákvári Géza: Üzenetek a jövőnek, 1973-ból, 3D-ben. *Nol*, 2017. nyár (http://nol.hu/kultura/20120725-uzenetek_a_jovonek-1320811) Utolsó letöltés: 2018.06.05.

a megtekintést. A korát meghaladó felvételek így hosszú éveikig a polcra kerültek. A portréfilmeket a Magyar Televízió Archívumának filmrestaurációs műszaki munkálatai után lehetett csak levetíteni. A nézők csak 2009-ben, az MTV 3D kísérleti adásakor láhatták végül a képernyőn. Riportfilmjei mellett, Molnár Miklós 3D krimisorozatot is tervezett. 1983 szeptemberében mutatták be az alig 10 perces Éjféli látogatók című kisjátékfilmet. A sorozat végül nem valósult meg, de az anaglif technikával vetített első képsorok és a térhatású tévéműsor nézőinek különleges élményben lehetett része.

Molnár Miklós tevékenysége, a sztereoszkóp képszámítással kapcsolatos műszaki munkássága a vezető hazai 3D stúdióinkban, napjainkban is ismert. Véleménye a távoli televíziós technológia jövőképét vázolja.

„Az újabb lépcsőfok pedig a hologramos tévé lesz. A képernyőn megjelenő tárgyat vagy színészt, nemcsak szemből, hanem más-más oldalról is megszemlélhetjük. Az lesz az igazi térbeli élmény. De ez még a jövő zenéje.” [57]

3.2. Holográfia, a ’teljes kép’ története

Az amerikai kutatók 1969-ben mutatták be a térhatású filmvetítéssel és televíziózással kapcsolatos kísérleti munkájukat a nagyközönségnek. A forradalmi technológia a sztereoszkópia, és a plasztikus képvetítés rendszerét igyekezett leváltani, amely az 1930-as évektől meghatározó szerepet töltött be a filmtechnika történetében. A vadonatúj megközelítés és a teljesen valósághű eredmények elkápráztatták a nézőket. A magyar sajtó így összegezte a kinti eseményeket: „Különös vetítés zajlott le nemrég New Yorkban. Világító tárgyak és mozgó alakok jelentek meg lebegve egy elsötétített terem közepén. A meglepett nézők nem szellemidézés, nem valamilyen túlvilági jelenet szemtanúi voltak, csupán egy háromdimenziós film pergett előttük.”[58]

Bár a filmvászon már korábban is helyet adott a 3D technikának, a speciális képi eljárások meglehetősen kevés nézőhöz jutottak el. A kutatók úgy vélték, a holográfia képes

[57] P. Szabó Dénes: Térhatás most és régen. *Filmvilág*, 2012. nyár (http://filmvilag.blog.hu/2012/07/16/terhatas_most_es_regen) Utolsó letöltés: 2018.05.10.
[58] Greguss Ferenc: Térhatású tv - hologram a képernyőn közvetítés lézersugárral. *Magyarország*, 1969. nyár (https://adtplus.arcanum.hu/hu/view/MagyarorszagUj_1969_2/?query=hologram%20a%20k%C3%A9perny%C5%91n%20k%C3%B6zvet%C3%ADt%C3%A9s%20l%C3%A9zersug%C3%A1rral&pg=438&layout=s) Utolsó letöltés: 2018.01.20.

lesz széles-körben is teret hódítani a háromdimenziós megjelenítés világában, leváltva a sztereoszkóp eljárást. A sztereoszkóp illúzióvilág nem volt képes biztosítani a teljes térlátás élményét, mivel nem lehetett a tárgyak mögé tekinteni. A plasztikus- és sztereófelvételekből valójában éppen a harmadik dimenzió hiányzott. A sztereó-lencsék, a tárgyakról visszapattanó fénysugarak közül pedig csak azokat rögzítették, melyek éppen a látószögükbe estek. A többi fénysugár gyakorlatilag elveszett, kimaradt a felvételekről. A fénysugarak utólagos rekonstrukciós módszere pedig részleges volt, így a nézői élmény sem teljesedhetett ki.

A legelső holográfiával kapcsolatos kísérlet 1959 decemberében P.V. Smakov szovjet professzor nevéhez köthető[59]. Ő mutatta be az első holo-tv adást, amely két kamera által rögzített kép volt. A felvételeken egy-egy tv-adás szerepelt, melyeket különleges módszerrel egymásra vetítettek.[60] A nézők a sztereo-eljárásból ismert polarizációs szemüvegen keresztül tapasztalhatták meg az egyedülálló élményt. Az első adások persze még sokkal inkább hasonlítottak a plasztikus elképzelésre, mint a későbbi, térben is kiteljesedő, valóban 3D alakot öltő, továbbfejlesztett megjelenítési módra. A képsorokat rögzítő technika még épp csak szárnyakat bontogatott. Nem volt képes a 3D-s alakzatok egészének tárolására, csupán a két kamera látószögének egy szeletét vizsgálhatta. Hasonlóképp működött a korábbi „légyszemű" fényképezőgép is, melyet a Nobel-díjas francia fizikus, Lippmann fejlesztett ki. A sok egyidejű fényképkészítés elve azonban nem működött jól a gyakorlatban.[61] A képsokaságot tekintve nem alakulhatott ki a térhatás, hiszen agyunk nem képes arra, hogy kettőnél több képet dolgozzon fel egységes képpé.[62]

A reform-technika alapelvét Gábor Dénes, Nobel-díjas magyar származású tudósunk dolgozta ki, több mint egy évtizeddel korábban, 1948-ban. Bravúros műszaki megközelítése elvi szinten már akkor lehetővé tette az olyan képek előállítását, melyek valóságos térhatást

[59] Cservény József: A televízió jövője. *Korunk*, 1957. tavasz (https://adtplus.arcanum.hu/hu/view/Korunk_1957/?query=P.V.%20Smakov%20professzor&pg=1151&layout=s) Utolsó letöltés: 2018.01.20.
[60] Kopülov Tacskov: Televízió és holográfia. Műszaki Könyvkiadó, Budapest, 1978. p.86
[61] Greguss Pál: A térbeli fényképezés: A holográfia. *Népszabadság*, 1968. nyár (https://adtplus.arcanum.hu/hu/view/Nepszabadsag_1968_06/?query=A%20t%C3%A9rbeli%20f%C3%A9nyk%C3%A9pez%C3%A9s&pg=202&layout=s) Utolsó letöltés: 2018.01.23.
[62] Greguss Pál: Holográfia - az információk rögzítésének új útja. *Haditechnika - Haditechnikai szemle* 2., 1968. 4. szám. (https://adtplus.arcanum.hu/hu/view/Haditechnika_1968/?query=Greguss%20p%C3%A1l&pg=139&layout=s) Utolsó letöltés: 2018.01.23.

adtak a hagyományos megjelenítő eszközökön is. Találmánya annyira megelőzte korát, hogy az ötvenes években a holográfia technikai okokból nem tudott lényegesen továbbfejlődni. Munkássága 1964-ben került újra előtérbe.

Leith és Upatnieks ugyanis sikeresen ekkor tudta rekonstruálni egy tárgy teljes háromdimenziós képét lézeres eljárásával. A lencse nélküli fényképezés módszerével hirtelen minden képi információ rögzíthetővé vált, újrajátszható, megismételhető formában. A holográfia története innentől gyors fordulatot vett. 1966-ra már közel száz amerikai kutatólaboratórium foglalkozott a magyar kutató ígéretes eljárásával.[63]

Gábor Dénest fiatalon is a fény mérésével kapcsolatos tudományos kérdések foglalkoztatták.[64] Angliában, tenisz mérkőzés közben fogalmazta meg munkásságának egyik legkritikusabb felvetését. Foglalkoztatni kezdte a kérdés, hogy vajon miért nem képes a technológia egy olyan egyszerű tárgy, mint a teniszlabda valósághű megörökítésére. Gábor Dénes, a képi megörökítés problematikájára így emlékszik vissza: „Már régen, először 17 éves koromban, kezdtem érdeklődni azok iránt a problémák iránt, amelyek végül is a holográfiához vezettek.. Feltettem magamnak a kérdést: ha fényképet készítünk, a kép megjelenik a lemez síkjában. … Annak az információnak, amely bemegy a képbe, ott kell lennie minden síkban a lemez előtt, és ugyanúgy a lencse előtt levő síkban is. Vajon milyen formában lehet ott? Miért nem tudjuk kivenni onnan, láthatóvá tenni? Kérdésemre teljes választ csak sok évvel később tudtam adni..." [65]

1948-ban a Holografika eljárással tudta megválaszolni korábbi felvetéseit. A felfedezés elnevezése a görög „Holos" szóból származik, mely egészet, osztatlant jelent.[66] A „holophote" szó, a világítótornyokban levő egyik felszerelés neve, mely a teljes fényerővel

---

[63] Bodó Barna: Mi a holográfia? *Megyei Tükör*, 1971. tél (https://adtplus.arcanum.hu/hu/view/HaromszekMegyeiTukor_1971_02/?query=Mi%20a%20hologr%C3%A1fia%3F&pg=75&layout=s) Utolsó letöltés: 2018.01.20.

[64] Ötletek, borotválkozás közben. *Tükör*, 1971. 49. szám. (https://adtplus.arcanum.hu/hu/view/Tukor_1971_10-12/?query=%C3%96tletek%2C%20borotv%C3%A1lkoz%C3%A1s%20k%C3%B6zben%20&pg=303&layout=s) Utolsó letöltés: 2018.01.15.

[65] Bodó Barna: Mi a holográfia? *Megyei Tükör*, 1971. tél (https://adtplus.arcanum.hu/hu/view/HaromszekMegyeiTukor_1971_02/?query=Mi%20a%20hologr%C3%A1fia%3F&pg=75&layout=s) Utolsó letöltés: 2018.01.20.

[66] Jánossy Mihály: A holográfia története. *Akadémiai Értesítő - Magyar Tudomány*, 1972. 7-8.szám. (https://adtplus.arcanum.hu/hu/view/AkademiaiErtesito_MATUD_1972/?query=hologr%C3%A1fia%20t%C3%B6rt%C3%A9nete%20&pg=546&layout=s) Utolsó letöltés: 2018.01.15.

bíró jelzőfény felhasználását teszi lehetővé. A második szótag a "gráfia", vagyis „Grapho", a *(képi)* írásra, összevonásra, rögzítésre utal.[67]

Kutatása a teljes képi információ tárolásának, későbbi hozzáférésének, előhívásának lejegyzése és módszertana. Tudományos munkásságának origóját a fényterjedés mátrix elméletében röviden összegezte. „Háromdimenziós fényképkészítés, lencsék nélkül, koherens sugarak segítségével, amikor is lehetővé válik az egész tárgy képének rögzítése". [68] A „lézersugár a tárgy hologramképét rögzíti a fényképezőlemezen, és amikor erre a lemezre előhívás után ismét lézersugarakat irányítanak, a tárgy vagy a tárgyak teljesen hű térbeli képét látjuk mögötte" [69]

Találmányának alappillére a fény amplitúdójának, továbbá fázisának mérése és rekonstrukciója a megfelelő pontokban. Az elméleti megközelítést hullámfront rekonstrukciónak nevezte el. Lényege, hogy a rögzített tárgyakat valós, térbeli másolatként jeleníti meg, teljesen eredeti formájában és kiterjedésében. Képi teljességének titka, hogy a tárgyról jövő fényhullám adatai mellett, a fény fázisainak információit is eltárolja. A képi megörökítést a hologramlemez fotólemezének bevilágítása, exponálása teszi lehetővé, mely hasonló a hagyományos fotólemezek előhívásához. "A holográfia lényegében lézersugarak „randevúja" fényérzékeny fotolemezen. Itt találkoznak a tárgyról visszaverődő sugarak az úgynevezett referenciasugárral, aminek következtében különös interferenciakép *(hologram)* alakul ki. " [70] A hologram minden információt őriz az eredeti alakzatról. A hologram különösen fontos tulajdonsága, hogy az egész tárgyat, mélységében is teljes élességgel vizualizálni tudja. Nem szükséges a fényhullámok fókuszálása. A teljes hologram egy részletének sérülése, eltávolítása után is rekonstruálni lehet az egész képet. Ez annak is köszönhető, hogy egy 10x10 cm-es fényképezőlemez közel 100 millió képpontot tudott

[67] Ozogany Ernő: A térhatású kép | 2. *A Hét,* 1978. 9. szám (https://adtplus.arcanum.hu/hu/view/Ahet_1978_1/?query=t%C3%A9rhat%C3%A1s%C3%BA%20k%C3%A9p%20%7C%202&pg=185&layout=s) Utolsó letöltés: 2018.01.25.
[68] Vajda Péter: A háromdimenziós portré. *Népszabadság*, 1971. 269. szám. https://adtplus.arcanum.hu/hu/view/Nepszabadsag_1971_11/?query=A%20h%C3%A1romdimenzi%C3%B3s%20portr%C3%A9&pg=151&layout=s) Utolsó letöltés: 2018.01.15.
[69] Vajda Péter: A háromdimenziós portré. *Népszabadság*, 1971. 269. szám. https://adtplus.arcanum.hu/hu/view/Nepszabadsag_1971_11/?query=A%20h%C3%A1romdimenzi%C3%B3s%20portr%C3%A9&pg=151&layout=s) Utolsó letöltés: 2018.01.15.
[70] Greguss Ferenc: Térhatású tv - hologram a képernyőn közvetítés lézersugárral. *Magyarország,* 1969. nyár (https://adtplus.arcanum.hu/hu/view/MagyarorszagUj_1969_2/?query=hologram%20a%20k%C3%A9perny%C5%91n%20k%C3%B6zvet%C3%ADt%C3%A9s%20l%C3%A9zersug%C3%A1rral&pg=438&layout=s) Utolsó letöltés: 2018.01.20.

tárolni. A kirajzolódó emlékképeket akár körbe is járhatjuk, minden oldalról látjuk, mivel a fotolemez minden milliméteren másként rajzolja ki a tárgyat. Színes képet rekonstruáló hologram különböző alapszíneket sugárzó lézerek segítségével készíthető el. A hologram tehát fekete-fehér, a színinformációt pedig a különböző sűrűségű interferenciacsík-rendszerek tartalmazzák. [71]

Film és televízió

Az eredeti elképzelés elméleti oldalról tehát már működőképes volt. A mozgóképes rögzítést is kivitelezhetőnek tekintetették. A tervek szerint a hologramot a kamera fényérzékeny rétegére vetítik, a mozgó elektronsugárból villamosjelekké alakítja a képelemeket. Az adás vételénél a hologramot lézersugárral átvilágítják és így jelenik meg a képernyőn. A holográfia megteremtette az elvi lehetőséget a háromdimenziós filmkészítéshez és a televíziós broadcast rendszerek fejlesztéséhez. A kutatók azonban komoly nehézségekbe is ütköztek a mozgóképgyártás terén. Egyik ilyen probléma a képelemek, képpontok számából fakadt. Míg a hagyományos standard formátumok esetén kisebb mennyiséggel kellett kalkulálni, a hologram közel 400 milliárd képelemből is állhatott. [72] „Szerencsére a hologramoknak van egy sebezhető pontjuk: túlságosan sok képpontból áll össze a háromdimenziós kép, mintha valaki bőbeszédűen mesélne el egy viccet. Ha lefaragják a képpontok számát, a hologram is belekényszeríthető mai tv-rendszerünkbe."[73]

A kulcs a felbontás csökkentése volt, melyre számos kísérlet született. Az egyik sikeres próbálkozás során egy diapozitívet fényképeztek le, és sikeresen továbbították a jelet. A további kísérletek arra vonatkoztak, hogy a diapozitívok felbontását növelni tudják, és minél nagyobb látószögből tudják a holografikus képet továbbítani. A térhatású vevőkészülékek létrejöttében döntő szerepe volt a termoplasztikus filmeknek is, melyeken négyzetcentiméterenként hatmilliárd információ adható át. [74] A sajtó egyre bizakodóbban számolt be a képi fejlesztésekről: "A lézersugáron alapuló holográfia hozzásegíthet a

[71] Jánossy Mihály: A holográfia története. *Akadémiai Értesítő - Magyar Tudomány,* 1972. 7-8.szám. (https://adtplus.arcanum.hu/hu/view/AkademiaiErtesito_MATUD_1972/?query=hologr%C3%A1fia%20t%C3%B6rt%C3%A9nete%20&pg=546&layout=s) Utolsó letöltés: 2018.01.15.

[72] Greguss Ferenc: Térhatású tv - hologram a képernyőn közvetítés lézersugárral. *Magyarország,* 1969. nyár (https://adtplus.arcanum.hu/hu/view/MagyarorszagUj_1969_2/?query=hologram%20a%20k%C3%A9perny%C5%91n%20k%C3%B6zvet%C3%ADt%C3%A9s%20l%C3%A9zersug%C3%A1rral&pg=438&layout=s) Utolsó letöltés: 2018.01.20.

[73] Uo.

[74] Uo.

tökéletes térhatású mozi és televízió kifejlesztéséhez is. Megvetheti az alapját a televízió-programok egyszerű konzerválásának, rögzítésének, majd a közönséges tv-vevőkön való visszajátszásának. Szakértői vélemények szerint ez lesz a lézereknek a fogyasztók legszélesebb köreit érintő, legfontosabb felhasználása. “[75]

Magyar szabadalom alapján készült az első térhatású holografikus film is, melyet Galilei életéről forgattak. A szakmai sajtó úgy számol be a vetítésről, hogy a nézők nemcsak a vásznon láthatták a képsorokat, hanem a közönség valós háromdimenziós térben figyelhette meg a színészeket és a tárgyakat. A rendezők úgy vélték, a filmvetítés új korszaka kezdődött, és „összhatásában felülmúlja az eddig készített ilyenfajta, sztereó-filmeket”.[76] A nyolcvanas évekre körvonalazódott a film és televízióadások technológiai fejlődésének igénye. A térhatás rendkívüli lehetőségekkel szélesítette a filmtechnikai kutatások irányát. A speciális nézőszemüvegek után, hatalmas lépést jelent a holografikus megjelenítés, amely a lencse nélküli valóság-leképzés úttörője.

Gábor Dénes úgy vélte, hogy a jövő 3D televíziója még várat magára. „meg lehet oldani, talán nem nagyon gyorsan, de azért szeretném még megélni. A háromdimenziós mozi jóval közelebb áll hozzánk időben, érdekesebb a 3 dimenziós televíziónál — ez utóbbi csak akkor lép versenybe, ha egy televízióképernyő egész falat fog betölteni, ennek elterjedése azonban már a jövő század regénye lesz.”[77]

A számítógépes animáció a hatvanas években egyre közelebb került a filmkészítéshez. Bár az erőforrások rendkívül költségesek és minimálisak voltak, a kísérletező jellegű munkának nem jelentettek akadályt.[78]

3.3. Korai fejlesztések és innovációk a magyar CG és CGI számítógépes animáció terén

[75] Várhelyi Tamás: A lézersugár fényes jövője. *Népszava*, 1970. tél (https://adtplus.arcanum.hu/hu/view/Nepszava_1970_12/?query=A%20l%C3%A9zersug%C3%A1r%20f%C3%A9nyes%20j%C3%B6v%C5%91je%20&pg=230&layout=s) Utolsó letöltés: 2018.01.15.
[76] Magyarok a nagyvilágban. *New Yorki Magyar Élet*, 1973. 35. Szám (https://adtplus.arcanum.hu/hu/view/NewYorkiMagyarElet_1978/?query=New%20yorki%20magyar%20%C3%A9let&pg=0&layout=s) Utolsó letöltés: 2018.01.15.
[77] Varga Zsuzsa: Fényképezés lézersugárral. *Pajtás,* 1975. nyár (https://adtplus.arcanum.hu/hu/view/Pajtas_1975_2/?pg=659&layout=s&query=f%C3%A9nyk%C3%A9pez%C3%A9s) Utolsó letöltés: 2018.01.15.
[78] M Tóth Éva, Kiss Melinda: Animációs mozgóképtörténet I. Budapest, Typotex Kiadó, 2014. 15.1

A magyar CGI animáció az 1970-es években bontakozott ki és eleinte még szinkronban tudott fejlődni a világ mozgóképtechnológiai környezetváltozásának ütemével is.[79] A CGI technika *(Computer Generated Image)*, azaz a számítógép által létrehozott képsorok jelentősége, hogy képes volt a filmkészítés korai eszköztárát animációs és videógrafikai munkafázisokkal kiegészíteni[80], illetve teljes egészében kiváltani. Segítségével az alkotók olyan technológiai lehetőséget kaptak, melyeknek korábban az analóg eljárások szabtak határt. Az CGI lehetőséget teremtett statikus alakzatok megelevenítésére, életre keltésére. A korszak alkotói sok esetben a megrajzolt videóelemeket egészítették ki a digitális elemekkel. A hatvanas évek szárnypróbálgatásai után az első grafikai anyagok az egyszerűbb geometriai formákkal, szabályosabb alakzatokkal tudtak látványosabb eredményt elérni. A számítógép által generált grafika evolúciójának gyökerei a XX. század közepéig vezethetők vissza.

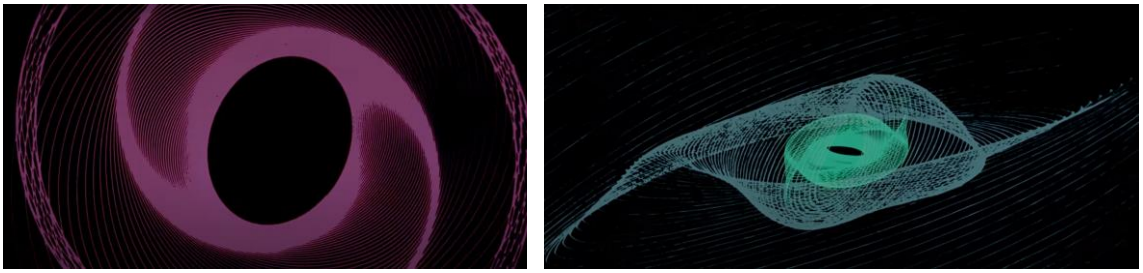
3.3.1. A magyar 3D CGI animáció nemzetközi kontextusban: filmtechnológiai előképek és hatások

Az 1950-es évek a CGI történetének origójaként is értelmezhető, mivel ekkor kezdtek el először mechanikus számítógépeket használni az animációs cellák, azaz animált celluloid képkockák mintáinak generálásához. Ezen mintázatok nem csupán izolált képkockák formájában maradtak, hanem hosszabb vizuális sorozatokká, sőt, teljes filmalkotásokká álltak össze. [81] Ez a folyamat egy újszerű, interdiszciplináris kapcsolatot hozott létre a számítástechnika-tudomány és a filmgyártás releváns szakterületei között[82], és jelentős hatással bírt a magyar CGI alkotók első munkáira.

Az első olyan film, amely CG(I) technológiát használt, Alfred Hitchcock 1958-as műve, a *Szédülés* volt. John Whitney számítógépművész egy átalakított anitballisztikus

[79] Varga Zoltán: A magyar animációs film: intézmény- és formatörténeti közelítések. Szeged, Pompeji Alapítvány, 2016. p. 280.
[80] Zemankó Anikó: Giccsszerelmek tengerfenéken, őszinte kapcsolatok pedig a süllyedő hajón - Titanic Filmfesztivál. *Apertúra,* 2010. tavasz (http://magazin.apertura.hu/film/giccsszerelmek-tengerfeneken-oszinte-kapcsolatok-pedig-a-sullyedo-hajon-titanic-filmfesztival-2010/843/) Utolsó letöltés: 2019.06.20.
[81] Judson Rosebush: A chronology of animation history. A history of computer animation. Chron4.doc, 1989. (https://studylib.net/doc/8170363/chapter-4--a-history-of-computer-animation---a) Utolsó letöltés: 2023.08.18.
[82] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: *Magyar művészek és a számítógép*. Budapest, HUNGART, 2016. p16.

légvédelmi célzó számítógépet használt fel a főcím képsorainak elkészítéséhez. Az analóg komputerrendszeren a film szédüléses illúzióját keltő spirálokat rajzolt.[83]



A számítógépes animáció első filmkockái.
Forrás: Vertigo - Alfred Hitchcock, 1958.

A CG, vagyis a *komputergrafika* kifejezést elsőként az amerikai Boeing repülőgépgyár tervezője, William Fetter használta 1960-ban.[84] A vállalat számára 3D rácsstruktúra-rendszerű *(wireframe jellegű)* emberi figurákat terverezett, melyekkel egy pilótafülke ergonómiai adatait biztosították a vezérlőpult műszereinek elrendezését tervező mérnökök számára. Fetter pár évvel később, 1965 mutatta be *first-person (vagy első személyű)* CGI 3D repülőgép-szimulációját a *Carrier Landing*-et. Ebben az időszakban jelentek meg Edward Zajec számítógéppel generált, térbeli szatellit pályát modellező vonalrajz műve is: a *Simulation of a Two Gyro-Gravity Gradient Attitude Control System* (1963). [85]

Az első sztereoszkópikus 3D számítógépes animáció 1965-ben készült el. Michael Noll, a *Hypercube (1965)* című alkotásában valósította meg egy *négydimenziós* hiperkocka *(kocka-a-kockában)* forgó animációját.

Az első művészeti célú CGI animációt az Amerikában élő, magyar származású Charles Csuri, *Kolibri (1967)* című munkájában láthatta a közönség. A *FORTAN* programozási nyelven íródott, harmincezer képből álló digitális alkotása egy kolibrimadár morfolódó vonalrajz-mozgását utánozta.[86] Csuri később *a Computer Graphics Research Group*-ban fejlesztette ki az *Animát*, az első komputeranimációs nyelvet.

[83] Tóth Edit: Hitchcock Szédülése és Kepes György fényművészete a háború utáni nagyvárosban 1. rész. *Balkon*, 2017. ősz (http://epa.oszk.hu/03000/03057/00099/pdf/EPA03057_balkon-2017-10_021-025.pdf) Utolsó letöltés: 2023.08.26.
[84] Najmányi László: A MakerBot 3D nyomtató és társai. *Artportal*, 2011. nyár (https://artportal.hu/magazin/najmanyi-laszlo-a-makerbot-3d-nyomtato-es-tarsai/) Utolsó letöltés: 2018.08.27.
[85] Aaron Connolly: The History of 3D Animation. *Animated Explainers*, 2021. tél (https://animationexplainers.com/the-history-of-3d-animation/) Utolsó letöltés: 2018.08.23.
[86] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: *Magyar művészek és a számítógép*. Budapest, HUNGART, 2016. p16.

A korszak egyik legjelentősebb újítása a digitális mozgáskövetés (*motion capture)* volt. Lee Harrison *The stick man (1967)* című alkotásban használtak először a 3D tervezésben ismert objektum-hierarchia rendszert*(parenting),* de ismert volt a komplexebb képi világgal rendelkező *Beautiful Noise (1968), a The Dynne* (1968) és a *Turn On! Dancer (1969)* című alkotás is. Ekkoriban készült el a *Mr. Computer Image* nevű animáció, mely broadcast grafikai bemondót biztosított volna az amerikai elnökválasztás estéjére. Az ABC televíziós társaságnak készült rövid, promóciós jellegű videóban először jelenik meg beszélő CGI karakter.[87] Népszerű volt még az oxfordi *The Flexipede* című vonalrajz-animáció is, melyet Tony Pritchett készített el a Londoni Egyetem *Atlas* nevű számítógépével.[88] Stanley Kubrick rendezésében, Arthur C. Clarke és Kubrick forgatókönyve alapján készült el a *2001: Űrodüsszeia (1968)* című film, melyben Douglas Trumbull, VFX supervisor az ún. slit-scan fotózást alkalmazta.[89]

A Kanadai Nemzeti Filmalapban 1969-ben indult el a kísérletezés az animációhoz kapcsolódó számítógépes technológiákkal. A magyar származású animációs művész, Földes Péter, 1971-ben készítette el a *Metadata* című filmjét. A szerző úgy animálta a rajzokat, hogy az egyik képből fokozatosan áttért a következőre, egy olyan technikával, amit *inbetweening* interpolációs eljárásként ismerünk. A *Metadata*, akárcsak a *The Apterix and the Easter Bunny (Rend.: Johnny Hart, 1970)* című animáció, az elsők között vetített színes animált képsorokat a közönségnek. Az Oscar jelöléssel kitüntetett Földes Péter a vektorgrafikus eljáráshoz hasonló módszerekkel készítette el az *Éhség (1974)* című alkotását is.[90] A korszak kiemelkedő komputerművésze volt a szintén magyar származású Jules Engel *(Engel Gyula)* is, aki „tiszta grafikai koreográfiának" nevezte alkotásait.

A 3D CGI nagy áttörését 1972-ben hozta meg Edwin Catmull és Fred Parke *A Computer Animated Hand*, vagyis *Egy számítógépesen animált kéz* című rövidfilmje. Ebben a műben Catmull 350 háromszög és poligon digitális reprezentációját készítette el, amelyeket tintával rajzolt a kezére. Az adatok számítógépes rögzítése után, egy egyedi szoftvereljárással

[87] Kerlow, V. Isaac: The Art of 3D. *Computer Animation an Effects.* Hoboken, John Wiley&Sons, Inc., 2004. p16.
[88] Alan Dorin: Moving Pictures. *Monash.edu*, 2009. ősz (https://users.monash.edu/~cema/courses/FIT3084/lectureNotes/lect23.html) Utolsó letöltés: 2023.08.30.
[89] David Parkinson: Douglas Trumbull, visual effects visionary behind 2001 and Blade Runner, 1942 to 2022. BFI, 2022. tél (https://www.bfi.org.uk/news/douglas-trumbull-1942-2022) Utolsó letöltés: 2023.08.30.
[90] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: Magyar művészek és a számítógép. Budapest, *HUNGART*, 2016. p22.

vizualizálta a 3D alakzatot. Fred Parke a *Faces and Body Parts (1974)* című művében vitte tovább korábbi forradalmi eljárásait.

Ebben az időszakban jelent meg Alan Kitching, az *Atlas Laboratórium* tervezőjének alkotása is, a *Flying Cube (1973).* A világ első 3D poligonos kockaanimációja a Fortran alapú *Antics* nevű szoftver funkcióit prezentálta. [91] A *Feltámad a vadnyugat (Michael Crichton: Westworld, 1973.)* című filmben látható képi trükk elkészítéséhez is számítógépes animációt használtak. Az android szereplő nézőpontját bemutató gépies, erősen pixeles-hatású renderek fontos mérföldkőnek számítanak, hiszen a CGI történetében ez volt az első olyan jelenet, ahol a számítógéppel generált képeket élőszereplős felvételekkel kombinálták.[92] Richard T. Heffron *Eljövendő világ (Futureworld, 1976)* című filmje az informatikai alapú vizuális művészet korai időszakának egyik meghatározó alkotása volt. A film vizuális effektusainak szakértői először használtak realisztikus, számítógéppel generált modelleket egy emberi arc ábrázolására.

A korszak kiemelkedő nemzetközi alkotásai és egyedülálló eredményei ellenére még évekbe telt, mire a programozók, komputerművészek és az animációs alkotók teljesen ki tudták használni a számítástechnika által nyújtott lehetőségeket.

A magyar CGI úttörő képsorai is ebben a megújuló környezetben készülhettek el, a nemzetközi gyakorlatban bevált multidiszciplináris technológiai modellek alapján.

3.3.2. Mikrobi

Mata János rendezésében valósult meg a magyar filmtörténet első olyan képsora, melyben CGI animációt alkalmaztak.

A *Mikrobi* című animációs sorozat az azonos nevű, 52 részes rádiójátékból született. Mozgóképes adaptációjának alapgondolata Kovács Bélától, a Magyar Televízió akkori Ifjúsági és Gyermekosztályának főszerkesztőjétől származik, aki egy gyermekrajzpályázatot hirdetett a főhős karakterének megformálására.[93] A rajzfilm forgatókönyvét ezután Bálint

[91] Alan Kitching: The Antics computer animation system. *Atlas*, 1975. (https://www.chilton-computing.org.uk/acl/applications/graphics/p003.htm) Utolsó letöltés: 2023.08.28.
[92] Herb A. Lightman: Behind the Scenes on Westworld: AC Talks to Writer-Director Michael Crichton. *American Cinematographer*, 2020. tavasz (https://theasc.com/articles/behind-the-scenes-on-westworld) Utolsó letöltés: 2023.08.28.
[93] Kiricsi Zoltán: A robot, aki utált mosogatni. Comment:com, 2006. tél (https://comment.blog.hu/2006/12/14/mikrobi_1) Utolsó letöltés: 2018.05.26.

Ágnes készítette el. A *Mikrobi* 1973 és 1975 között a Magyar Televízió kiemelkedően népszerű rajzfilmsorozata lett. Képsoraiban a robot űrhajójának egyes mozgásfázisait számítógépes trükk segítségével dolgozták ki.[94]

A 3D űrhajó felemelkedésének és térbeli mozgásának pillanatait egyedülálló szoftveres eljárással Kassai Árpád villamosmérnök valósította meg[95]. A *Pannónia Filmstúdió* rendezője, Mata János így emlékezett a fejlesztő korszakalkotó munkájára: „A Központi Fizikai Kutató Intézetnek volt egy számítástechnikai fejlesztő részlege. Szükségük volt egy animátorra, így keveredtem én ebbe a dologba. Animációs programot kellett fejleszteni. Igen egyszerű módszerekkel, hiszen akkor még 360 k volt egy winchester. És ezen belül kellett megoldani olyan feladatokat, amelyek ma másfél-két gigás területet igényelnek. A rakétát megrajzoló program azonban nem ott készült, hanem egy Kassai nevű programozó srác írta meg otthon. A lángokat viszont kézzel kellett megrajzolni, kifesteni."[96]

A *Mikrobi* egyedülálló képsorainak technológiai háttere megközelítése Európa szerte is újdonság volt. [97] Az algoritmikus és adatalapú alkalmazáslogika elősegítette a digitális média és a hagyományos filmkészítés konvergenciáját. Ennek következtében nem csupán a filmkészítők, hanem a programozók, a grafikusok, animátorok és az informatikai szakemberek is egyre több közös platformon dolgozhattak, ami hozzájárult a komplex multimediális alkotások megjelenéséhez.

3.4. Programozott film és adatalapú jelenetsorok

Az 1970-es évek közepén a filmkészítésben egyre jobban felerősödő interdiszciplináris együttműködések nyomán az animációs projektek tovább integrálták az informatikai és programozási technológiákat és eljárásokat. Bár a komputeranimáció és a programozás közötti kapcsolat ebben az időszakban még nemzetközi mércével is kezdeti fázisban volt, már akkor is jelentős hatást gyakorolt a filmipari és a kapcsolódó tudományos diszciplínákra. Az innovatív technikák nem csak az esztétikai megújulás lehetőségét kínálták, hanem a technológiai paradigmaváltás ígéretét is magukban hordozták. A programozott

[94] M Tóth Éva, Kiss Melinda: Animációs mozgóképtörténet I. Budapest, Typotex Kiadó, 2014. 15.1
[95] Varga Zoltán: Cseh András és Mata János animációi. Papírvizsla, madárkomédia. *Filmvilág,* 2018. tavasz (https://www.filmvilag.hu/xista_frame.php?cikk_id=13571) Utolsó letöltés: 2023.08.25.
[96] Kiricsi Zoltán: A robot, aki utált mosogatni. *Comment:com,* 2006. tél (https://comment.blog.hu/2006/12/14/mikrobi_1) Utolsó letöltés: 2018.05.26.
[97] Kiricsi Zoltán: Mikrobi és a derékszög generátor. *Comment:com,* 2006. tél (https://comment.blog.hu/2006/12/22/title_1820) Utolsó letöltés: 2018.05.26.

alkotások, az adatalapú animációs filmek a filmtechnológia-történetének fontos mérföldkövei lettek, és új utat nyitottak az interaktív média területein.

3.4.1. Az adatalapú komputeranimáció kialakulásának nemzetközi kontextusa a kezdetektől a magyar „proxemikai sémák" megjelenéséig

Az adatalapú komputeranimáció egy olyan technológiai módszer, amelyben a karakterek, tárgyak és események mozgása és viselkedése programozott adatstruktúrákon és algoritmusokon keresztül kerül meghatározásra és szimulálásra. Ebben a generatív rendszerben a grafikus elemek nem egyszerűen előre animált szekvenciák, hanem dinamikus, adatvezérelt modellek, amelyek lehetővé teszik a komplexebb és interaktívabb vizuális narratívák létrehozását. Az ilyen típusú animációk adaptívabbak és rugalmasabbak, mivel a szereplők és a környezet változó paraméterek alapján reagálhatnak. Az 1960-as években számos nemzetközi kísérlet indult a technológiai innovációk felfedezésére. 1964 után egyre több kiállításon, komputeranimációs szimpóziumon jelenik meg a számítógép és a művészet interdiszciplináris témaköre.

1965-ben Németországban került megrendezésre az első generatív grafikai alkotásokra fókuszáló kiállítás, a *Generative Computergrafik*. Ezen az eseményen mutatták be Max Bense és Georg Nees matematikusok az ALGOL programozási nyelven készült művüket, a *Kreisbogengewirre*-t *(Körívzavar)*. [98]

Tokióban a *Computer Technique Group,* Buenos Aires-ben a *CCEAC (Centro de Estudios de Arte y Comunicación)*, Párizsban a *GRAV (Groupe de Recherche d'Art Visuel)* és Bécsben az *Ars Intermedia csoport* voltak a komputeranimáció úttörői. Az évtized végére a számítógépes grafika Hollandiában *(R. D. E. Oxenaar)*, Brazíliában *(Waldemar Cordeiro),* Spanyolországban *(Eusebio Sempere)*, Olaszországban *(Auro Lecci)*, Jugoszláviában *(Zoran Radovic´)* és Csehországban *(Zdene˘k Sýkora)* is elismert művészeti médiummá vált.[99]

A szovjet filmipar sem maradt ki a komputeranimáció algoritmikus művészeti ágának nemzetközi fejlődési üteméből, sőt, saját innovatív módszerekkel is hozzájárult az animáció új formáinak kibővítéséhez. 1968-ban egy fizikusokból és matematikusokból álló csoport a macska mozgásának szimulációját hozta létre egy teljesen egyedi program fejlesztésével. A

[98] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: Magyar művészek és a számítógép. Budapest, HUNGART, 2016. p20.
[99] u.o.

Moszkvai Egyetem professzora, Nikolay Nikolaevich Konstantinov vezetésével az alkotók több száz, macska formájú numerikus mintát generáltak és egyenként fényképezték le őket. A *Kitty (Koshechka, 1968)* az egyik első kísérlet a digitális formában realisztikus állati mozgás megjelenítésére.[100]

A külföldön élő, magyar gyökerekkel rendelkező alkotók is a korszak kiemelkedő úttörői közé tartoztak.

A korábbam említett *GRAV* csoport egyik magyar származású alapítója, Molnár Vera, az algoritmikus művészet úttörője volt. *Machine Imaginaire* sorozatában matematikai arányrendszerekre építette műveit. Julesz Béla generatív algoritmus alapján létrehozott véletlenszerű *Random Dot Picture* eljárással készülő komputergrafikai műalkotásokat készített.[101] Az magyar származású Ausztráliában élő Frank Eidlitz, és a kanadai *Computer Graphics Group* alapítója, Leslie Mezei is kísérletező úttörője volt a számítógépes művészeteknek.

A nemzetközi generatív képkorszak határán Énekes Ferenc elsőként készített komputergrafikát Magyarországon. A programozó-matematikus FORTAN nyelven alkotta meg Koncz Zsuzsa énekesnő portréját[102].

Az első magyar generatív komputer-portré, és a *Mikrobi*-ban bemutatott 3D CGI mozgóképsorok után nem sokkal egy újabb innovatív mozgóképtechnológiai eljárás is megjelent.

3.4.2. Pszichokozmoszok

Az első adatalapú komputeranimáció Bódy Gábor és Szalay Sándor nevéhez köthető. Bódy a Balázs Béla Stúdióban dolgozott fiatal rendezőként. A filmalkotók számára kísérleti csoportot alakított, melyben a komputerfilmmel közösen kísérleteztek.

1976-ban mutatták be a magyar filmtörténet egyik legizgalmasabb animációs alkotását, amely korának egyik legkiemelkedőbb mozgókép-technikai fejlesztése volt. Bódy

[100] Wilson, Booth: Computer animationacross the iron curtain: early digital character design in Kitty. *Animation Journal*, 2013. (https://www.academia.edu/5154005/Computer_Animation_Across_the_Iron_Curtain_Digital_Character_Design_in_Kitty_1968_) Utolsó letöltés: 2023.08.21.
[101] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: Magyar művészek és a számítógép. Budapest, HUNGART, 2016. p18.
[102] uo.

Gábor kísérleti művét olyan rendkívüli háttérmunka és grafikai fejlesztés-sorozat előzte meg, amely a jelenkorig előremutató és egyedülálló maradt a hazai videóművészetben. A *Pszichokozmoszok* teljesen számítógép által előállított képsorozata alakzat-rekonstrukciós és adatvezérelt, reaktív vizuális ábrázolásmódot tudott képernyőre festeni. A Szalay Sándor elméleti atomfizikus segítségével készült képkockák vizuális kódokkal megformált elképzelése a mai legmodernebb 3D MI animációk előfutáraként is értelmezhetők. A film az ELTE Atomfizikai Tanszékének TPA 1001/i típusú, integrált áramkörökkel ellátott lyukkártyás számítógépén, egy termodinamikai és kvantummechanikai jelenségek leírására alkalmas algoritmus adaptálásával készült.[103] A „sejtautomata-modell" kölcsönhatásokra épülő kialakítását John Conway az *Életjátéka* című alkotásából örökítette tovább a rendező.[104]

A *Pszichokozmoszok* alkotópárosának nevéhez köthető tehát az első magyar programozott filmalkotás. A modern VFX[105] részecskerendszereihez hasonló képi eljárás segítségével megvalósult 12 perces animációban a digitális karakterek elrendezési sémái váltakoztak. Az alkotók egyedi alkalmazással képesek voltak „olyan diagramok, sémák megalkotására, amelyek a részek térbeli mozgásviszonyainak feszültsége révén különböző pszichikai érzeteket keltenek". [106] A 35mm szalagra felvett fekete-fehér szekvenciákban kirajzolódó látványelemek egymással kölcsönhatásba tudtak lépni. Szalay Sándor fejlesztésének jelentősége, hogy megteremtette a többszereplős adatfilm dinamikus technológiai keretét, melyben a szereplők folyamatosan változó komplex tulajdonság- és kapcsolatrendszerei automatikusan *(újra)*szervezik a történet egészét. Ez egy lényeges előrelépés Michael Noll *Computer Ballet (1965)* című munkájához képest, ahol bár már megjelentek többszereplős animációs szekvenciák, a főhősök nem rendelkeztek automatikusan változó adatkarakterisztikával, és a szereplők sem álltak interakcióban egymással. Ezzel szemben, a Pszichokozmozok története a vetítés pillanatában is alakítható volt. A néző már a legelső képsorokon szembesül a film generatív vizualizációs elméletével.

[103] Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: Magyar művészek és a számítógép. Budapest, HUNGART, 2016. p20.
[104] uo. p24.
[105] Filmtrükkök, vizuális effektusok.
[106] Varga Zoltán: A magyar animációs film: intézmény- és formatörténeti közelítések. Szeged, Pompeji Alapítvány, 2016. p. 281.

„Egy történetet nem kell végigírni. Elég a szereplőket és a történet szabályait megalkotni. Ezután már csak az a feladatunk, hogy megfigyeljük az eseményeket. Ha jónak látjuk, változtathatunk a szerelők tulajdonságain vagy a történet szabályain.” [107]

Bódy a továbbiakban úgy jellemezte a Balázs Béla Stúdióban készült kísérleti filmalkotást, mint számítógépes modell-kísérletet egy történet konstruálásában.

A számítógép által generált komputeranimációs alkotás, értelmezhető egy örökké újjászülető filmként is, mely képes az önálló cselekmény kialakítására, előre-paraméterezett környezetben képes a szereplők viselkedését valós *(játék)*időben alakítani. A grafikus elemeket olyan jellemzőkkel ruházták fel, melyek rugalmas és alakítható szabályrendszer segítségével definiálták a felületet. Az *agresszív, védekező és közömbös* határozókat ezután a gépi logikára, véletlenszerű szabályozásra bízták a tervezők. Amennyiben a rendező nem elégedett a kialakított *(vagyis kialakult)* történettel, bármikor változtathattak a szereplők tulajdonságian és a bonyolítás szabályain is.

A Pszichokozmoszok programozott képsorai doktori munkásságomra is jelentős hatással voltak, és további kutatói tevékenységem irányát is megadták.

3.4.3. A Pszichokozmoszok újraírásának kísérete

Másodéves doktori kutatóműhelymunkám során lehetőségem volt egy prezentáció keretében bemutatni a Pszichokozmoszok újraírt történetét. A film kísérleti jellegű újratervezése több mint négy évtizeddel a premier után sem volt egyszerű feladat számomra. Az interaktív grafikai programnyelvek alapján újra-modellezett film bemutatásakor megpróbálttam képernyőre vinni a modulárisam paraméterezhető karaktereket, emellett szabályozható történeti sebességet, és dinamikus karakterszámú vezérlést adtam a rendező (*azaz a néző*) kezébe.

Egy újratervezhető, monitorról átvett kísérleti-történet technikai környezete 3D web engine-k nélkül is kialakítható, amennyiben a megjelenő karakterek nem igényelnek komplex grafikai módszertant. Könnyűszerrel illeszthető a film cselekménye a böngészők adaptív képernyőjére, HTML5 Canvas-ra, vagy SVG-re, egyszerűbb JavaScript könyvtárak

[107] Pszichokozmoszok (Bódy Gábor, 1976)

segítségével [108]. Bár a történet újraírható az itt részletezett módszerekkel, összetett vizualizáció esetén már további alkalmazások bevonása is nélkülözhetetlen lenne.

Az interaktív történetmesélés megkerülhetetlen modern módszere a WebGL és a WebGPU alapú web 3D technológia lett. A következő fejezetek ebből a kontextusból próbálják feltárni a modern adatfilmes és broadcast eszközöket, a legkorszerűbb eljárásokat, és átörökíteni a hazai mozgóképtörténet vizsgált módszereit a legújabb technológiákra.

[108] Crawford, C. HTML5 Canvas: The Basics. *IBM*, 2023. (https://www.ibm.com/developerworks/library/wa-html5-Canvas-basics/) Utolsó letöltés: 2023.02.03.

4. Web 3D technológia

Ebben a fejezetben összefoglalom a web 3D technológia működési mechanizmusait. Bemutatom a WebGL és a WebGPU API egyedi és közös jellemzőit, azonosítom a technológiák felépítésének kulcsfontosságú különbségeit, és ismertetem az alkalmazási lehetőségeket. A vizsgálatot az interaktív digitális terek megalkotásának aspektusából kezdem.

4.1. A Web 3D technológia bemutatása és alkalmazási lehetőségei a művészeti produktumok aspektusából

A Web 3D fogalom olyan eljárássorozatot jelöl, mely során a 3D modellek és animációk létrehozásához használt alkotói szoftverkörnyezetben *(pl.: Blender, Autodesk Maya, Autodesk 3ds Max)* exportált objektumok különféle transzplantációs eljárásokon keresztül dinamikus alkalmazáskörnyezetbe adaptálhatók. A módszer lehetővé teszi a művészek és fejlesztők számára, hogy kellően részletgazdag, később pedig valósághű 3D-s környezetet alakítsanak ki, megszakítva a napjainkban még igazolható ellenérzést a kisebb felületszámú, alacsonyabb kidolgozhatóságú vizuális végeredménnyel szemben. A modern web 3D élő adások környezetében, interneten, applikációs környezetben és a kiterjesztett valóság eltérő digitális színterein is képes lesz a tervező művészi koncepcióját megőrizni, szempontjait eszközfüggetlenül érvényesíteni, és az elképzeléseket az eredeti ideának megfelelően vászonra rajzolni.

Napjaink kutatási- és fejlesztési irányai a jelenettulajdonságok megőrzését, a nagyteljesítményű vizualizációs képességet, továbbá a bonyolult transzplantációs folyamat egyszerűsítését támogatják.[109] Olyan hatékony eljárások implementálása szerepel a kitűzött célok között, melyek már párhuzamos adatszámítással biztosítják az összetett grafikai folyamatok végrehajtását. A nagyobb teljesítmény, a kibővített eszközkészlet biztosítja a korábban nélkülözött vizuális elemek újra-integrálását.[110] Az így megvalósuló 3D tervezői munkafolyamat eredménye azonban még semmiképpen sem tekinthető a végleges művészi produktum publikálásra alkalmas megfelelőjének.

[109] Google LLC: WebGPU API Specification. Chrome Platform Status. *Google*, 2022. (https://chromestatus.com/feature/6213121689518080) Utolsó letöltés: 2023.03.04

[110] François Beaufort & Corentin Wallez: Access modern GPU features with WebGPU. *Web.dev*, 2023. (https://web.dev/gpu/) Utolsó letöltés: 2023.03.07.

A fejlesztői utómunka, a 3D-s színterek adaptációja korántsem értelmezhető egyszerű feladatnak online környezetben. A tartalom integrálása és manipulációja a webes alkalmazásokban olyan keretrendszereket igényel, amelyek során a művészi produktum programozási oldala is könnyen érvényesíthető. Ilyen könyvtárak közé tartozik például a *Three.js*, a *Babylon.js*, vagy és az *A-Frame*, amelyek JavaScript alapú megoldást kínálnak a 3D-s tartalom kezelésére. Ezek a keretrendszerek elősegítik a gyors prototipizálást és a kódolással töltött idő csökkentését, miközben az egyszerűbb és hatékonyabb adaptációs módszereket is biztosítják. A grafikus 3D könyvtárak működése a disszertáció alapfogalmainál is tárgyalt technológiákra épül.

A WebGL és WebGPU webes grafikai alkalmazások létrehozására szolgáló API-k, amelyek lehetővé teszik 2D és 3D illusztrációk megjelenítését a böngészőkben. A plugin-ek és kiegészítő nélkül működő grafikai könyvtárak olyan felbontás- és platformfüggetlen eszközök, melyek a GPU teljesítményét kihasználva képesek magasminőségű 3D grafikai tartalmak előállítására.[111] A W3C *(World Wide Web Consortium)* által szabványosított technológiák számára közös specifikációk deklarálják a kompatibilitást, biztosítva a kreatív fejlesztők számára, hogy a rendelkezésre álló hardveres erőforrásokat technológiától függetlenül ki tudják használni. A jövő JavaScript alapú keretrendszerei és könyvtárai tovább könnyítik majd az egyedi applikációk kialakítását a broadcast mozgóképes szakemberek számára is.

Ennek megfelelően a WebGL és a WebGPU technológiák televíziós elterjedése jelentősen növekedni fog a következő években is, és mindkét eljárás számtalan jövőbeli videografikai alkalmazás alapjaként értelmezhető. Az új algoritmusok és optimalizációs technikák bevezetése növelni fogja a megjelenítés grafikai teljesítményét és minőségét, emellett a mesterséges intelligencia és a gépi tanulás integrációja új dimenziókat nyit majd a broadcast tartalomelőállításban is. A WebGL és a WebGPU technológiák fejlődése, a felhasználói kezelőfelületek kialakítása, automatizálása hozzájárulhat ahhoz, hogy a webes grafikus alkalmazások egyre inkább elérhetővé váljanak a 3D animátorok, és tervezőgrafikusok számára is. A fejlesztők és a művészek számára ez egyaránt új lehetőségeket, kreatív eszközöket teremt a digitális mozgóképtartalom kialakítására. Annak

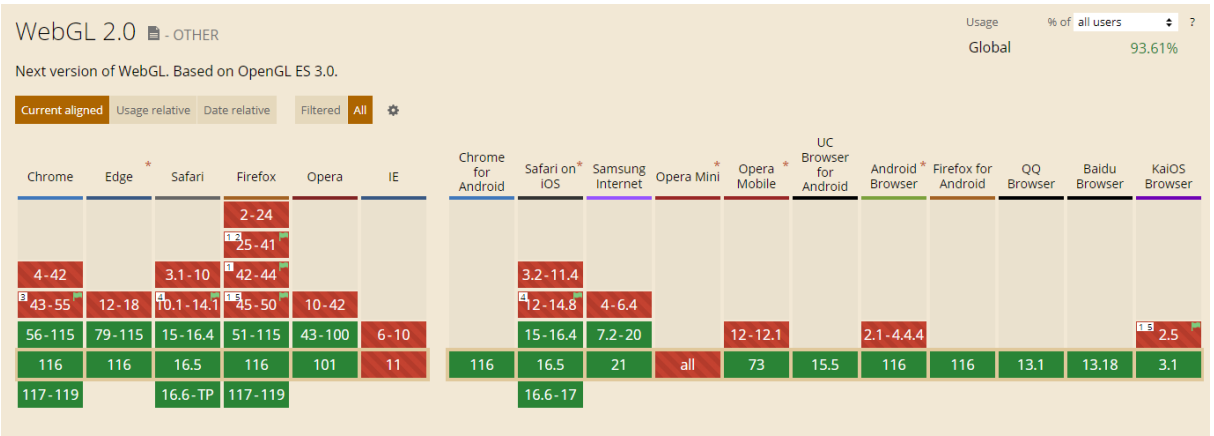
[111] Google LLC: WebGPU API Specification. Chrome Platform Status. *Google*, 2022. (https://www.chromestatus.com/feature/5682474635577344) Utolsó letöltés: 2023.03.04

ellenére, hogy a két technológia azonos karakterisztikákkal jellemezhető, eltérő piacra kerülési ütemezésük, fejlesztői eszköztáruk, és böngészőtámogatásuk miatt külön ismertetésük szükséges.

A következő bekezdés azt az alkalmazásprogramozási felületet ismerteti, mely az első sikeres 3D grafikai integrációs eszköz volt a böngészők világában.

4.2. WebGL alkalmazásprogramozási felület - a 3D grafikai integráció első sikeres lépése a böngészők világában

A WebGL megjelenése óta, az elmúlt egy évtized alatt széles körben ismert technológia lett a programozó szakemberek körében. Napjainkban a webes grafikai applikációkörnyezet széleskörben alkalmazott, elsődleges 3D eszköze.



[WebGL támogatás eltérő böngészőkben. Caniuse.com, 2023.nyár (https://caniuse.com/?search=webgl) Utolsó letöltés: 2023.08.31]

Számos grafikus fejlesztői 3D JavaScript könyvtár *(Three.js és a Babylon.js stb.*) a WebGL-re épül, és könnyen használható felületet biztosít a komplex térbeli alkotások számára.

A WebGL architektúrája magában foglalja a grafikus műveletek végrehajtásához szükséges JavaScript API-t, a shader programok létrehozásához szükséges GLSL-t *(OpenGL Shading Language),* továbbá a WebGL Context objektumot, amely a grafikus erőforrások kezelését és a renderelési pipeline vezérlését biztosítja.[112] A WebGL programozási modellje GLSL-t használ shader programok írására, melyek közvetlenül a GPU-n futnak és felelősek a grafikus objektumok megjelenítéséért, és árnyalásáért. A WebGL *vertex shader*-e

[112] W3C Community Group: GPU for the Web. WebGPU API Specification. *W3C*, 2021. (https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2023.01.25.

meghatározza a 3D objektumok csúcspontjainak helyét és tulajdonságait, míg a *fragment shader* a pixel-színek és textúrák definiálását adja.

A WebGL *vertex shader* a 3D objektumok csúcspontjainak helyzetének, normáljainak és egyéb attribútumainak transzformációjáért és átviteléért felelős.[113] A transzformációk magukban foglalják a modell-, néző- és projekciós mátrixok alkalmazását, amelyek meghatározzák az objektumok helyzetét a virtuális térben, a nézőpont és a kamera viszonyában. A *fragment shader*, más néven *pixel shader*, a végső színek és textúrák értelmezéséért felelős a képernyőn megjelenő képpontokon. Olyan eljárásegyüttes optimalizálja tehát a megjelenítést, mely segít az egyedi alkalmazások számára abban, hogy automatikusan alkalmazkodjanak a nézői eszközökhöz, így kialakítva egy rugalmas és könnyen adaptálható rendszert.

A WebGL széleskörben elterjedt, könnyen hozzáférhető, platformfüggetlen és a böngészők *(Chrome, Mozilla Firefox, Microsoft Edge és Apple Safari)* natívan támogatják.[114] Kompatibilis a legtöbb modern online technológiával, beleértve az HTML5-t, CSS3-t és JavaScript-et. Fejlesztői szempontból a WebGL könnyen integrálható új, vagy meglévő webes alkalmazásokba és infrastruktúrákba. Ezen előnyök miatt, és nyílt szabványrendszerének köszönhetően a WebGL mára a web 3D animációs kompozíciófejlesztés egyik alappillére lett.

A WebGL animációs eszközkészlete már megfelelően nagy rugalmasságot biztosít a tervezők számára. Az egyszerű, lineáris mozgásoktól kezdve, bonyolult, dinamikus mozgásokig és effektekig elérhető a szoftveres támogatás. A valós idejű renderelés és az interaktív grafikák lehetővé teszik az animációk könnyű módosítását és testreszabását; a fejlesztők azonnali visszajelzést kapjanak a jelenet állapotáról, megkönnyítve az eddig nehezebben integrálható interakciós eszközöket. Az animációk egyaránt futtathatók különböző digitális felületeken és operációs rendszereken, továbbá mobilböngészőkben, vagy

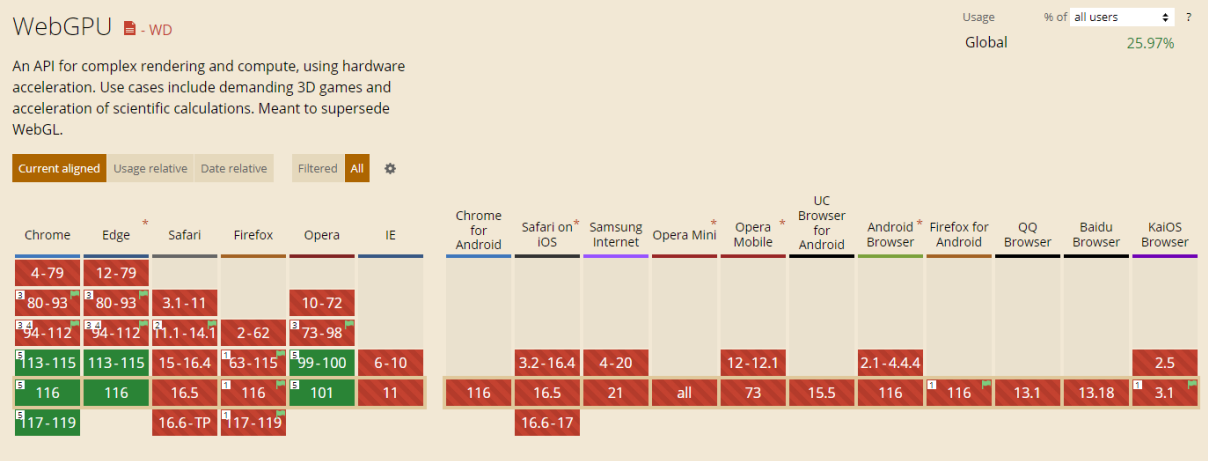
[113] Brandon Jones, Donovan Hutchence, Jaume Sánchez, Takahiro Aoyagi: WebGL and WebGPU Updates. *Khronos,* 2022. tél (https://www.khronos.org/developers/linkto/webgl-webgpu-updates-january-2022) Utolsó letöltés: 2022.03.14.
[114] Google LLC: WebGPU API Specification. Chrome Platform Status. *Google*, 2022. (https://www.chromestatus.com/feature/5682474635577344) Utolsó letöltés: 2023.03.04

applikációként integrálva. Számos előnye mellett a fejlesztések azonban nem tekinthetők kiforrottnak.

Bár a *Khronos Group* által fejlesztett szabvánnyal a technológia képes kellően összetett vizuális élményt nyújtani, a valósághű térábrázolás még távolról sem valósulhatott meg ezen az architektúrán. Számos megkötés mellett, rengeteg optimalizációs eljárással lehet kizárólag elfogadható minőségű, a mai 3D animációs elvárásoknak kisebb részben megfelelő alkotást létrehozni. Az utat a valóban jó minőségű grafikai megjelenéshez a WebGPU előretörése nyitja meg.

4.3. Térhódítás alatt a WebGPU, a jövő web 3D eszköze

Az új technológiát a W3C WebGPU Web Community Group munkacsoport hozta létre azzal a céllal, hogy olyan alacsony szintű *(vagy gépközeli)* API-t biztosítson a kreatív fejlesztőknek, mely leküzdi a WebGL korlátait és hiányosságait.[115] A WebGPU az elérhető legmodernebb 3D webes technológia [116], melynek fejlesztését éppen jelen disszertáció befejezésekor publikálták[117], így böngészőtámogatottsága még elég alacsony.



[WebGPU támogatás eltérő böngészőkben. Caniuse.com, 2023.nyár (https://caniuse.com/?search=webgpu) Utolsó letöltés: 2023.08.31]

[115] François Beaufort & Corentin Wallez: Access modern GPU features with WebGPU. *Web.dev*, 2023. (https://web.dev/gpu/) Utolsó letöltés: 2023.03.07.

[116] Ninomiya, Kai - Wallez, Corentin - Malyshau, Dzmitry: WebGPU Explainer. Draft Community Group Report. *GPU for the Web Community Group*. 2023. tavasz (https://gpuweb.github.io/gpuweb/explainer/). Utolsó letöltés: 2023.04.04

[117] Beaufort François - Wallez Corentin: Chrome ships WebGPU. Google Developer, 2023. tavasz (https://developer.chrome.com/blog/webgpu-release/) Utolsó letöltés: 2023.04.06.

A WebGPU az eddigieknél nagyobb teljesítményű, és a magas minőségű, komplex 3D grafikai képmegjelenítést teszt lehetővé.[118] A párhuzamos adatszámítást is biztosító új technológia egy olyan webes API, amely a modern számítógépes képszámítási képességeket *(lásd DirectX 12, Metal, and Vulkan)*[119, 120] biztosítja a grafikus feldolgozóegységen keresztül. A WebGPU a modern grafikus hardverek képességeit jobban kihasználó, párhuzamos számításokra és erőforrás-kezelésre optimalizált architektúrával rendelkezik.[121]

Egy 3D animátor számára ez a technológia teljesen új platformot jelent, mely a dinamikus térelemek létrehozását biztosítja böngészőkörnyezetben. Támogatja a képi utómunka folyamatokat *(post-processing)*, a dinamikus fényelést- és árnyékolást, a bonyolultabb részecskerendszereket és szimulációs eljárásokat. Segítségével megvalósíthatók az alakábrázoláshoz szükséges morfolások, a rig- és csontvázrendszerek is. Azokat a 3D alkotói rutinokat segíti, melyek nélkül a tervezői státuszból konvertált online variánsokat nem lehetne minőségveszteség nélkül reaktív környezetbe illeszteni.

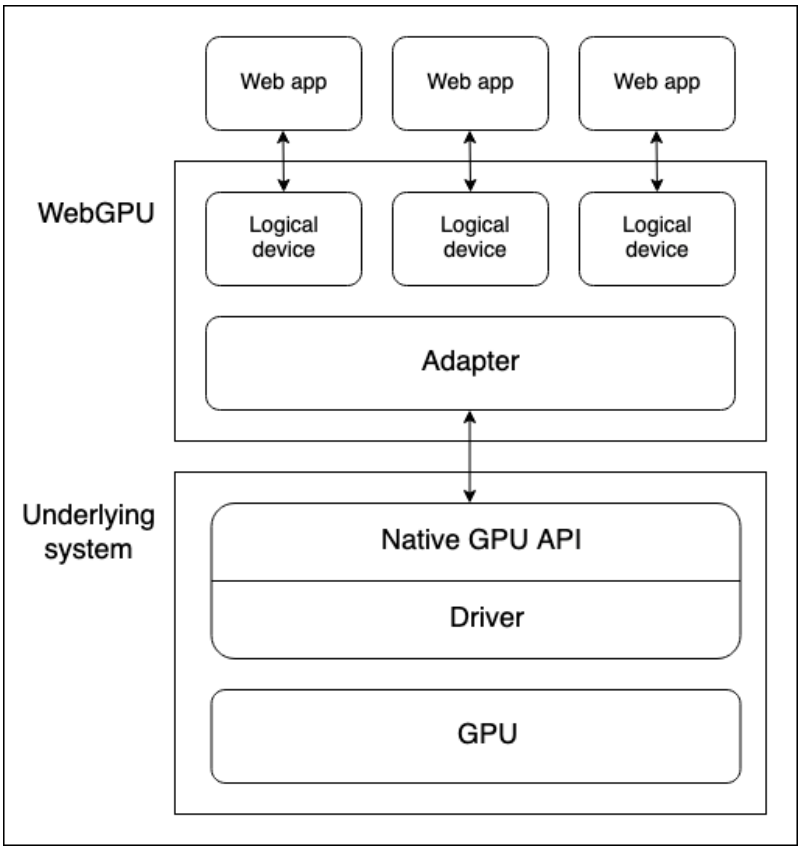
Ricardo Cabello, a világ legelterjedteb online 3D könyvtárának alapítója így nyilatkozott az új eszközöktől.

„Tíz évvel azután, hogy a WebGL lehetővé tette a 3D grafikák online publikusát a weben és megannyi új kísérleti lehetőséget biztosított, eljött végre az ideje a teljes infrastruktúrafrissítésnek, és annak, hogy a fejlesztők teljes mértékben ki tudják használni a modern grafikus kártyákban rejlő lehetőségeket. A WebGPU épp időben érkezik!”[122] A Babylon.js alapítója, David Cathue pedig így nyilatkozott: „A WebGPU közelebb juttat minket a hardverhez, és lehetővé teszi a számítási shader erejének feloldását a fejlesztők számára”.[123]

---

[118] Galvan, Alain: Raw WebGPU. *Github*, 2023. tél (https://alain.xyz/blog/raw-webgpu) Utolsó letöltés: 2023.01.02.
[119] Dzmitry Malyshau: A Taste of WebGPU in Firefox. Mozilla.org, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.
[120] Gullen, Ashley. A brief history of graphics on the web and WebGPU. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/brief-history-graphics-web-1517) Utolsó letöltés: 2022-07-01.
[121] Munro, I. (2018). WebGPU - A New Graphics API for the Web. (https://dev.to/azure/webgpu-a-new-graphics-api-for-the-web-25om) Utolsó letöltés: 2022.04.04.
[122] Google LLC: WebGPU API Specification. Chrome Platform Status. *Google*, 2022. (https://developer.chrome.com/docs/web-platform/webgpu/) Utolsó letöltés: 2023.03.04
[123] Uo.

A WebGPU fejlesztését emellett a vezető innovációs cégek is kivétel nélkül támogatják. *A Google Dawn* nevű platformja kifejezetten a WebGPU bevezetéséhez biztosít optimalizált megoldásokat a programozók számára. A Dawn API C ++ nyelven íródott, és majdnem egy az egyben leképezi a WebGPU IDL-t *(Interface Definition Language),* mely így könnyen használható más rendszerekkel, például webböngészőkkel integrálva.[124] A Dawn tehát a fejlesztők számára biztosítja, hogy WebGPU szabványon alapuló kódot írhatnak, amely így már futtatható a böngészőben és natív alkalmazásként is. Az új vizualizációs API-k a WebGPU megújuló architektúrájára építkezhetnek.



[WebGPU API: A WebGPU API-t futtató webböngésző és az eszköz GPU-ja között több absztrakciós réteg található. MDN Web Docs, 2023.nyár (https://developer.mozilla.org/en-US/docs/Web/API/WebGPU_API) Utolsó letöltés: 2023.08.31]

A WebGPU a GPU hardvereket adapterekként kezeli. [125] *[Fig.WebGPUAbstraction]* Az adapterek a saját memóriával rendelkező GPUDevice-on keresztül biztosítanak kapcsolatot,

[124] Google LLC: Dawn, a WebGPU implementation. *Googlesource*, 2020. (https://dawn.googlesource.com/dawn/+/refs/heads/chromium-gpu-experimental/README.md) Utolsó letöltés: 2022.12.10.

[125] Bynens, Mathias: WebGPU — All of the cores, none of the canvas. *Surma.dev*, 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.

amelyek felelősek olyan erőforrások létrehozásáért, mint a textúrák és a pufferek.[126] A GPUCommandEncoder eszköz lehetővé a renderelési és számítási folyamatok kódolását. Ezeket a parancsokat a GPUCommandBuffer objektum fogadja be. A GPUQueue a parancsok végrehajtásáért felelős, és gondoskodik a megfelelő sorrendről. A GPUBuffer és a GPUTexture olyan fizikai erőforrások, amelyek a GPU memóriáját használják.[127] A GPUCommandBuffer és a GPURenderBundle tárolják a felhasználó által rögzített parancsokat. A GPUShaderModule a shading-gel kapcsolatos információkat és kódokat tartalmazza. A GPUSampler és a GPUBindGroup a fizikai erőforrások használatának beállításait határozzák meg.[128] A GPURenderPipeline és a GPUComputePipeline objektumok felelősek a vezérlési struktúra nagy részéért[129]. Segítenek optimalizálni a folyamatot és irányítani a grafikus erőforrások felhasználását. [130, 131] A feldolgozott és renderelt eredményeket végül a böngésző vászon elemén jeleníthetjük meg, vagy akár több HTML5 Canvas elemen is egyszerre. Az egész eljárássorozat a hatékony és gyors grafikus renderelést biztosítja a böngészőben, amely így hozzájárul a komplex 3D animációk és vizualizációk megjelenítéséhez.

A WebGPU technológia összességében új lehetőségeket nyit a 3D animátorok és grafikus tervezők számára, akik immár a böngészőben is kiváló minőségű és teljesítményű grafikákat hozhatnak létre. Ez a paradigmaváltás a webes grafikus API-k terén lehetővé teszi a nézők, és a felhasználók számára, hogy a natív grafikus API-k által nyújtott előnyöket megtapasztalják; vagyis a gyorsabb képmegjelenítést, a magasabb minőségű térbeli alkotásokat, és a valósidejű interakciót. Az Apple, a Mozilla, a Microsoft, és a Google mérnökei által létrehozott eszközrendszer jelentős változást hoz a broadcast tervezői folyamatokban is. Egyre több élő adásgrafikai vizualizációs szoftver fogja alkalmazni a fenti

[126] W3C Community Group: GPU for the Web. WebGPU API Specification. *W3C*, 2021. (https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2023.01.25.
[127] Dzmitry Malyshau: A Taste of WebGPU in Firefox. *Mozilla.org*, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.
[128] W3C Community Group: GPU for the Web. WebGPU API Specification. *W3C*, 2021. (https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2023.01.25.
[129] Bynens, Mathias: WebGPU — All of the cores, none of the canvas. *Surma.dev*, 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.
[130] Dzmitry Malyshau: A Taste of WebGPU in Firefox. *Mozilla.org*, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.
[131] W3C Community Group: GPU for the Web. WebGPU API Specification. *W3C*, 2021. (https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2023.01.25.

metódusokat, így a televíziós, vagy adás-streaming környezetben dolgozó fejlesztők és művészek is könnyebben integrálhatnak real-time 3D tartalmakat műsoraik számára.

4.4. A WebGL és WebGPU technológiai összehasonlítása

A WebGL és WebGPU közötti fő különbségek az architektúrában, a shader nyelvben és a teljesítményben érzékelhetők.[132] Míg a WebGL az OpenGL ES-en alapul és GLSL shader nyelvet használ, a WebGPU teljesen új API-t definiál, mely a WGSL *(WebGPU Shading Language)* shader nyelvre támaszkodik. A WGSL shader nyelv a WebGL által használt GLSL shader nyelvnél modernebb, újabb funkciókat és lehetőségeket nyújt a programozóknak.[133] A WebGL inicializálásakor a felhasználó létrehoz egy shader programot. Amikor a megpróbálja felhasználni ezt a shader-t, a vezérlő figyelembe veszi az egész állapotstruktúrát, és változás esetén újra rendereli a teljes shader programot. A WebGPU ezzel ellentétben a felhasználó által előre definiált pipeline állapotokat képes külön objektumként kezelni (*GPURenderPipeline és GPUComputePipeline)*. Ez jelentősen optimalizálja az időigényes folyamatokat.[134]

A WebGL a jellemzője a viszonylag alacsonyabb teljesítmény, és az alacsony szintű erőforrás-kezelés hiánya. Ezzel ellentétben a WebGPU új technológiai megközelítést jelent, amely a jövő webes grafikai alkalmazásainak alapjául szolgálhat.[135] A web shader nyelve lefordítható SPIR-V-re is, ami egy eszközkompatibilitásra használt köztes platformfüggetlen nyelv. Az programozók lényegesen több beállítási lehetőségeket érhetnek el az erőforrások hatékonyabb kihasználása érdekében *(Bindig model: GPUBindGroup, GPUBindGroupLayout)*. A csoportos beállításopciók egyetlen funkcióhívással teszik a

[132] Gullen, Ashley: A brief history of graphics on the web and WebGPU. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/brief-history-graphics-web-1517) Utolsó letöltés: 2022-07-01.
[133] Seguin Damien: Graphics on the Web and Beyond with WebGPU. *Medium*, 2020. nyár(https://dmnsgn.medium.com/graphics-on-the-web-and-beyond-with-webgpu-13c4ba049039) Utolsó letöltés: 2023.04.02.
[134] Gullen, Ashley: From WebGL to WebGPU in construct. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/webgl-webgpu-construct-1519) Utolsó letöltés: 2022-07-03.
[135] Brandon Jones, Donovan Hutchence, Jaume Sánchez, Takahiro Aoyagi: WebGL and WebGPU Updates. *Khronos*, 2022. tél (https://www.khronos.org/developers/linkto/webgl-webgpu-updates-january-2022) Utolsó letöltés: 2022.03.14.

WebGL-nél lényegesen gyorsabbá a munkát. A WebGPU a nagyobb teljesítményű vizualizációs eszközöket párhuzamos kép-, ill. adatszámítással támogatja. [136]

Elemzésem során arra következtetésre jutottam, hogy a WebGPU a jövőbeli online ábrázolástechnikai módszerek megkerülhetetlen forrása lesz. Hosszútávon pedig a modernebb rendszer kétségkívül kiváltja majd a WebGL technológiát, így minden webes 3D absztrakciós könyvtár is adaptálni fogja alkalmazáskörnyezeteibe. Az artistok és animációs szakemberek számára ez nagyobb poligonszámú térábrázolási lehetőséget, magasabb FPS számú animációs képességet, lényegesen összetettebb jelenetábrázolást és képmegjelenítést, a néző számára pedig élvezhetőbb képminőséget jelent.

4.5. A programozási ismeretek szerepe a Web 3D tervezésben

Összefoglalva elmondható, hogy a hagyományos 3D modellező és animációs eszközök, a JavaScript alapú keretrendszerek és könyvtárak, a speciális kimeneti formátumok és az eltérő online platformok együttesen képezik a Web 3D technológia alapját. A hatékony vizualizációs munkafolyamat kialakításához elengedhetetlen a programozási ismeretek elsajátítása is. Folyamatosan tanulmányozni kell a leíró és szkriptnyelvi szintaktikát, integrálni a kódoláshoz szükséges szoftvereket, kialakítani a programozási rutinokat és feltérképezni a 3D JavaScript-hez kapcsolódó kiegészítő módszereket. Emellett, hangsúlyozni kell, hogy számos felhőalapú, fizetős szolgáltatás érhető el *(például Clara, Cesium, Sketchfab, p3d, PlayCanvas, Verge3D)*, amelyek lehetővé teszik a digitális látványvilágok publikálását és szerkesztését. Bár az online szolgáltatások megfelelő kezelőfelületet biztosítanak a térhatású elemek szerkesztéséhez, korlátozottak az egyéni testreszabhatóság és a szabad felhasználhatóság tekintetében. Korlátozó jellegű tulajdonságaik miatt, a teljesen testre szabható vizualizációs applikáció még nem áll a broadcast 3D tervezők rendelkezésére.

Ezen érvek alapján nyilvánvaló, hogy televíziós környezetben, az online 3D térbeli elemek létrehozásához egy teljesen testre szabható, egyedi web 3D fejlesztői környezet kialakítása indokolt. A következő fejezet ezen eljárásegyüttes részletes ismertetésével foglalkozik.

[136] Dzmitry Malyshau: A Taste of WebGPU in Firefox. *Mozilla.org*, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.

5. Web 3D Full-Stack fejlesztői környezet - WebGL és WebGPU alapú 3D animációs és vizualizációs alkotások környezetének definiálása

Ebben a fejezetben ismertetem, hogy a web 3D alkalmazások létrehozásához pontosan milyen fejlesztői környezet kialakítása szükséges. Feltárom azokat az ismeretanyagokat, amelyek elengedhetetlenek a hipotézisek megválaszolásához, egyben elősegítik a DLA 3D vizualizációs mestermunka sikeres megvalósítását.

5.1. Architekturális és technológiai összefüggések a 3D webfejlesztési stackben

A web 3D rendszertervezés és fejlesztés komplex diszciplínájában az integrált fejlesztői eszközök és környezetek kiemelkedő jelentőséggel bírnak az ökoszisztéma koherenciájának fenntartásában. Ebből következik, hogy a teljes rendszerkörnyezet definiálása és ismertetése szükséges, mely tartalmazza a frontend és backend architektúrát, a kliens- és szerveroldali logikát, az adatbázis-kezelést, és a felhasználói interakciókat. Ez a fejezet egy átfogó képet nyújt a teljes fejlesztői stackről, amely a komplex web 3D alkalmazások létrehozásához szükséges.

5.1.1. Alkalmazásszintű technológiák rendszerben elfoglalt helye és szerepe

Ebben a bekezdésben a 3D webfejlesztési stack multifunkcionális komponenseinek analízisét végzem. Az elemzés fókusza a különböző technológiai rétegek közötti funkcionális szinergiák feltárása, valamint a komplex függőségi viszonyok és kapcsolódási pontok strukturált körvonalazása. A következőkben a fejlesztési komponensek rendszerben elfoglalt helyét és szerepét definiálom.

Front-End technológiák

A webes kliens-szerver architektúra kliensoldali rétegét a *HTML5, Cascading Style Sheets (CSS3)*, és *ECMAScript* szabványok alapján fejlesztik. A *HTML5* nem csak strukturális keretet biztosít a webes tartalomnak, de modern szemantikus elemeket és API-kat is kínál, amelyek segítségével komplex 3D alkalmazások építhetők. A *CSS3*, mint stílusleíró nyelv, olyan fejlett vizualizációs elemek megtervezését teszi lehetővé, mint átmenetek, animációk vagy flexibilis elrendezési modellek.

Az *ECMAScript* szabványaira épülve a *JavaScript* dinamikus interakciók, állapotkezelések és aszinkron adatműveletek lehetőségét kínálja. A *TypeScript,* mint statikus

típusrendszerrel kiegészített JavaScript, fokozott biztonságot és robusztusságot kínál a fejlesztési folyamat során. Specifikusan a 3D grafikai megjelenítés WebGL és WebGPU API-kon keresztül valósul meg. Ezek az API-k -*a korábban ismertetett módon*- közvetlen hozzáférést biztosítanak a grafikus hardver erőforrásaihoz, lehetővé téve komplex 3D modellek, textúrák és shader algoritmusok hatékony megjelenítését.

Back-End Technológiák

A szerveroldali logika implementációjára a Node.js platformot alkalmazom. Ezáltal biztosított az erőforrások hatékony kihasználása, valamint az alkalmazás magas rendelkezésre állása. Az *Express.js*, egy *Node.js* keretrendszer, mely olyan funkcionalitást biztosít, mint a HTTP útvonalak és kéréskezelés egyszerűsítése, valamint middleware komponensek modularizált integrációja. Adatbázis interakciók objektum-relációs leképezők segítségével valósulnak meg *(Sequelize)*, amelyek absztrahálnak a natív SQL query-k komplexitásától és biztonsági kockázataitól. Ezzel lehetővé válik a modellek és az adatkapcsolatok programatikus kezelése az alkalmazásban, valamint a CRUD műveletek is egységesen definiálhatók és karbantarthatók.

5.1.2. Intermediális technológiák

Adatbázis management rendszer

Az adatbázis kezelés a 3D webalkalmazás egyik legkritikusabb aspektusa, amely közvetlen hatással van a rendszer teljesítményére és skálázhatóságára. A MySQL adatbázis-kezelő rendszer a 3D webalkalmazásom egyik alapvető pillére, amely támogatja a komplex adatkapcsolatok kezelését, a 3D modellekhez kapcsolódó metaadatok tárolása jól tervezett adatsémákban történik. A normalizáció elvének követése lehetővé teszi az adatok hatékony és redundancia nélküli tárolását a MySQL adatbázisban.

API és web szolgáltatások

A kliens és szerver közötti adatáramlást RESTful *vagy GraphQL* alapú API segíti, amelyek lehetővé teszik a komplex adatszerkezetek hatékony lekérdezését és manipulációját.

5.1.3. Infrastrukturális technológiák

Saját Express alapú szerver

A 3D alkalmazás egy saját fejlesztésű, Express alapú szerveren fut, melynek logikája egy *server.tsx* fájlban van implementálva. Ez a konfiguráció lehetővé teszi az útválasztási szabályok, az adatbázis-interakciók és az egyéb köztes szoftverek kontrolját és menedzsmentjét. A Node.js a szerveroldali JavaScript futtatását, az Express keretrendszer pedig a hozzáadott kényelmi réteget és struktúrát biztosítja.

Fontos megjegyezni, hogy bár jelenleg saját fejlesztésű Express alapú szerveren fut az alkalmazás, más hosting lehetőségek, mint például az AWS vagy az Azure, is kiválóan alkalmasak lehetnek a jövőbeni skálázásra vagy redundancia igények esetén.

5.1.4. Fejlesztői eszközök és fejlesztői környezet

A *Git* verziókezelő rendszer a forráskód hatékony menedzsmentjéért felel, emellett alapvető szerepet játszik a 3D modellek, textúrák és hozzájuk tartozó metaadatok konzisztens tárolásában is. Ebben a kontextusban az integrált fejlesztői környezetek, mint például a *Visual Studio Code*, különféle specializált bővítmények és modulok segítségével komplex támogatást nyújtanak a 3D fejlesztési ciklus minden lépéséhez. Ezzel párhuzamosan, a *build* és automatizálási eszközök, például a *Webpack*, automatizálják a transzpilációs folyamatokat és a függőségek menedzsmentjét.

A következő szakaszban elemzést végezek a kliensoldali technológiai stack kiválasztásának stratégiai jelentőségéről, különös tekintettel a 3D fejlesztői környezet specifikus kihívásaira és lehetőségeire. Az értékelés a megvalósítás elméleti lépéseinek ismertetésével történik.

5.1.5. Hardveres feltételek és követelmények

A fejezet áttekintést nyújt az információs architektúra és szoftvertervezés hardveres dimenzióiról, különös tekintettel azokról az erőforrásokról, amelyek elengedhetetlenek a háromdimenziós webalkalmazások stabil működéséhez, továbbá működtetéséhez. Összegzem a hardverkomponensek kritikus szerepét, és bemutatom a minimális rendszerkövetelményeket, az alapvető hardveroptimalizálási paradigmákat és megvalósítási stratégiákat.

Grafikus Feldolgozó Egység

A Grafikus Feldolgozó Egység *(GPU)* kiemelt jelentőséggel bír a webes 3D grafikai vizualizáció területén, hiszen a grafikai feldolgozást ez az eszköz végzi. A WebGL és WebGPU technológiák modern GPU-ken futnak, és lehetőséget biztosítanak a grafikai erőforrások hatékony kihasználására. WebGL alapú megoldások esetében a GPU jellemzően a textúrázás, árnyékolás és komplex geometriai számítások gyors végrehajtásáért felelős. A WebGPU, mint az újabb technológia, még több funkciót és jobb teljesítményt kínál. Kifejezetten a párhuzamos feldolgozásra és a hardveres gyorsításra lett tervezve, így lehetővé teszi az erőforrás-intenzív grafikai műveletek még gyorsabb végrehajtását. Ezek a technológiák lehetővé teszik a valósidejű grafikai műveletek végrehajtását a böngészőben, minimalizálva ezzel a CPU terhelését. A GPU tehát nem csak a vizualizáció gyorsításában játszik szerepet, hanem abban is, hogy a CPU erőforrásait más, kritikusabb feladatokra lehessen fordítani. A modern GPU-architektúrák továbbá speciális funkciókat is kínálnak, mint például a *ray tracing* vagy a *tensor műveletek*, amelyek új dimenziót nyitnak a 3D webvizualizációban. Az alkalmazások így nemcsak gyorsabbak és szebbek lehetnek, de olyan komplex vizualizációs algoritmusok is megvalósíthatóak, amelyek eddig csak dedikált szoftverekben vagy *high-end* grafikus állomásokon voltak lehetségesek. Mivel a WebGL és WebGPU fejlesztési folyamatai szorosan kötődnek a hardverek képességeihez, a megfelelő hardverválasztás kulcsfontosságú. Az új generációs GPU-k gyakran rendelkeznek olyan speciális funkciókkal, amelyek tovább növelik a 3D webalkalmazások teljesítményét és megbízhatóságát, így a hardver és a szoftver közötti szoros integráció érdekében érdemes naprakészen tartani a technológiai ismereteket.

A webes 3D grafikai vizualizáció során használt technológiák, mint a WebGL és WebGPU, kiemelten támaszkodnak *a Grafikus Feldolgozó Egység (GPU)* teljesítményére, ugyanakkor a *Központi Feldolgozó Egység (CPU)* szerepe sem elhanyagolható, és bizonyos esetekben kritikus jelentőségű lehet.

Központi Feldolgozó Egység

A *CPU* az alkalmazás alapvető logikai és aritmetikai műveleteiért felel, és a többszálas feldolgozás hatékonyságára is jelentős hatással van. Az újabb generációs processzorok általában elegendő teljesítményt nyújtanak, de az alkalmazás komplexitásától függően specifikusabb hardveres követelmények is felmerülhetnek. A modern többmagos

CPU-architektúrák előnye kettős. Egyrészt lehetővé teszik a párhuzamos feldolgozást, ami kritikus lehet komplex geometriai számítások vagy fizikai szimulációk esetében. Másrészt az alkalmazás backend algoritmusai is ezen futnak, melyek meghatározzák az alkalmazás logikáját és dinamikáját. Így a CPU nem csak az általános teljesítmény növelésében, hanem a valósidejű, interaktív vizualizáció estén is kritikus szerepet játszik.

Memória erőforrások

A *memória erőforrások* is fontos szerepet játszanak a webes 3D vizualizációs alkalmazások teljesítményében és stabilitásában. A CPU és GPU a számítások gyors végrehajtásáért felel, míg a memória *(RAM és VRAM)* az adatok ideiglenes tárolásáért és gyors hozzáféréséért.

A rendszer memóriája *(RAM)* kritikus az alkalmazás alapvető műveletei szempontjából, beleértve a szövegek, képek, 3D-s geometriák és egyéb adatok kezelését, fájlok betöltését és a többszálas műveletek közötti adatcserét. A rendszermemória kapacitásának hiánya teljesítményproblémákhoz vezethet, nagy adatmennyiségek vagy komplex geometriai modellek kezelése esetén. A grafikus memória, más néven VRAM, amelyet a GPU használ, kifejezetten a textúrák, *shader-ek* és egyéb grafikai adatok tárolására szolgál. A VRAM gyors hozzáférést biztosít a grafikai erőforrásokhoz, ami elengedhetetlen a valósidejű 3D vizualizációknál. Kevés VRAM esetén a GPU-nak a lassabb rendszermemóriához kell nyúlnia, ami jelentős teljesítményvesztéssel járhat. A memória erőforrások optimalizálása különösen fontos lehet olyan eszközökön, amelyek korlátozott memóriaerőforrásokkal rendelkeznek, mint például a mobil eszközök. Memória-optimalizálási technikák, például adatkompresszió vagy *lazy loading*, alkalmazhatók a rendelkezésre álló erőforrások hatékonyabb kihasználására. A megfelelő memória-gazdálkodás nem csak a teljesítményt növeli, de az alkalmazás stabilitását és reszponzivitását is jelentősen javítja.

A háromdimenziós webalkalmazások teljesítménye további hardverkomponenstől és konfigurációs tényezőtől is függ.

A tárolóeszközök I/O teljesítménye, mint például a Solid State Drive (SSD) vagy az M.2 NVMe, alapvetően befolyásolja textúrák és shader programok gyors és hatékony betöltését. A rendszermemória korlátozottsága esetén a merevlemez használata a feldolgozási sebesség jelentős csökkenéséhez vezethet.

A hálózati kapcsolat minősége szintén befolyásolhatja az alkalmazás teljesítményét. Egy magas sávszélességű és alacsony késleltetésű kapcsolat csökkentheti az online erőforrások betöltési idejét, így növelve az alkalmazás elérhetőségét, teljesítményét és reszponzivitását. A hálózati kapcsolat minősége más szempontokból is befolyásolhatja az alkalmazás teljesítményét, például a felhasználói interakciók sebességét vagy a hálózati késleltetést.

A CPU és GPU közötti adatátvitel sebességét a PCIe busz sebessége is meghatározza, amelynek korlátozott sávszélessége a komponensek közötti kommunikáció sebességének csökkenését eredményezheti. A PCIe busz sebessége más szempontokból is befolyásolhatja az alkalmazás teljesítményét, például a GPU-nak a CPU-tól kapott adatok feldolgozási sebességét.

A hűtési rendszer szerepe rendkívül fontos szempont a háromdimenziós webalkalmazások stabil és optimális működése érdekében. Egy nem megfelelő hűtési rendszer túlmelegedéshez, vagy kritikus termikus állapot kialakulásához vezethet. A hűtési rendszer hibája nem csak a teljesítmény csökkenéséhez, hanem a hardver károsodásához is vezethet.

Az operációs rendszer és a megfelelően konfigurált illesztőprogramok is kulcsfontosságúak a WebGL és WebGPU megfelelő teljesítményéhez. Ebben az összefüggésben, az operációs rendszer paraméterei, így a hardvergyorsítási opciók kiemelten fontos tényezők lehetnek. Emellett a böngésző legfrissebb verziójának használata is kritikus jelentőségű, mert a frissítések gyakran tartalmaznak teljesítményoptimalizációs eljárásokat és biztonsági javításokat, amelyek közvetlen hatással lehetnek a WebGL és WebGPU teljesítményére. Összességében az erőforrások megfelelő menedzsmentje és optimalizálása az alkalmazás minőségét és a felhasználói élményt is garantálhatja.

A következő fejezet a fejlesztési stratégiák kliensoldali eszközkészletének definiálást célozza.

5.2. Front-End fejlesztési stratégiák és kliensoldali eszközök meghatározása. A modern keretrendszerek és applikációk szerepe a 3D fejlesztői környezet kialakításában.

A fejlesztői környezet definiálása már egy HTML5 oldalon, *vagy* SPA *(Single Page* App) egyoldalas alkalmazáson is lehetséges [137]. A modern webes Front-End keretrendszerek *(React, AngularJS, Vue)* preset-jei is lényegesen leegyszerűsítik a környezet kialakítását.[138] A React.js applikációk közül a Create React App *(CRA)* kiválóan megfeleltethető a grafika-szemléletű modellezés megkezdéséhez[139], és a 3D animációs fókuszú vizualizációs munkakörnyezet kialakításához. A telepítéshez szükséges a JavaScript alapú szerverkörnyezet: a Node.js[140] installálása[141] is. A typescript-template setup VSCode CLI-n keresztül egyetlen sor kóddal telepíthető Yarn-nal[142], npm-el, vagy futtatható npx-el. Az összeállítás rendkívül egyszerű használatot biztosít; minimalizálja a fordító- és modulcsomagok konfigurálás- és telepítésigényét, nem szükséges a beépülő eszközök *(Webpack, Babel [143, 144])* előzetes *(vagy manuális)* paraméterezése sem. A kialakított, platformfüggetlen környezet adaptálható web, WebVR [145, 146] *(pl. React 360[147]),* broadcast

[137] Simpson, K.: Understanding React: A Guide for Frontend Developers. *Envato Tuts+*, 2021. (https://webdesign.tutsplus.com/series/understanding-react--cms-1519) Utolsó letöltés: 2023.02.12.
[138] Freeman, A.: AngularJS Directives Fundamentals. *Pluralsight* 2021. tél (https://app.pluralsight.com/library/courses/angularjs-directives-fundamentals/table-of-contents) Utolsó letöltés: 2021.04.10.
[139] ReactJS.org: Create React App. *Facebook*, 2021. (https://create-react-app.dev/docs/getting-started/) Utolsó letöltés: 2023.02.12.
[140] Node.js: Node.js v16.10.0 documentation. *Node.js*, 2022. (https://nodejs.org/docs/latest-v16.x/api/) Utolsó letöltés: 2022.03.27.
[141] Mardan, A.: Node.js 16: The Complete Guide. *Udemy*, 2022. (https://www.udemy.com/course/nodejs-the-complete-guide/) Utolsó letöltés: 2023.11.01.
[142] Yarn: Yarn Package Manager. *Yarn documentation*, 2020. (https://classic.yarnpkg.com/en/docs/) Utolsó letöltés: 2022.11.24.
[143] Kódnyelvi fordító-eszközök. A modul bundlerek különböző állományokból (JS, SCSS, TS, stb. statikus állományokat hoznak létre.)
[144] Juho Vepsäläinen: SurviveJS – Webpack 5. *SurviveJS*, 2023. (https://survivejs.com/webpack/) Utolsó letöltés: 2023.04.03.
[145] Mozilla Developer Network (MDN): Mozilla VR: Getting Started with WebVR. *Mozilla Corporation*, 2022. (https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API/Using_the_WebVR_API) Utolsó letöltés: 2022.10.21.
[146] Diego Marcos, Don McCurdy, Kevin Ngo: A-Frame: A web framework for building virtual reality experiences. Aframe.io, 2023. (https://aframe.io/) Utolsó letöltés: 2022.10.27.
[147] Facebook Inc.: React 360 documentation. *GitHub*, 2022. tavasz (https://facebook.github.io/react-360/) Utolsó letöltés: 2022.09.03.

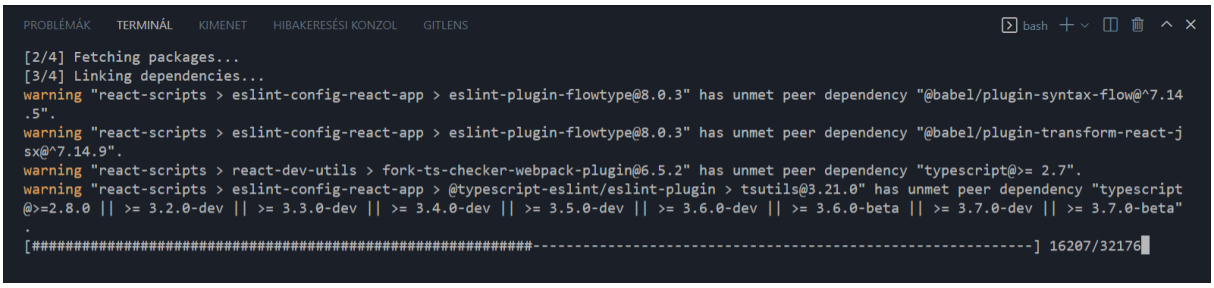
rendszerek és az eltérő virtuális színterek *(XR)* irányába is[148]. A telepítés- és a környezet inicializálása parancsos utasításokkal kezdeményezhető.

5.2.1. Telepítés és inicializálás, csomag- és függőségkezelés

A Yarn optimalizált csomag- és függőségkezelő rendszer, amely rövid parancssori utasítások segítségével támogatja a komplex telepítési eljárásokat, a kódcsomagok kezelését: megosztását, frissítését, cseréjét. Integrálásával egyidőben kialakítható a CRA és TypeScript[149] fejlesztői környezet[150].

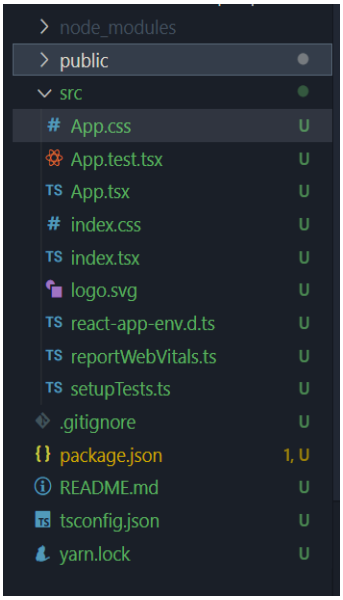
```
yarn create react-app . --template typescript
npx create-react-app . --template typescript

yarn add @types/node
```

[Yarn parancssori utasítás.]



[Yarn telepítés VsCode terminállal. Forrás: Balogh Áron, 2023.]

[148] W3C: WebXR Device API. *Github*, 2022. (https://immersive-web.github.io/webxr/) Utolsó letöltés: 2023.01.20.
[149] Basarat Ali Syed: TypeScript Deep Dive. *GitBook*, 2022. tél (https://basarat.gitbook.io/typescript/) Utolsó letöltés: 2022.01.14.
[150] Microsoft: TypeScript documentation. *Microsoft*, 2022. (https://www.typescriptlang.org/docs/) Utolsó letöltés: 2023.03.20.



[Könyvtárszerkezet a telepítése után. Forrás: Balogh Áron, 2023.]

A *(node_modules mappába)* telepített modulokat, vagy programcsomag-függőségeket a package.json fájl dokumentálja. A leírófájl egyszerűen áttekinthető az aktuális alkalmazás-architektúra, és megvalósítható a beépülő komponensek verziókezelése, az alkalmazás későbbi újratelepítése, bővítése. [151] *[Fig.Package]*

5.2.2. A TypeScript programozási nyelv szerepe a kliensoldali fejlesztési modellben. Az ECMAScript és a JavaScript kapcsolata.

A kialakított futtatási környezet alapja a TypeScript programozási nyelv, amely az ECMAScript-szabványrendszere épülő JavaScript statikusan típusos bővítése. Működésének megértéséhez az ECMAScript[152] és a JavaScript kapcsolatát is értelmezni szükséges.

5.2.2.1. ECMAScript – JavaScript

Az ECMAScript általános nyelvi szabványrendszere a JavaScript nyelv alapja[153]. A JavaScript (*mint az ECMAScript egy dialektusa)*, olyan általános célú szkriptnyelv, amely az ECMA-262[154] szabványára támaszkodik. Ez határozza meg a JS kódsorok szintaktikájának

[151] Xu, Jack: Practical WebGPU Graphics: Creating Advanced Graphics on the Web Using WebGPU. *Google Books, Apress,* 2022. (https://books.google.hu/books?id=Qwo9zgEACAAJ.) Utolsó letöltés: 2023-04-04.
[152] Ecma International: Language Specification. *ECMAScript*, 2021. (https://www.ecma-international.org/publications/standards/Ecma-262.htm) Utolsó letöltés: 2023.04.03/
[153] Mozilla Developer Network (MDN): ECMAScript. *Mozilla Corporation*, 2023. (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources) Utolsó letöltés: 2023.04.03.
[154] Axel Rauschmayer: Exploring ECMAScript 6: Upgrade to the Next Version of JavaScript. O'Reilly Media, 2015. (https://exploringjs.com/es6/) Utolsó letöltés: 2019.02.14.

számottevő részét. A szabvány fejlesztése 1997-óta folyamatosan zajlik. Az ES6 *(azaz ECMAScript 2015)* rengeteg új funkcióval bővítette a nyelvet, mely azóta is évenkénti verziófrissítéssel azonosítható. A grafikus, vagy front-end fejlesztő felelőssége a megírt kódok aktuális sztenderdekhez igazítása, a nyelvi eljárások dinamikus változásának nyomon követése. Ezeket a modern tulajdonságokat fejlesztési munkafolyamatok optimalizálása miatt, a hatékonyabb és egyszerűbb kódírás érdekében publikálják.

A szkriptek értelmező, végrehajtó programja egy JS *engine*, amely a modern böngészők többségében[155] *(továbbá Node.js-en is)* megtalálható. A prototípus alapú programozási nyelv elemeit *(pl. osztályait és függvényeit)* objektummá alakítja. Ezek az objektumok típustalan[156] adatokat tulajdonságokká, függvényeiket metódusokká alakítják. A forráskód fejlesztési környezete típusosan is inicializálható, TypeScript bővítménnyel.

5.2.2.2. TypeScript

A TypeScript nyílt forráskódú, rendszerfüggetlen, statikusan típusos nyelv[157], amely végeredményként JavaScript -re fordul *(vagyis konvertálódik)*. A típus-definíció, és típusértékelés a szintaktikailag helyes szkriptek előállítását, fordítás-idejű ellenőrzést, továbbá objektumorientált fejlesztést: osztályok, interfészek, modulok használatát támogatja.[158] TS segítségével a 3D grafikai környezet forráskódjának kialakítása, és validálása egyszerűen végrehajtható. A felhasználó felület megvalósításához a React.js könyvtár is használható. *[Fig.TsConfig]*

5.2.3. Komponens alapú fejlesztési modell: React.js

A React.js[159] alkalmazások interaktív felületének tervezését biztosítja. A komponens alapú fejlesztés leegyszerűsíti a komplex kezelőfelületek kódolását, kialakítását. A kisebb részegységekbe szervezett szkript-blokkok újra-renderelése lényegesen megkönnyíti az állapotváltozások kezelését, és a DOM-on -*vagyis a tényleges grafikus felületen*- végbemenő változások vizualizációját a kliens számára. Az újrahasználható kódegységekkel a

[155] Chrome: V8 engine, Firefox: SpiderMonkey, Edge: Chakra
[156] Automatikus adattípus meghatározás (URL: http://www.inf.u-szeged.hu/~tarib/javascript/primitivek.html#tipusszerkezet)
[157] Tari Balázs: Programozás és algoritmizálás JavaScript nyelven. *Educatio Társadalmi Szolgáltató*, 2019 nyár (http://www.inf.u-szeged.hu/~tarib/adatlap.html#tema). Utolsó letöltés: 2023.01.05.1
[158] Xu, Jack: Practical WebGPU Graphics: Creating Advanced Graphics on the Web Using WebGPU. *Google Books, Apress*, 2022. (https://books.google.hu/books?id=Qwo9zgEACAAJ.) Utolsó letöltés: 2023-04-04.
[159] Facebook Inc.: React dokumentáció. *ReactJs*, 2022. (https://reactjs.org/) Utolsó letöltés: 2023.04.03.

fejlesztési idő jelentősen csökkenthető, így interaktív 3D vizualizációs eszköz tervezéséhez megfelelő választás lehet. A külön komponensek JSX, és TSX formátumban is menthetők [160].

5.2.4. A frontend fejlesztés fájltípusainak szerepe és jelentősége a funkciók megtervezésében és létrehozásában

A frontend fejlesztés egyik fontos aspektusa az alkalmazásban használt fájltípusok pontos meghatározása, amelyek kulcsszerepet játszanak az alkalmazás funkcióinak megtervezésében és fejlesztésében. Az adott fájltípusok határozzák meg az alkalmazás használatának felhasználási módját, illetve biztosítják az egyes funkciók megfelelő működését is. Ezen fájltípusok közül kiemelkedő szerepet tölt be a TypeScript *(.ts),* amely a JavaScript (*.js)* nyelvhez képest külső szintaktikai rétegként is értelmezhető. A TypeScript kódját a TypeScript compiler *(.tsc)* konvertálja JavaScript fájlokká a fordítás során. React alapú alkalmazáskörnyezetben a .tsx *(TypeScript)* fájl azt jelöli, hogy az állomány rendelkezik valamilyen DOM visszatérési értékkel. Ennek megfelelően a JSX is olyan kiterjesztéstípus, mely a HTML elemek használatát biztosítja JavaScript-ben.

Az .mjs fájlok a JavaScript moduláris fejlesztésének alappillérei. Ezek teszik lehetővé a kódbázis szétválasztását kisebb fájlokra, és biztosítják ezek egyszerű importálását és exportálását további állományok irányába. A modularizáció során egyszerűbbé, követhetőbbé válik a forráskód szintaktika-ellenőrzési folyamata is.

5.2.5. Kódanalízis bővítmények alkalmazása a fejlesztési folyamatban

A forráskód szintaktikai elemzését az ESLint bővítmény garantálja, a kód formázást pedig a Prettier kiegészítő. *[Fig.Eslint]* Mindkét bővítmény hatékonyabbá és karbantarthatóbbá teszi a forrásállományt.

Az ESLint szintaktikai hibákat és kódszervezési problémákat azonosít a JavaScript kódban. Az egyéni szabályok szerint konfigurálható modul legnagyobb előnye, még a programsorok írása közben jelzi és javítja a hibás részeket.[161] A Prettier VsCode kiegészítő lehetővé teszi a kód formázásának teljes automatizálását, így könnyítve a sokezer karakter moduláris elhelyezéséből adódó esetleges formátumeltéréseket és hibákat.

---

[160] Microsoft Corporation: JSX. *Microsoft* 2021. (https://www.typescriptlang.org/docs/handbook/jsx.html) Utolsó letöltés: 2023.03.23.

[161] OpenJS Foundation: ESLint documentation. *OpenJs*, 2023. (https://eslint.org/docs/user-guide/getting-started). Utolsó letöltés: 2023.04.05.

yarn run eslint --init (@eslint/create-config)
npx eslint --init
illetve:
yarn add --dev @typescript-eslint/parser @typescript-eslint/eslint-plugin eslint typescript

[ESLint telepítés parancssori utasítása..]

5.2.6. CSS preprocesszorok a modern webfejlesztésben: Sass és a stíluslapok modularizációja

A HTML5 felületek megjelenése CSS *(Cascading Style Sheets)* stíluslapokkal formázható. A W3C[162] által szabványosított egyszerű nyelv az egyes DOM elemeket szelektorok és a deklarációs szakaszok segítségével definiálja. CSS leírás a preprocesszorok, azaz előfordítók segítségével gyorsabban készíthető. Az egyedi szintaktika olyan nyelvi kiterjesztéseket tartalmaz, amely gyorsítja a fejlesztés folyamatát, javítja a kód olvashatóságát, a nyelvi elemek későbbi frissítését.

Az Sass *(Syntactically Awesome Style Sheets)* olyan CSS előfeldolgozó-, *ill. fordító-nyelv*, amely lehetővé teszi változók, mixinek[163], függvények használatát CSS-sel megegyező szintaxis-környezetben. Sass szkriptnyelvvel modulárisan strukturálható állományok alakíthatók ki, amely jelentősen egyszerűsíti a formázásállomány olvashatóságát, áttekinthetőségét.

yarn add node-sass
yarn add @types/node-sass

yarn add sass
yarn add @types/sass

npm install @types/sass

[Sass telepítése]

5.2.7. A programkód dokumentálásának jelentősége

A vizuális programozási eljárás szerves része a forráskódok dokumentációjának elkészítése is, mely elengedhetetlen az alkalmazás életiciklusának nyomkövetésében, illetve továbbfejlesztése és karbantartása során. A grafikus felület

[162] W3C: World Wide Web Consortium. *W3.org,* 2023. (https://www.w3.org/) Utolsó letöltés: 2023.04.03.
[163] Sass: Syntactically Awesome Style Sheets. *Sass-lang.com*, 2022. (https://sass-lang.com/) Utolsó letöltés: 2023.01.25.

dokumentálására a JSDoc[164] API-ja használható. A szkriptsorokhoz közvetlenül megjegyzések fűzhetők, megkönnyítve a forráskódok későbbi elemzését, értelmezését.

5.2.8. A Git forráskódkezelő rendszer alkalmazása a fejlesztési munkafolyamatban

A létrehozott saját állományok a Git forráskódkezelő rendszerrel tárolhatók. A végbemenő fejlesztési munka állapotváltozásait, előzmény- vagy helyreállítási információit Git Repository-k őrzik. A repository-k a forrás teljes állományát elosztott rendszereken képesek tárolni. A branch-ek fejlesztési szakaszokra, ágakra bontják a megvalósítás lépéseit. A rendszer, az előrehaladás lépéseit commit-okkal *(verziókkal)* rögzíti, és a *(lokális, vagy távoli)* tárolók szinkronizálásával automatikusan frissíti. Számos telepítési- és futtatási környezetben használt állomány szükségtelen az alkalmazás egyes munkafázisainak verziókezelésekor. Git környezetben létrehozható egy *gitignore* fájl, mely a nem nyomon követett elemek listáját tartalmazza. Broadcast- vagy 3D fejlesztés esetén a webes kiterjesztések médiatípusok szerint is szűrni kell.

5.3. Back-End stack és a szerveroldali architektúra kialakításának stratégiája

A 3D full-stack applikáció MVN minta szerint tervezhető, szerveroldali architektúrája JavaScript eszközökkel definiálható. Adat- vagy modellkezelés rétegét a Node.js és Express.js biztosítja.

5.3.1. Node.js és az Express.js használata a 3D CRA alkalmazáskörnyezetben

A 3D CRA alkalmazás a yarn Webpack[165] module bundler alapbeállítással megfelelően optimalizált statikus állományokat alakít ki. A szerkesztőből futtatható kliens-felület back-end részét a Node.js minimalista keretrendszere, az Express.js egyszerűsíti.[166]

Az Express.js egy webalkalmazás-keretrendszer, amely széleskörű szolgáltatás-készletet biztosít a back-end folyamatok kidolgozásakor. [167] Az alkalmazás útvonalakhoz

---

[164] JSDoc: JSDoc Official documentation. *JsDocApp*, 2020. (https://jsdoc.app/) Utolsó letöltés: 2022.10.20
[165] Webpack Group: Webpack documentation - Getting Started. *WebpackJs*, 2023. (https://webpack.js.org/guides/getting-started/) Last accessed: 2023.04.03.
[166] Xu, Jack: Practical WebGPU Graphics: Creating Advanced Graphics on the Web Using WebGPU. *Google Books, Apress*, 2022. (https://books.google.hu/books?id=Qwo9zgEACAAJ.) Utolsó letöltés: 2023-04-04.
[167] StrongLoop - IBM: Express documentation. *ExpressJs*, 2023. (https://expressjs.com/) Utolsó letöltés: 2023.04.05.

kötött kontrollerek kezelését, és az ezeket szabályozó metódusokat támogatja. A Node.js kiegészítő rétegekként egyszerűsíti a szerverek és az alkalmazás-útvonalak kezelését, a készülő interaktív web 3D animációs állományok adatbáziskérések kiszolgálórétegét.

```
yarn add @types/express
npm install @types/express --save

yarn global add @types/nodemon
npm link nodemon
```

[Express és a nodemon[168] telepítése]

A fenti stack mellett megfelelő alternatívát jelent React.js használata esetén a Next.js full-stack keretrendszere is. A React ökoszisztéma mellett a nyílt forráskódú, TypeScript alapú Angular jelenthet még valós alternatívát, amely garantálja a React-hoz hasonló moduláris programozást, és a korszerű fejlesztési környezetet.

A kialakított környezet az MVN *(angolul MVC)* architektúrája szerinti logikai tördelést is támogatja.

5.3.1.1. MVN, azaz Modell, Nézet, Vezérlő

A felhasználói műveletek kéréseit a Vezérlő *(Controller)* látja el. A Modell *(Model)* állományok segítségével a típusdeklarációk, és az adatszerkezettel kapcsolatos feladatok definiálhatók.[169] A disszertáció során létrehozott alkalmazás backend mappájában kaptak helyet az adatbázis konfigurációs fájljai, a vezérlők, továbbá az adatmodellek. A Nézet modul állománya pedig a CRA front-end mapparendszerébe került. A lekérések útvonala az Express.js Router funkcióján keresztül paraméterezhető, és adatbázissal kapcsolatos eseményeket hív meg.

[168] Node.js alkalmazások automatikus (változtatás alapú) újraindításához használt segédalkalmazás.
[169] Microsoft Docs: Model-View-Controller (MVC) overview. *Microsoft*, 2021. nyár (https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0) Utolsó letöltés: 2023.02.10.

5.3.2. 3D modellek adatbázisban: A MySQL és Sequelize ORM alkalmazása a Node.js platformon

A 3D modellel kapcsolatos információk tárolására (*relációs)* adatbázis használható. A MySQL adatbázis-kezelő rendszer letöltése[170] és telepítése után, az adattáblák könnyen elkészíthetők és paraméterezhetők a MySQL Workbench[171] felületén. Ez a GUI leegyszerűsíti az adatbázis kapcsolatok létrehozását és menedzselését, kiválóan használható adatmodellek tervezésére és SQL-lekérdezések futtatására.

A szerkezetet a Sequelize ORM *(Object-Relational Mapping)* keretrendszer segítségével is kialakítható [172]. A Node.js platformhoz kialakított adatbázis-független könyvtár front-end kódokkal képes az adattáblák automatikus kialakítására. Ezeket az ORM rendszer JavaScript oldalon dinamikusan is ki tudja alakítani újraszerkeszthető formában. A Sequelize a lekérdezések kezelésével rendkívül mértékben támogatja az adatbázisműveletekkel kapcsolatos bonyolult munkafolyamatot. [173]

yarn add sequelize, yarn add @types/sequelize

[Sequelize telepítése]

Az adatbázis-séma megvalósítása és az adatbázis importálása után *(Server/Data Import)* után a felhasználói jogosultságok beállításával lehet véglegesíteni a folyamatot. A MySQL adatbázis a VsCode beépülő modulján *(plugin-jén: a VsCode MySQL management tool-on)* keresztül is megtekinthető, kereshető. *[Fig.MySqlWorkB] [Fig.VsMysql]*

Megjegyzés: Localhost-on (a MySQL Workbench alternatívájaként) a XAMPP (ill. WAMP) is alkalmazható, segítségével modellezhető a kapcsolódás távoli Apache szerverhez, továbbá a MySQL / MariaDB adatbáziskezelő-rendszerhez. Az alkalmazás PMA felületén pedig szintén nagyon egyszerűen kialakíthatók az 3D adatbázistáblák, és a modelleket, vagy animációkat definiáló egyedi mezők is könnyen áttekinthetők.

---

[170] Oracle Corporation: MySQL documentation. *MySQL*, 2021. (https://dev.mysql.com/doc/) Utolsó letöltés: 2023.04.03. *(telepítése https://dev.mysql.com/downloads/installer/)*

[171] Oracle Corporation: MySQL Workbench Documentation. *Oracle*, 2023. (https://dev.mysql.com/doc/workbench/en/) Utolsó letöltés: 2023.04.03.

[172] Sequelizedocs contributors: Sequelize. *Readthedocs*, 2022 tavasz (https://sequelize.org/master/) Utolsó letöltés: 2023.04.03.

[173] OpenCollective: Sequelize documentation. *Sequelize.org*, 2023. (https://sequelize.org/docs/v7/) Utolsó letöltés: 2023.03.13.

yarn add mysql2
yarn add @types/mysql
(npm install @types/mysql
npm link mysql / npm install mysql)

[MySQL telepítése]

5.3.3. CRUD műveletek Express.js RESTful API segítségével

A tábla rekordjai kliensoldali űrlapon keresztül tölthetők fel adatokkal, az ehhez szükséges front-end komponenseken keresztül. Az adatrögzítés, és az adatkérés folyamata a Express.js RESTful API-ján keresztül valósul meg. Ezek a RESTful szolgáltatások HTTP kéréseket használnak a CRUD műveletek végrehajtására A CRUD az adatbázis-kezelés alapvető funkcióinak gyűjtőfogalmát takarja. A *Create* a létrehozás, a *Read* az olvasás, az *Update* a frissítés, és a *Delete* a törlés funkcionalitását jelenti.

A táblák médiatípusok szerint eltérő információkat tárolnak a 3D animációs jelenetekkel kapcsolatban. A fájlelnevezés mellett az egyes rekordok olyan adatokat is megőriznek, melyek a térbeli jelenetek leírásához szükségesek lehetnek. DLA mestermunkám elkészítésekor több adatbázistáblát alakítottam, melyek adatfeltöltése minden esetben előzetes validációval *(pl.: Formik[174])* történik.

5.3.4. A médiatartalmak adatfeltöltésének folyamata - Formidable

Az adatfeltöltés folyamata során különböző médiatartalmat *(képeket, videókat, 3D fájlokat)* másolunk a szerver adott mappájába, továbbá az adatbázisban is rögzíthető a releváns információ. A Formidable[175] *(vagy Express Multer[176])* segédkönyvtár támogatásával ez a munkafolyamat könnyen kivitelezhető az Express szerveren. A *"multipart/form-data"* típus meghatározásakor, a multipart üzenetet body és fájl részre különül. A body rész a szöveges mezőket, míg a file-rész, a feltöltött állományokat tartalmazza. Ezeket az állományokat automatikusan átnevezve és egyedi azonosítóval ellátva kell a szerver adott

[174] Formium, Inc.: Formik documentation. *Formik,* 2023. (https://facebook.github.io/react-360/) Utolsó letöltés: 2022.11.24.
[175] Formidable Labs, LLC: Formidable documentation. *Github,* 2023. (https://github.com/node-formidable/formidable /) Utolsó letöltés: 2022.11.24.
[176] Mutler: Mutler documentation. *Github,* 2023. (https://github.com/expressjs/multer) Utolsó letöltés: 2023.01.21.

könyvtárába feltölteni. A feltöltés folyamatának aktuális állapotát a felhasználói felületen százalékos formátummal jeleztem.

```
yarn add @types/formidable
yarn add formidable
```

[Formidable telepítése]

Ezek után már csak azokat a segédalkalmazásokat dokumentálom, melyek az alkalmazás továbbfejlesztésében nyújtanak jelentős támogatást.

5.3.1. További függőségek telepítése a 3D alkalmazás felhasználói felületéhez és adatbázis-kezeléséhez

A grafikus felület, és az adatbázis alapvető függőségei a Yarn segítségével telepíthetők, majd inicializálhatók. A React-Bootstrap[177] a Bootstrap könyvtár[178] komponens alapú alternatívája. A teljes eszközkészlet olyan előre kidolgozott összetevőket tartalmaz, melyek jelentős mértékben gyorsítják a UI *(User Interface) t*ervezését és kialakítását.

```
yarn add react-bootstrap, yarn add @types/react-bootstrap
npm install @types/bootstrap
yarn add @mui/material @emotion/react @emotion/styled
yarn add @mui/types

npm install @mui/material @emotion/react @emotion/styled
```

[Bootstrap és MUI függőségek telepítése]

A Bootstrap könyvtár telepítése után az alapvető, és azonnal használható stílusgyűjteményeket a 3D app nyitó kódsoraiba kell meghívni.

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

[A Bootstrap css állományának betöltése az index.tsx fájlban]

A React-Bootstrap mellett kiváló eszközkészletet jelenthet a Google MUI. A Material UI nevű gyűjtemény számos React komponenssel segítheti, gyorsíthatja a tervezést, az egyedi

[177] React-Bootstrap: React-Bootstrap documentation. *Github*, 2023. (https://github.com/react-bootstrap/react-bootstrap) Utolsó letöltés: 2023.04.05.
[178] Facebook Inc.: Bootstrap documentation. *GitHub*, 2022. (https://getbootstrap.com/docs/5.3/getting-started/introduction/) Utolsó letöltés: 2022.10.10.

felületek kialakítását. A felület layout-ja és tartalma URL hívás alapján, a react-router-en keresztül is vezérelhető.

```
yarn add react-router-dom
yarn add @types/react-router-dom
yarn add @types/react-dom
npm install react-router-dom
npm install @types/react-router-dom
```

[A React router telepítése]

Az Axios olyan Promise API támogatással rendelkező JS könyvtár, amelyet HTTP-kérések küldésére és fogadására használtam, Node.js környezetből, valamint a böngészőből induló XMLHttpRequest-ek kezelésére.[179] A modul jelentősen leegyszerűsítette a kliensoldali lekérdezések szintaktikáját.

```
yarn add axios
yarn add @types/axios

npm install axios
npm install @types/axios
```

[Axios telepítése]

Adott tartományon kívüli, külső erőforrások *(kép, zene, videó stb.)* betöltésekor az azonos eredetű szabály paraméterezésére is szükség van. A CORS *(Cross-Origin Resource Sharing)*[180] a böngészők biztonsági előírásait támogatja, de a megfelelő beállítások segítségével a külső tartományok forrásai is elérhetők.

```
yarn add cors
yarn add @types/cors
```

[CORS telepítése]

A csomagkezelő és az applikáció függőségeinek egyidejű frissítését az alábbi parancssorokkal lehet frissíteni.

```
npm install --global yarn
yarn update
```

[179] Axios: Axios documentation. *GitHub*, 2023. (https://axios-http.com/docs/api_intro) Utolsó letöltés: 2022.07.12.

[180] Mozilla Developer Network (MDN): CORS. *Mozilla Corporation*, 2023. (https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS) Utolsó letöltés: 2023.04.03.

[A már telepített csomagok frissítése]

A 3D alkalmazásfejlesztés során használt összes modult a *package.json* fájl tartalmazza. Ez az állomány írja le, hogy mely összetevőkre épül az applikáció. A lista részletezi a beépülő modulok neve mellett a függőségek verziószámát is jelzi. Ez lényegesen leegyszerűsíti a projektek kezelését, az összetevők későbbi frissítését.

5.4. A 3D applikáció alkalmazásindításának stratégiái

Az előkészületek után a szerver- és a kliensoldali demo-alkalmazás a Concurrently[181] segítségével localhost-on terminálról is indítható, és az alapértelmezett böngészőben azonnal láthatóvá válik. [182]

yarn add @types/concurrently
npm install @types/concurrently

[Concurrently telepítése]

A párhuzamos indítás további feltétele a projekt leírófájljának *(package.json)* módosítása. A meglévő objektumot egyedi kulcsokkal *(client, server, 3d)* és a hozzá kapcsolódó értékekkel *(pl.: "nodemon -r dotenv/config ./src/back-end/server.ts")* bővítve, egyetlen sorra egyszerűsíti a programindítást.

```
"scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "client": "react-scripts start",
    "server": "nodemon -r dotenv/config ./src/back-end/server.ts",
    "3d": "concurrently \"yarn run server\" \"yarn run client\""
},
```

[A VsCode scirpts objektuma. Forrás: Balogh Áron, 2023.]

yarn run 3d

[Alkalmazás egyedi indító parancssora]

[181] Henke Gustavo: Concurrently - Run commands concurrently. *GitHub*, 2023. tavasz (https://github.com/open-cli-tools/concurrently) Utolsó letöltés: 2023.04.07.
[182] A teljes konfigurációs minta a package.json fájlban található

```
$ yarn run dev
yarn run v1.21.1
$ concurrently "yarn run server" "yarn run client"
$ nodemon -r dotenv/config ./src/server/server.js
$ react-scripts start
[0] [nodemon] 2.0.20
[0] [nodemon] to restart at any time, enter `rs`
[0] [nodemon] watching path(s): *.*
[0] [nodemon] watching extensions: js,mjs,json
[0] [nodemon] starting `node -r dotenv/config ./src/server/server.js`
[0] App server now listening to port 3001
```

[A Szerver indítása 3001-es porton. Forrás: Balogh Áron, 2023.]

```
Starting the development server...
Compiled successfully!

You can now view 3d in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.64:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully
No issues found.
```

[CRA indítása 3000-es porton. Forrás: Balogh Áron, 2023.]

```
src > TS index.tsx > ...
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import './index.css';
4   import App from './App';
5   const root = ReactDOM.createRoot(document.getElementById('3DApp') as HTMLElement);
6   root.render(
7     <React.StrictMode>
8       <App />
9     </React.StrictMode>
10  );
```

[Alkalmazás betöltése az adott DOM részre. Forrás: Balogh Áron, 2023.]

Tesztelési- és fejlesztési munka során az egyidejű indítás helyett, az alkalmazást külön szerver- és kliensoldali indítóparancsokkal is indíthatjuk.

yarn run server *(szerver indítása)*
yarn client *(kliens indítása)*

[Az alkalmazás különálló indítóparancsai]

A fenti módszerekkel elkészült full-stack applikáció alapja megfelelő hátteret és felhasználói kezelőfelületet biztosít a térbeli jelenetek tárolására, és a 3D jelentek integrációs munkáinak megkezdéshez.

6. Jelenetintegráció: A webvizualizációs 3D segédkönyvtárak alkalmazása

Ebben a fejezetben az adaptálhatóság aspektusából elemzem a legmodernebb és a hagyományos grafikus modellező felületeket. Az a célom, hogy bemutassam a 3D modellek és jelenetek webes platformokra történő transzplantálásának folyamatát, különös tekintettel a WebGL és a WebGPU technológiákra. A fejezetben rávilágítok azokra az eszközökre és módszerekre, amelyek lehetővé teszik a digitális 3D tartalmak internetes környezetben történő megjelenítését. Ismertetem a hagyományos tervezői és animációs szoftverek releváns funkcionalitását, a real-time game engine-ek és a broadcast valósidejű képalkotó eljárások kapcsolódó tulajdonságait, és bemutatom az exportálási lehetőségek kimeneti formátumopcióit.

A megfelelő absztrakciós szint kiválasztása és definiálása során a web 3D segédkönyvtárak transzplantációs készleteit értelemezem. Kísérleti példa segítségével mutatom be a tartalomintegrációs módszerek tulajdonságegyüttesét; a 3D állománybetöltést, a jelenetkezelést és jelenetparaméterezést, valamint az elkészült alkotás publikációval kapcsolatos osztálymetódusait.

6.1. 3D tervezői forráseszközök ismertetése

Ebben az alfejezetben rövid összehasonlító körképet adhatok az aktuálisan elérhető web 3D kimeneti formátumok támogatottságról. Elemzésem során néhány népszerű szoftver *(Unity, Unreal Engine, Autodesk Maya, Blender, valamint Viz Artist)* exportálással kapcsolatos funkciókészletét hasonlítom össze.

6.1.1. A valós idejű játékmotorok adatforrás aspektusának elemzése

Valósidejű vizualizációs platformok GPU API-jának tervezésekor az OpenGL-re épített keretrendszerek mellett a BgFX[183], a Sokol-GFX [184], a Vulkan Portablility [185], vagy épp a korábban már említett Metal és a DirectX 12 jelenthet kiindulópontot[186]. Kétségtelen

[183] Bgfx: Cross-platform rendering library. *Github*, 2023. (https://github.com/bkaradzic/bgfx) Utolsó letöltés: 2023.04.01.
[184] Sokol: Simple STB-style cross-platform libraries for C and C++. *Github*, 2023. (https://github.com/floooh/sokol) Utolsó letöltés: 2023.04.01.
[185] Khronos Group Inc.: Vulkan Porting Layers. *Khronos*, 2022. (https://www.vulkan.org/porting) Utolsó letöltés: 2023.03.21.
[186] Dzmitry Malyshau: Point of WebGPU on native. *Github*, 2020. nyár(https://kvark.github.io/web/gpu/native/2020/05/03/point-of-webgpu-native) Utolsó letöltés: 2022.01.10.

azonban, hogy ezek az API-k nem nyújtanak megfelelően értelmezhető kezelőfelületet az animációs alkotók, vagy mozgóképes szakemberek számára. Valósidejű 3D animációs szempontból napjaink elsődleges felületei a legnépszerűbb *low-level* grafikus API-kra építkező *real-time game engine-ek*, vagyis az Unreal Engine és a Unity engine.

Az Unreal Engine és a Unity Engine a valósidejű grafikus és animációs tervezőrendszerek jelenlegi piacvezetői. Mindkét játékmotor kiváló kiindulópontja a grafikus applikációfejlesztésnek és az interaktív mozgóképes alkotások tervezésének. Ezek a rendszerek olyan felhasználói felülettel *(GUI)* rendelkeznek, amelyek a hagyományos 3D tervezői platformokhoz hasonlítanak, és kellően intuitívak is az alkalmazógrafikusok számára. Az Unity Engine kiváló fejlesztői dokumentációval rendelkezik[187], és a kifejezetten az ajánlott platformok közé sorolható. Natív WebGL exportálási opciót biztosít; fájlmentéskor WebGL sablonokat és megfelelően paraméterezhető exportálási felületet ad a kreatív szakemberek kezébe. A játékmotorként használt Unreal Engine grafikus vezérlője, kutatásom szempontjából kiemelkedő fontosságú volt. A programban elérhető *Blueprint* technológia a node-alapú tervezői és fejlesztői koncepcióhoz áll a közelebb, mégis vizuális programnyelvek segítségével leírható eszközkészletet biztosít. Annak ellenére, hogy az Unreal Engine világvezető vizualizációs alkalmazás, keretrendszerében jelenleg nem kapott helyet a WebGL és a WebGPU kimeneti támogatása.[188] Esetében egyértelműen kijelenthető, hogy ez a hiányosság átmeneti jellegű, és a WebGPU 3D technológia térhódításával hamarosan használható kimeneti forrása lesz.

Bár a valósidejű tervezés produkciós igénye folyamatosan bővül, a 3D animációs és modellező szoftverek továbbra is a térbeli jelenettervezés legelterjedtebb eszközei. Ugyan ezek a modellező applikációk is rengeteget változtak, és egyre közelebb kerültek a valós idejű képalkotás funkcionalitásához *(lásd Blender EEVEE[189])*, egyelőre csak részlegesen, beépülő modulokkal képesek az interaktív színterek irányába dolgozni.

[187] Unity Technologies: Unity Engine. *Unity3D,* 2023.
(https://docs.unity3d.com/2023.2/Documentation/Manual/webgl.html) Utolsó letöltés: 2023.04.01.
[188] Unreal Engine: Unreal Engine 5 Documentation. *UnrealDocs,* 2023.
(https://docs.unrealengine.com/5.0/en-US/) Utolsó letöltés: 2023.04.01.
[189] Blender Foundation: Blender documentation – Eevee. Doc. *Blender,* 2023.
(https://docs.blender.org/manual/en/latest/render/eevee/materials/settings.html) Utolsó letöltés: 2023.04.05.

6.1.2. 3D modellező és animációs szoftverek és az elérhető exportálási lehetőségek ismertetése

A legnépszerűbb térgrafikai alkalmazások is csak pluginek támogatásával képesek biztosítani a web 3D exportálást. Számos külső kiegészítő létezik, mely biztosítja ezt a funkcionalitást, ezek produkciós felhasználása ugyanakkor korlátozott, verziókövetésük sok esetben nem megfelelő. Az alkotók leginkább bevált módszere, hogy az elkészülő modelleket előzetes konverzióval új 3D-s formátumra transzkódolják a lengépszerűbb *Open Source* alkalmazások valamelyikén. A *Blender*[190] napjaink leggyakrabban használt klasszikus modellező és egyben web 3D tervezői forrása lett. Kiválóan paraméterezhető, natívan beépülő konverziós osztályrendszere széleskörű kimeneti támogatást nyújt az artist-ok számára. Az exportálás -*minden szoftver esetén*- a Khronos által meghatározott web 3D fájlformátumok irányába kezdeményezhető.

6.2. A szabványosított interoperábilis 3D reprezentációs modellformátumok jellemzése

A glTF *(GL Transmission Format)* és a GLB fájlformátumok nyílt forráskódú szabványok, melyek a 3D modell- és jelenetadatok univerzális adattovábbítását és a művészi tartalom interoperábilis felhasználatát biztosítják.[191] Ez azt jelenti, hogy a glTF fájl szöveges JSON állomány, melynek tartalmi elemeit és paramétereit könnyen olvasni és editálni lehet. Paraméterei tehát lehetővé teszik az adatok értelmezését eltérő megjelenítési kontextusok szerint is.[192]

A GLB a glTF formátum bináris megfelelője, mely kompozíciós eljárás során egyetlen fájlba csomagolja a 3D modellel kapcsolatos összes információt: a jelenet és a benne szereplő elemek adatait, a textúrákat és egyéb összetevőket. A GLB a web 3D ökoszisztéma egyik legfontosabb és széles körben alkalmazott formátuma. Mindkét típus egyébként alapértelmezésként tekinthető, és a megfelelő fejlesztői absztrakciós szinten könnyedén érvényesíthető.

---

[190] Blender: Open Source 3D modellező: https://www.blender.org/

[191] Khronos Group: GLTF documentation. *OpenGL.org*, 2023. tavasz (https://www.khronos.org/gltf/) Utolsó letöltés: 2023.02.10.

[192] Jones, Brandon: Efficiently rendering glTF models - A WebGPU Case Study. *Github*, 2023. (https://toji.github.io/webgpu-gltf-case-study/) Utolsó letöltés: 2023.03.21.

6.3. Kényelmi absztrakciós fejlesztői réteg definiálása web 3D vizualizációkhoz

A modellezés során kialakított grafikus objektumok, térbeli modellek exportálása után az adatok befogadásának környezeti feltételeinek megteremtése szükséges. Ennek alapfeltétele a web 3D segédkönyvtárak ismeretanyagának feltárása.

6.3.1. Web 3D könyvtárak ismertetése és összehasonlítása

A Web 3D könyvtárak online környezetben teszik lehetővé gLTF, vagy GLB formátumú 3D animációs jelenetek publikálását. A 3D JavaScript *libary-k* támogatják az eltérő formátumú modellek, jelenetek importálását *(beolvasását)*. Az eljárás-gyűjtemények különféle funkcionalitással és eltérő preferenciákkal rendelkeznek, felhasználásukat a kiválasztott adaptációs- és kimeneti formátum jelentősen befolyásolja. A WebGL és WebGPU-ra épülő absztrakciós osztályok közös tulajdonságaik alapján összegezve is jellemezhetők.

A 3D segédkönyvtárak többsége nyílt forráskódú szoftver. Olyan szabadon felhasználható, módosítható magasszintű grafikus keretrendszerek, melyek forráskódja publikus, és mindenki számára elérhető. Alkalmazásprogramozási interfészük segítségével lényegesen leegyszerűsítik a fejlesztői munkafolyamatokat, melyek elengedhetetlenek a gyorsabb grafikai vizualizációk kialakításához. A dinamikus kontrollerek lehetővé teszik, hogy a felhasználók interakcióba lépjenek az elkészült grafikákkal, animációkkal. Támogatják a beolvasott modellek jelenetparaméterezését: a térbeli pozícionálást, bevilágítását, textúrázását. Bár a fények és árnyékok kezelését számos paraméterdefiníciós opció segíti, sok esetben azonban a hagyományos tervezőprogramokban elért vizuális hatástól elmaradó grafikai eredményeket lehet *(jelenleg)* elérni, ami jelentősen megnehezíti a transzplantációs eljárást.

Az eszközök funkciókészletét, sajátos eljárásait a szoftverek egyedi dokumentációja összegezi, amely mindegyik példa esetében ingyenesen elérhető. A programozó és tervezőgrafikus felhasználók közössége folyamatos javításokkal, javaslattételekkel és funkcióbővítéssel támogatja a könyvtárak továbbfejlesztését. Összetételük, forráskódjuk így dinamikusan és folyamatosan fejlődik. Az elérhető legújabb verziók közötti átállást, a verziók közötti változásokat, előzetes tesztelések után számos platformon publikálják, dokumentálják. A legmegfelelőbb könyvtárak kiválasztását az animáció célja, jellege, formátuma mellett, az alkalmazási terület, továbbá a teljesítményigény is meghatározhatja.

Egyes web 3D könyvtárak konverziós-exportálási módszereket is biztosítanak az Extended Reality immerzív platformjainak irányába. Az XR felületekre előzetesen paraméterezett digitális jelenetek az adaptációs eljárások során olyan attribútumokat szereznek, melyek lehetővé teszik az AR, VR, vagy MR publikációt. Az XR, vagy Web XR támogatás biztosítja a grafikus tervezők számára, hogy olyan látványos interaktív animációkat hozzanak létre, melyek az elképzelt digitális környezetet valósághű térélménnyé, kiterjesztett fél- vagy többletvalósággá tehetik.

Előzetes elemzéseim alapján megállapítottam, hogy a Three.js és a Babylon.js magas szintű könyvtárak, melyek megfelelő absztrakciós szintet jelentenek web 3D front-end fejlesztők vagy multimédiatervezők számra.[193] Ezek a felületek már sokkal inkább közelíthetők a 3D szoftverek használatához *(Blender, C4D, stb.)* mint a tényleges GPU interakcióhoz. Ennek hiányában ugyanis még a gyakorlottabb fejlesztők számára is kihívást jelenthet a térbeli ábrázolás fejlesztési folyamatainak összeállítása. A világ egyik vezető applikációjának programozója is hasonlóképp jellemzi az OpenGL alapú WebGL fejlesztés időszakát. Arról a személyes tapasztalatáról ír, mely során még natívan kísérletezett WebGL objektumok kialakításával, és nem használta az következő fejezetben ismertetett 3D segédkönyvtárak absztrakciós szintjét.

„Az OpenGL API-ja önmagában még régebbre nyúlik vissza, és a mai sztenderdek szerint nem egy nagyszerű API. A tervezés középpontjában egy belső, globális állapotobjektum áll. [...] az API hívások sorrendje rendkívül fontos, és mindig úgy éreztem, hogy ez megnehezíti az absztrakciók és a könyvtárak építését. Minden pointert és állapotelemet nagyon gondosan kell kezelni és visszaállítani az előző értékére, hogy az absztrakciók helyesen összeálljanak. Gyakran előfordult, hogy üres fekete vászonra bámultam *(mivel ez gyakorlatilag az összes hibaüzenet, amit WebGL-ben kapni lehet)*, és próbáltam kitalálni, hogy melyik pointer mutat most éppen rossz irányba. Őszintén szólva fogalmam sincs, hogyan tudta a Three.js ilyen robusztusra építenie rendszerét, de valahogy sikerült neki. Úgy gondolom, hogy ez az egyik fő oka annak, hogy a legtöbb ember a ThreeJS-t és nem a WebGL-t használja közvetlenül.”[194]

---

[193] Seguin Damien: Graphics on the Web and Beyond with WebGPU. *Medium,* 2020. nyár(https://dmnsgn.medium.com/graphics-on-the-web-and-beyond-with-webgpu-13c4ba049039) Utolsó letöltés: 2023.04.02.

[194] Bynens, Mathias: WebGPU — All of the cores, none of the canvas. *Surma.dev*, 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.

Bár a 3D segédkönyvtárak kissé eltávolodnak a GPU API-k tényleges működési modelljétől, azonban megfelelő kiindulópontot jelentenek minden 3D vizualizációs tervező számára. Ezek alapján elismerem, hogy multimédiafejlesztői szempontból sok esetben nehezen indokolható az OpenGL, és a WebGPU natív fejlesztői környezetének használata. Összességében elmondható, hogy a *Three.js és a Babylon.js* megfelelő *kényelmi réteget* nyújtanak a webes 3D vizualizációs fejlesztésekhez.

6.3.2. Three.js

A Three.js olyan JavaScript segédkönyvtár, mely 3D vizualizációs és animációs környezet kialakítására használható böngészőalapú környezetben. Mint ahogy azt korábban összefoglaltam, a WebGL technológia *(egyelőre!)* OpenGL-re épül[195],, nyílt forráskódú keretrendszerként egy évtized alatt az interaktív képalkotás meghatározó szereplőjévé vált. Kényelmi absztrakciós fejlesztői réteget biztosít a GPU-alapú böngészőben megjelenő vizualizációkhoz. (*A disszertáció megírásának idején a WebGPU támogatás még nem elérhető a libary-ben.)* A keretrendszer biztosítja az online tervezést, továbbá az előre elkészített jelenetek beolvasását is. A jelenetkezelést, a térbeli kompozíciók paraméterezését saját függvényei segítségével egyszerűsíti. A HTML5 Canvas DOM elemébe épülő 3D WebGL technológia broadcast térhódítása is jelentősen felgyorsult, széles körben felhasználható interfészt biztosítva a televíziós képalkotók számára is. A könyvtár használatához további kiegészítők, vagy modulok nem szükségesek.

A Three.js-ben elérhető objektumok és geometriák, a textúrák, fények és árnyékok használatának támogatása a magasabb minőségű 3D jelenetek megalkotását is elősegíti.[196] A keretrendszerben megtalálható eszközök és modulok támogatják a PBR-t (*Physically Based Rendering)* [197], ami a realisztikusabb megjelenésű anyagok és felületek létrehozásában segít. A WebGL 2.0 óta bevezetett újítások, mint például az *instancing*, és a tömörített textúrák támogatása jelentősen javítják a Three.js teljesítményét és memóriakezelését. A Three.js továbbá a *Shader Material* funkcióval támogatja a GLSL *(OpenGL Shading Language)* alapú

[195] Armour, Theo - Cabello, Ricardo - Masson, Paul: three.js. *Github Inc.*, 2018. (https://threejs.org/docs/index.html#manual/introduction/Creating-a-scene) Utolsó letöltés 2018.04.27.
[196] Tanács Attila: Three.js jegyzet Számítógépes grafika gyakorlathoz. Szegedi Tudományegyetem, 2022. (http://www.inf.u-szeged.hu/~tanacs/threejs/index.html) Utolsó letöltés: 2022.07.09.
[197] Adobe Inc.: Everything you need to know about physically based rendering. *Adobe Systems*, 2022. (https://www.adobe.com/products/substance3d/discover/pbr.html) Utolsó letöltés: 2022.07.12.

egyedi árnyalók használatát.[198] Az árnyalók lehetővé teszik, hogy a vizuális effektek és az animációk teljes mértékben testre szabhatók legyenek.

A Three.js támogatja a *virtuális valóság (VR, Virtual Reality)* a *kiterjesztett valóság (AR, Augmented Reality)* és a *vegyes (vagy kevert) valóság (MR, Mixed Reality)* technológiák integrálását is. A Three.js libary tehát a valósidejű XR *(Extended Reality, vagy bővített valóság)* alkalmazásokat is támogatja a böngészőkörnyezetben.

Meglévő alkalmazáskörnyezetbe az alábbi parancssorral illeszthető.

yarn add three, yarn add @types/three

[Three.js telepítése]

6.3.2.1. React Three Fiber

Az disszertáció során megalkotott fejlesztői ökoszisztéma részét képezi a React Three Fiber segédkönyvtár is. Olyan komponens alapú kiegészítő, mely a React.js és a Three.js összekapcsolásában segíti. A könyvtár automatikusan gondoskodik a Three.js objektumok életciklusának kezeléséről: térbeli elemek létrehozásról, paramétereinek frissítésről és a memóriakezeléséről. [199]

6.3.3. Babylon.js

A Babylon.js egy rendkívül jelentős és dinamikusan fejlődő JavaScript könyvtár a webes 3D képmegjelenítés terén. Jelenleg még egyedülállóként rendelkezik WebGPU támogatással, amely magasabb teljesítményt és gyorsabb renderelést biztosít a kreatív front-end fejlesztők és grafikus tervezők számára.[200] A babylon.js beépített fizikai szimulációs felületeivel kiemelkedő animációs és vizualizációs alkotások valósíthatók meg.

A Babylon.js a Cannon.js-re épülő nyílt forráskódú szimulációs motorral, azaz realisztikus fizikai hatásokkal bővíti a webes 3D jelenetek tervezési és megjelenítési eszköztárát. A környezetei effektorok automatikus kalkulációja mellett a dinamikai, és szimulációs eljárások is elérhetővé váltak. A részecske-, a folyadék- vagy ütközésszimulációs

[198] Three.js: Three.js documentation. *Github,* 2023. (https://threejs.org/docs/#api/en/materials/ShaderMaterial) Utolsó letöltés: 2023.04.01.
[199] Pmndrs: React Three Fiber. *Github,* 2023. (https://docs.pmnd.rs/react-three-fiber/getting-started/introduction), Utolsó letöltés: 2023.04.05.
[200] Babylon.js: Babylon.js documentation. *Github*, 2023. (https://doc.babylonjs.com/) Utolsó letöltés: 2023.04.01.

effektek, a reakció- és hatásmechanizmusok vizualizációs lehetőségei egyértelműen a web alapú CGI valósághű képalkotás első fontos mérföldköveit jelentik. A Babylon.js ezek mellet a mozgás összetett ábrázolását is támogatja beépített inverz kinematika *(BoneIKController)* opcióval.

A Babylon.js grafikus és animációs könyvtár saját *.babylon* kiterjesztésű állománydeklarációval rendelkezik, amely a legtöbb információt képes kezelni az összes támogatott formátum közül. Egyetlen hátránya, hogy saját exporterének használata megkerülhetetlen, így alkalmazási környezete jelentősen beszűkül. Vizsgálatom arra következtetésre jutottam, hogy egyelőre célszerű a korábban említett platform-független GLB, vagy glTF fájlformátumok használata.

6.4. Tartalomintegrációs módszerek kutatási eredményeinek összegzése

A disszertáció részeként létrejövő 3D adatbázis példamunkái során a Three.js és a Babylon.js könyvtárakat használtam fel.[201] Lehetőségem volt tanulmányozni az 3D állománybetöltés módszereit és a jelenetkezelés beállításaival kapcsolatos sajátosságokat.

Egyaránt kísérletezem gLTF és GLB fájlformátumok betöltésével. Olyan 3D modelleket importáltam, melyeket előzetesen eltérő tervezőgrafikai platformokon készítettem. A jelenetek különböző geometriai modellekkel bővítettem, majd a kamera pozícióját és nézőpontját állítottam be a megfelelő vizualizáció érdekében. Fejlesztői tevékenységem során a materiálok és textúrák kezelését, valamint paraméterezését is elvégeztem. Ezt követően fényeket és árnyékokat adtam a jelenethez, amelyek finomították vizualizáció megjelenítésminőségét, élethűségét. A kísérletek során a plasztikus kamerák modern megfelelőjét, a Three.js és Babylon.js natív sztereó kameráját is kipróbáltam, amely lehetővé tette a 3D tartalom két különálló képként történő megjelenítését, végeredményként élethűbb térérzetet biztosítva. A fizikai tulajdonságok és részecskerendszerek integrációjával tovább bővítettem a kísérleti alkalmazás funkcionalitását, amelyek segítségével realisztikus mozgásokat, ütközéseket és effekteket modellezhettem. Az utolsó kísérleti fázisban post-processing technikák alkalmazásával javítottam a végső képminőségen. Ezen módszerekkel

[201] Tanács Attila: Three.js jegyzet Számítógépes grafika gyakorlathoz. Szegedi Tudományegyetem, 2022. (http://www.inf.u-szeged.hu/~tanacs/threejs/index.html) Utolsó letöltés: 2022.07.09.

élsimítást, színkorrekciót és mélységélességet állítottam, melyekkel tovább tudtam növelni a vizualizáció élethűsége és részletessége.

A Three.js és Babylon.js könyvtárak alkalmazásával olyan 3D vizualizációs komponenseket és interaktív animációkat sikerült kifejlesztenem, amelyek széleskörű alkalmazhatósággal és adaptációs képességgel rendelkeznek. Eredményeim alapján megállapítható, hogy ezen könyvtárak ideálisak a háromdimenziós tartalmak webes reprezentációjához és menedzseléséhez, továbbá a jövőbeni televíziós rendszerek integrációs igényeihez is alkalmazhatók.

A kutatás során, a 3D adatátvitel és bonyolult konverzió miatt egyes esetekben problémákba ütköztem. A felmerülő akadályok megoldása érdekében kisebb felületszámú geometriai formákkal kísérletezem. A problématerületek könnyebb feltárásához az egyszerűbb geometriai alakzatok elemzésétől fokozatosan haladatam a magasabb komplexitású képi rendszerek felé. Problémafeltáró munkásságomat így néha az egyszerű vonalaktól kezdtem.

A grafikai adaptációk során a 3D kompozíciót leíró objektumok torzult paraméterkészletének problematikáját fedeztem fel. Egyes esetekben a forrásmodell csupán módosított paraméterállománnyal örökölhető a valósidejű grafikai rendszerekbe. Ha nem valósidejű környezetben hozzuk létre a térkompozíciókat, az egyszerű formák felületszáma, a görbék modellezése, a vertexek indexelési rendszere, sőt, egyes esetekben még a globális és lokális koordináták is a megjelenítési irányra optimalizálódnak. Az eltérő háttérállománnyal rendelkező mozgóképes tartalmak vizuális megjelenítése, méretezése, újraparaméterezése és reaktív tulajdonságai így nem voltak optimálisan kezelhetők egyes helyzetekben.

A fenti problémák megjelölése a valósidejű képgenerálás és a grafikus eljárás bonyolultságát is visszatükrözi. Annak a példázataként is értelmezhető, hogy még a legegyszerűbb geometriai alakzatok esetében is előfordulhatnak exportálási és indexelési problémák, amelyeket különböző komplexitású interaktív modellek ábrázolásával igazoltam.

A disszertációmban szerzett tapasztalataim és kísérleti kutatómunkám során ugyanakkor bebizonyosodott, hogy a MySQL, Express.js, Node.js, és a React ökoszisztémájú Full-Stack fejlesztési környezet, kiegészülve a Three.js *(React Three Fiber)* és a Babylon.js segédkönyvtárak tulajdonságrendszereivel, kiválóan alkalmazható böngészőalapú, platformfüggetlen 3D vizualizációs eszközként. Megfelelő minőségű eljárást alakítottam ki a

3D tervezői szoftveres környezetből *(Bledner, Unity, Autodesk Maya)* érkező 3D objektumok optimalizálásához *(Draco [202])*, továbbá a gLTF és GLB formátumú modellek transzplantálásához is. Sikerült megvalósítanom továbbá külső adatforrásokkal összekötött interaktív animációkat is, amelyeket broadcast munkáim során is alkalmaztam. Emellett érintőképernyős 3D vizualizációs felületek létrehozására is képes voltam, amelyeket a jövőbeli televíziós és online streaming rendszerekbe való integrálás terén is hasznosnak találtam.

A sikeres tartalomintegrációs formátumok és módszerek kísérleti munkássága után a publikálás stratégiai kérdéseit elemeztem.

[202] Google Inc.: Draco 3D data compression. *Github*, 2023. (https://google.github.io/draco/) Utolsó letöltés: 2023.04.05.

7. A web 3D vizualizációk publikálási stratégiái

A televíziós és online streaming szolgáltatók többsége már elkezdte az HTML5 alapú valós idejű 3D technológiák integrálását saját rendszereibe. A folyamat eredményeként, a fejlesztési- és videografikai szakemberek, az alkotói stábok új kihívásokkal néznek szembe. Jelen disszertációban rögzítem azokat a publikációs módszereket és stratégiákat, amelyek az általam korábban felvázolt módszerekkel megvalósíthatok.

7.1. A publikálás technológiai aspektusai

7.1.1. Valósidejű broadcast alkalmazáskörnyezet

A nem böngésző-alapú real-time engineket a mozgóképes területek többségén már régóta használják. *S*zámos olyan kereskedelmi szoftver áll a broadcast csatornák és streaming szolgáltatók rendelkezésre, amelyek a 3D tartalmak létrehozását és azonnali publikálást segítik. Ezek közé tartozik például a Vizrt *termékcsalád (Viz Artist, Viz Engine, Viz Trio, Viz Mosart, Viz Virtual Studio, stb.),* az Aximmetry, az Orad, a Brainstorm vagy épp a ChyronHego. Ezek a szoftverek különböző funkcionalitást biztosítanak a valós idejű grafikai tervezéshez és alkalmazáshoz. Webes integrációra többségük beépülő modulokat biztosít.

7.1.1.1. ON AIR vizualizáció: Viz Engine és a CEF integrációs módszer

A WebGL és WebGPU alapú fejlesztés a broadcast környezetben elérhető legnagyobb valósidejű szoftverek, alkalmazások grafikus eszközkészletét is kiegészítik. A böngésző alapú HTML5 megjelenítést a nemzetközi televíziós környezetben legtöbbször használt Viz Engine legfrissebb verziója CEF Library-n keresztül támogatja.[203] A CEF *(Chromium Embedded Framework)* meglévő alkalmazások számára teszi elérhetővé a HTML5-kompatibilis böngészőfunkcionalitást és eszközkészletet, ideértve a web 3D megjelenítést is. Következésképpen, a webgrafikus rendszerek integrált formában elérhetők lesznek a legnagyobb broadcast vizualizációs platformokon is. Erre példa a Viz Artist nevű szoftverkörnyezetben elérhető *Browser plug-in*[204] *(korábban BrowserCEF)*, mely lehetővé

[203] Chromium: CEF – Chromium Embedded Framework. *Github*, 2023. (https://github.com/chromiumembedded/cef) Utolsó letöltés: 2023.04.05.
[204] Vizrt Inc.: Browser plugin. Vizrt, 2023. (https://documentation.vizrt.com/viz-plugins-user-guide/5.0/Browser.html) Utolsó letöltés: 2023.04.01.

teszi a böngésző tartalmának streamelését egy textúrára, ezáltal a WebGL és WebGPU-ban létrejövő 3D alkotásokat is képes élő adáskörnyezetben interaktív tulajdonságokkal megjeleníteni.

A televíziós csatornák virtuális stúdiók formájában, *ON AIR* adatvizualizációs eszközként, AR grafikaként, MI tartalomgenerálás formájában, vagy a közönségbevonás interaktív felületeként hasznosíthatják a technológiát. A következő lehetséges integrációs módszer már az online streamingekhez köthető.

7.1.2. Online stream-ek: WebRTC

Egy WebRTC[205] *(Web Real-Time Communication)* stream[206] adásába ma már gond nélkül WebGL és WebGPU képtartalom illeszthető[207]. Ez azt jelenti, hogy bármilyen web- vagy applikáció élőadás-környezetében felhasználóhatók a web 3D technológiával létrehozott tartalmak. WebRTC stream libary-k segítségével (*SimpleWebRTC, Janus, PeerJS, EasyRTC)* lehetőség nyílik a valós idejű, magas minőségű 3D tartalmak közvetítésére a böngészőkörnyezetben[208].

7.1.3. Immerzív terek: WebXR

Az disszertációban ismertetett technológiai eljárások lehetővé teszik a virtuális és kiterjesztett valóságban történő publikációt is[209]. A WebXR *(Web Extended Reality)*[210] vizualizációs környezet például a már említett eszközök mellett, React-xt[211] és drei[212] könyvtárak használatával könnyen és hatékonyan megvalósítható. A WebXR technológia a

[205] WebRTC: Real-time communication for the web. *Google Developers*, 2023. (https://webrtc.org/) Utolsó letöltés 2023.02.10.
[206] Mozilla Developer Network (MDN): WebRTC API. *Mozilla Corporation*, 2023. (https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API) Utolsó letöltés: 2023.04.07.
[207] Damján Szabolcs: Real-time WebGL video manipulation. *Medium* 2022. tél (https://medium.com/docler-engineering/webgl-video-manipulation-8d0892b565b6) Utolsó letöltés 2023.02.10.
[208] Damján Szabolcs: Manipulating video in a browser. *Medium* 2022. tél (https://medium.com/docler-engineering/manipulating-video-in-a-browser-5b37f8149d9b) Utolsó letöltés 2023.02.10.
[209] Microsoft Learn: Oktatóanyag - Az első WebXR-alkalmazás létrehozása a Babylon.js használatával. *Microsoft*, 2023. tavasz (https://learn.microsoft.com/hu-hu/windows/mixed-reality/develop/javascript/tutorials/babylonjs-webxr-helloworld/introduction-01) Utolsó letöltés: 2023.04.08.
[210] WebXR: WebXR Device API Explained. *Github*, 2022. tél (https://github.com/immersive-web/webxr/blob/master/explainer.md) Utolsó letöltés 2023.04.07.
[211] Poimandres: React-XR. *Github*, 2023. (https://github.com/pmndrs/react-xr#readme) Utolsó letöltés 2023.04.07.
[212] Poimandres: Drei. *Github*, 2023. (https://github.com/pmndrs/react-xr#readme) Utolsó letöltés 2023.04.07.

headsetek és egyéb kiterjesztett valóság eszközök használatát támogatja a böngészőkön keresztül. API-jával térképek, navigációs adatok, és más valósidejű információ integrálására is lehetőség nyílik. A létrejövő térélmény kiterjesztheti video streamingek *(180 vagy)* 360 fokos adásképét, egyedülálló informatív tereket alkotva. Az XR technológiák mellett a beépített sztereó kamerák is jól használhatók.

7.1.4. A holografikus ábrázolás szimbolikus víziója

A korábban ismertetett 3D könyvtárak *(mint például a Three.js, Babylon.js vagy a A-Frame)* közvetlenül nem rendelkeznek beépített lehetőséggel a holografikus képábrázolásra. A holografikus lejátszók *(Glass Core JS, HoloPlay, Holo streaming player)*, valamint ezek renderelési felületei lehetővé teszik a 3D könyvtárak közvetlen megjelenítését hologram projektoron keresztül *(Hologram Shoe Box)* [213].

A fenti 3D JavaScript eljárás mellett a holográfia filmtechnikai fejlődése napjainkban is tovább zajlik. Bár a mozifilmek frame sebességének töredékén tud csak megvalósulni, a kutatások egyre több pozitív eredményt hoznak.[214] A holográfia elvén megvalósuló 3D-s film pedig egyre jobban összekapcsolódik a kiterjesztett valóság projekciós módszereivel. Balogh Tibor HoloVízió nevű fejlesztése a 2000-es években indult. 3D technológiai fejlesztésének kiemelt célja, hogy kijelző technológiájuk révén *(Holograika)* megvalósítsák a valós 3D interaktív televíziós térélményt. A segédalkatrészek nélkül is megtekinthető térbeli adásképet a világ egyik elismert televíziós fejlesztéseként tartják számon napjainkban is. [215]

7.1.5. Interaktív animációk és filmek

A Web3D interaktív alkalmazása során a fejlesztők szkripteket és egyedi eljárásokat használnak az animációkhoz és HTML5 alapú videókhoz kapcsolódó események kezelésére. Az ilyen alkotásokban a nézők felhasználói döntéseikkel aktívan befolyásolhatják a történetet*[216]*, meghatározva például a mozgás irányát, sebességét, vagy a megjelenített

[213] Hologram Shoe Box: https://thisisgrow.com/labs/hologram-shoe-box
[214] Közeleg a hologram fénykora. HVG, 2011.
(Https://hvg.hu/tudomany/kozeleg_a_hologram_fenykora_9j53di) Utolsó letöltés: 2018.01.20.
[215] Schreiber András: 3D idehaza, Magyar dimenzió. *Filmvilág*, 2006. nyár
(https://filmvilag.hu/xista_frame.php?cikk_id=8665) Utolsó letöltés: 2022.05.17.

[216] Bártfai Andrea: Az interaktív film. *Médiakutató*, 2011. tél

tartalmakat. A web 3D videótexturaként[217] képes megjeleníteni a HTML5 és JavaScript alapú videómegtekintő rendszerek *(Video.js, Popcorn.js, H5P)* tartalmát, az éppen lejátszott képszekvenciákat. A két technológia ötvözése lehetővé teszi a teljesen interaktív filmkészítését, továbbá kreatív eszközöket garantál az alkotói munkában résztvevők számára.

7.1.6. MI 3D böngészőkörnyezetben

A 3D webes grafikai alkalmazások és a mesterséges intelligencia egyidejű alkalmazása egyre inkább népszerűbbé válik a modern technológiai közegben. A WebGPU és a TensorFlow[218], illetve TensorFlow Graphics[219] együttes használatával kifejezetten izgalmas interaktív megoldások kelthetők életre a 3D böngészőkörnyezetben.

7.1.7. Cross-platform mobil alkalmazások

Hibrid alkalmazásépítéshez cross-platform eszközöket is igénybe vehetünk. A Cordova/PhoneGap, Appcelerator, Sencha, Cocos2d, Meteor, Capacitor natív jelleggel, platformspecifikus alkalmazásba csomagolja a meglévő kódállományt. Az alkalmazás csomagolását végző komponensei a WebView-k, melyek a platformok közötti átjárhatóságot biztosítják.

7.2. A publikálás a felhasználási területek aspektusából

A 3D WebGPU felhasználási területei igen széleskörűek. A technológia egyre nagyobb jelentőséget kap a broadcast és az online streaming szférájában, ugyanis a 3D grafikai megjelenítés és az intelligens adatfeldolgozás kiemelt stratégiája a modern képalkotásnak. A dinamikus vizuális eszközök használata ma már elengedhetetlen olyan területeken, mint például az időjárásjelentések megjelenítése, sportesemények és sportolók teljesítményadatainak vizualizációja, választási műsorok eredményeinek grafikus ábrázolása,

(https://mediakutato.hu/cikk/2011_04_tel/05_interaktiv_film) Utolsó letöltés 2023.04.07.
[217] Babylon.js: Video As A Texture. *Github*, 2023.
(https://doc.babylonjs.com/features/featuresDeepDive/materials/using/videoTexture) Utolsó letöltés: 2023.04.01., *továbbá* Three.js: VideoTexture. *Github*, 2023.
(https://threejs.org/docs/#api/en/textures/VideoTexture) Utolsó letöltés: 2023.04.01.
[218] Google LLC: TensorFlow. *Googlesource*, 2023.
(https://www.tensorflow.org/) Utolsó letöltés: 2023.04.07.
[219] Bouaziz Sofien - Valentin Julien: Introducing TensorFlow Graphics: Computer Graphics Meets Deep Learning. *TensorFlow Blog*, 2019. nyár
(https://blog.tensorflow.org/2019/05/introducing-tensorflow-graphics_9.html) Utolsó letötlés: 2023.04.08.

térképes grafikák készítése, a közösségi média integrációja, a különböző grafikai sablonrendszerek kialakítása, valamint virtuális stúdiók és digitális környezetek létrehozása.

Az adatfeldolgozás, a mesterséges intelligencia és a gépi tanulás integrációjának előnyeit kihasználva, a WebGPU technológia lehetővé teszi az automatikus tartalomgenerálást, a felhasználói preferenciák és viselkedés alapján testre szabott grafikai tartalmakat, vagy például mozgóképes hirdetési rendszerek kialakítását. Az adaptív technológia hozzájárul a személyre szabott tartalomgyártás és műsorkészítés fejlődéséhez.

Összefoglalva, a GPU-alapú 3D webvizualizációs eszközök alkalmazása révén, az alkotók olyan látványos és interaktív tartalmakat hozhatnak létre, amelyek újradefiniálhatják a megszokott élő közvetítések megjelenésrendszerét, emellett elősegítheti a broadcast és online streaming platformok megújulását.

8. Jövőkép és előretekintés – egyensúly gép és ember között

Jelen disszertáció keretei között nincs lehetőségem a 3D valósidejű képábrázolás és vizualizációs technológia jövőképének teljeskörű elemzésére. A Doktori Iskolában végzett kutatási tevékenységeim során ugyanakkor több jövőbemutató technológiai vízió vált körvonalazhatóvá, melyek rövid előretekintést indokolnak. Ezek a megújuló, vagy forradalmi eljárások jelentős mértékben átalakíthatják a mozgóképgyártás digitális művészeti területeit. Ugyanakkor, nem szabad elfelejtenünk a mozgókép-technológia fejlődésének ciklikus jellegéről sem.

A plasztikus filmek nézőablakán egybeolvadó sztereoszkóp látványvilág, a holográfia teljes képének térélménye, vagy az adatalapú és a számítógép által írt történetek megtekintése után, a jelenkor legkorszerűbb eszközei talán nem is tűnnek olyan forradalminak. Mindent figyelembe véve, ezen technológiák többsége már régóta a film- és mozgóképgyártás részét képezik. A legújabb vizuális programozási nyelvek alkalmazásai, a valós idejű 3D rendszerek, vagy a kibővített valóság illúziótereiben megjelenő mesterséges intelligencia által generált digitális térélmény valóságreprezentációja is mind azon folyamatosan megújuló formai keretrendszer részei, amely a film mellet, keletkezése óta jelen van. Arra a következtetésre jutottam, hogy ezeket a folyamatokat a mozgóképgyártás integráns részeként kell tehát értelmezni.

Ahogyan a történeti bevezetőben is kifejtettem, ezek a forradalmi lehetőségek leginkább kreatív eszközként hivatottak támogatni a képalkotó szakemberek tevékenységét, és hozzájárulhatnak az egyéni tehetség kibontakozásához. Ugyanakkor napjainkban erős tendencia figyelhető meg az automatizáció és a teljes gépi megoldások felé.

A Dall-E, Midjourney, DeepBrain AI és Synthesia alkalmazások a képi adatokból származó minták elemzésére, átszervezésére vagy átfogó átalakítására képesek. A DeepBrain AI, a Pictory, az InVideo, vagy a Lumen5 automatizált videókészítésre nyújtanak megoldást. Ezek a rendszerek a képi, szöveges és mozgásbeli összefüggések alapján gyűjtik össze az adatokat, ezt követően pedig új tartalmakat generálnak belőlük. Az általam végzett elemzés során arra a következtetésre jutottam, hogy a technológiák megfelelő alkalmazáskörnyezetének definiálásával, kialakítható egyfajta kreatív egyensúly gép és ember között. Habár az automatizált eszközök egyaránt szolgálhatnak inspirációs forrásként és

kivitelezési modellként, sosem pótolhatják az emberi kreativitást és az alkotóművészek egyedi, megismételhetetlen látásmódját.

Megjegyzés

A DLA kutatás során általam fejlesztett rendszer tovább bővíthető olyan korszerű technológiákkal, mint például a TensorFlow. A könyvtár segítségével a gépi tanulási algoritmusokat WebGL vagy WebGPU technológiákkal összekapcsolhatóvá tehetjük. Így a TensorFlow és a grafikus megjelenítő rendszerek együttműködésével automatizált videók készítése válik lehetségessé. A rendszer továbbfejlesztése érdekében integrálhatjuk a mesterséges intelligencia alapú programozást és kódolást támogató eszközöket *(OpenAI ChatGPT, Google Bard, GitHub Copilot, stb)*, amelyek a 3D adatvizualizáció kreatív eszközeit és automatizált folyamatait egyaránt segíthetik.

9. Konklúzió

Kutatásom során arra kerestem a választ, hogy a 3D vizualizációs elemek milyen művészeti módszerekkel és technológiai-stratégiákkal transzplantálhatók a legmodernebb fejlesztői eszközök irányába. Azt elemeztem, hogy mely optimalizációs módszerek alkalmazhatók a 3D tervezőszoftverek és grafikus webes 3D könyvtárak közötti integrációs folyamatban annak érdekében, hogy a térbeli jelenetek minőségveszteség nélküli transzplantációja megvalósítható legyen. Vizsgálatom fókusza az volt, hogy milyen módszerek állnak a mozgóképes szakemberek rendelkezésére, melyek lehetővé teszik a vizualizációs munka platformfüggetlen adaptációját az interaktív 3D környezet irányába.

Az disszertációban bemutatott 3D technológia gyökerei egy évtizedes múltra tekintenek vissza, mégsem állítható, hogy ezek az eljárások forradalmasították a jelenkor térbeli mozgóképalkotás módszerének gyakorlatát. Napjainkig csak számos megkötéssel, kizárólag az interdiszciplináris megközelítés érvelésével és érvényesítésével volt lehetséges a megvalósítás. Éppen ezért nem állítom, hogy az elérhető fejlesztői eszközök akadálymentesen adaptálhatók bármilyen mozgóképes folyamat részeként. Megállapítottam, hogy a klasszikus filmes technológiák csak nehezen ültethetők át a legmodernebbként jellemzett felületekre, számos konverziós nehézség akadályozza, torzítja az elképzelt megjelenés végeredményét. A tervezői szoftverek grafikai képe egyelőre nem minőségveszteség nélkül transzplantálható webalapú engine-ek irányába.

A rendkívül dinamikusan változó programozási környezet megannyi fejlesztési eszköz és eljárásrendszer ismeretét tette elengedhetetlenné. Számos szkriptnyelvi-, szintaktikai megkötés garantálja a böngészőben megjelenő vizuális tartalom leírókörnyezetét. Az elvárásrendszerek a hagyományos 3D tervezői szoftverek publikációs munkafolyamatait is jelentősen befolyásolják. Mindezek ellenére, kutatási munkám során egyértelműen körvonalazódott a web 3D technológia megkerülhetetlen térhódításának indoklása is.

DLA mestermunkámon keresztül megvizsgáltam egy web 3D animációs adatbázis elkészítésének fejlesztői folyamatait, és interaktív publikációsrendszerét. Feltártam a legfontosabb módszereket, és elemeztem a gyakorlati megvalósítás lehetséges lépéseit, irányait. Dokumentáltam a kivitelezéshez szükséges modulokat és funkciókat, elemzéseimmel rávilágítottam a legfontosabb fejlesztői és grafikai szempontokra.

Arra következtetésre jutottam, hogy a web 3D fejlesztői környezet kialakításának módszertanával, és az elemzett jelenetintegrációs módszerekkel igazolható azon kutatói tézisem, mely szerint a WebGPU technológia hozzájárul a broadcast televízió és online streaming 3D adatvizualizációs eszköztárának bővítéséhez, lehetővé téve a platformfüggetlen interaktív tartalmak előállítását és adaptációját. Meggyőződésem, hogy a dolgozatban ismertetett legújabb fejlesztői eszközök és módszertanok integrálása révén a hagyományos 3D tervezőszoftverekből származó jelenetek is hatékonyan adaptálhatók valósidejű adáskörnyezetbe.

Interaktív animációk és videóalkalmazások aspektusából a mozgóképkészítés olyan történeti- és modern vonatkozású szakterületeit is vizsgáltam, melynek részterületei folyamatosan közelítenek az internetes fejlesztési technikákhoz. Ebből adódik, hogy a 3D fejlesztői környezet kialakításának módszerének ismeretanyaga megértése rendkívül lényeges volt a 3D platform kialakításának szempontjából is A mozgóképes szakemberek egyre inkább elengedhetetlen feladata megismerni az interaktív technika működési rendszereit, és a releváns szkriptnyelveket.

A film- és videoművészettől egyre eltávolodó fogalomkörök és ismeretanyagok feltárása számomra -*meglepő módon*- éppen a mozgóképkészítés legújabb eszközeit hozta egyre elérhetőbb távolságba. Ezen tapasztalat alapján azt állítom, hogy a térbeli ábrázolóképesség módszereinek interdiszciplináris megközelítése nélkülözetten a modern multimédiaalkotói számára.

Mindent figyelembe véve, úgy gondolom, hogy sikerült igazolnom a tézist, mely szerint a WebGL és WebGPU technológiák integrációja a broadcast adatvizualizációs sablonrendszerekkel és a webes interaktív alkalmazás-elemekkel új, rugalmasabb és változatosabb vizualizációs paradigmát teremt a televíziós és online streaming csatornák számára. Arra a következtetésre jutottam, hogy a valósidejű 3D web vizualizáció olyan eszközt jelent a modern médiaművészek és fejlesztők számára, amely platformfüggetlen, dinamikusan paraméterezhető, könnyen adaptálható grafikai interfészként értelmezhető a jövőben.

Ezen túlmenően, a 3D WebGPU technológia jelentős hatással van az interaktív mozgóképtartalmak fejlődésére és a broadcast televízió adatvizualizációs lehetőségeinek átalakítására is. Arra az eredményre jutottam, hogy kutatásom záró tézisének állítása is

igazolható, mely szerint a mesterséges intelligencia és gépi tanulás technológiák integrációjával pedig lehetővé válik a személyre szabott műsorelemek készítése, és a mozgóképes tartalmak platformfüggetlen adaptációja.

10. Műalkotás - Doktori kutatáshoz készített alkalmazás

Doktori kutatásom részeként egy komplex 3D webvizualizációs Full-Stack alkalmazást hoztam létre. A megvalósítás során több ezer sor kódot írtam, mely legalább 120 000 karakterrel egészíti ki a disszertáció szövegének terjedelmét. Az műalkotás részeként az adatbázisban elérhetők lesznek a doktori védéshez készített 3D jelenetek is.

Forráskódomat és a 3D kompozíciókat az alábbi Github repository-ban publikáltam: https://github.com/balogharon/3d

11. Függelék

11.1. Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Dr. habil. M. Tóth Gézánk, aki a Doktori Iskolában végzett elméleti és gyakorlati kutatásaim során teljes szakmai támogatásáról és iránymutatásról biztosított. Köszönöm, amiért választott kutatási témám megvalósításában mindvégig mellettem állt.

Köszönetemet fejezem ki a Színház- és Filmművészeti Egyetem Doktori Iskola valamennyi oktatójának, akik a kezdetektől fogva támogattak és bátorítottak a kutatási folyamatban.

Köszönöm a Budapesti Metropolitan Egyetem és a Magyar Televízió munkatársainak, akik végig támogatták doktori tanulmányaim célkitűzéseit.

Végül, de nem utolsó sorban, szeretném megköszönni családom támogatását, akik a doktori képzés során mindvégig mellettem álltak.

SZFE Doktori Szabályzat

13. sz. melléklet

EREDETISÉGI NYILATKOZAT

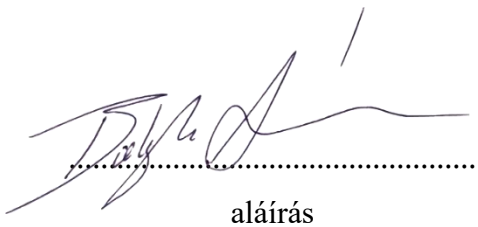
az értekezés benyújtásakor

Alulírott Balogh Áron Doktorandusz büntetőjogi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy a jelen nyilatkozat keletkezését megelőző két éven belül sikertelenül lezárt doktori eljárásom nem volt.

A doktori értekezésem (mind az értekezés szövege, mind a művészeti alkotás) saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt. Tudomásul veszem, hogy plágiumnak számít szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül; tartalmi idézet hivatkozás megjelölése nélkül; más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén doktori értekezésem visszautasításra kerül. Kijelentem továbbá, hogy doktori értekezésem nyomtatott és elektronikus példányai szövegükben, tartalmukban megegyeznek.

Budapest, 2023.04.10.



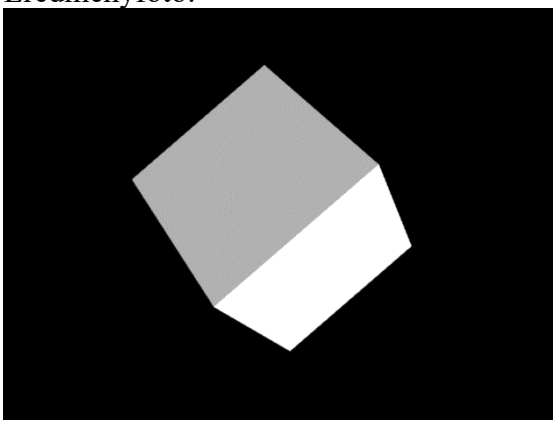
.....

aláírás

11.3. Kódjegyzék

[Script.WebGLCube]
Egyszerű 3D WebGL kocka, forgás animációval

Eredményfotó:



Eredeti forrás: MDN, WebGL: 2D and 3D graphics for the web. (URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API). Utolsó letöltés: 2021.12.06.

Az első forráskód annak a példázata, hogy a kocka alakzat létrehozása menyire bonyolult lehet natív WebGL programozás esetén.

```
declare let window: any;
let cubeRotation: number = 0.0,
    then: number = 0;

export default class WebGLCube {
    constructor() {
        const canvas: any = document.querySelector("#glcanvas"),
            gl =
            canvas?.getContext("WebGL") || canvas.getContext("experimental-WebGL");
        if (!gl) {
            alert(
                "Unable to initialize WebGL. Your browser or machine may not support it."
            );
            return;
        }
        const vsSource = `
attribute vec4 aVertexPosition;
attribute vec4 aVertexColor;
uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
varying lowp vec4 vColor;
void main(void) {
gl_Position = uProjectionMatrix * uModelViewMatrix * aVertexPosition;
vColor = aVertexColor;
```

```
}
`,
        fsSource = `
varying lowp vec4 vColor;
void main(void) {
gl_FragColor = vColor;
}
`,
        shaderProgram = this.initShaderProgram(gl, vsSource, fsSource),
        programInfo = {
          program: shaderProgram,
          attribLocations: {
            vertexPosition: gl.getAttribLocation(
              shaderProgram,
              "aVertexPosition"
            ),
            vertexColor: gl.getAttribLocation(shaderProgram, "aVertexColor"),
          },
          uniformLocations: {
            projectionMatrix: gl.getUniformLocation(
              shaderProgram,
              "uProjectionMatrix"
            ),
            modelViewMatrix: gl.getUniformLocation(
              shaderProgram,
              "uModelViewMatrix"
            ),
          },
        },
        buffers = this.initBuffers(gl);

      const render = (now: number) => {
        now *= 0.001;
        const deltaTime = now - then;
        then = now;
        this.drawScene(gl, programInfo, buffers, deltaTime);
        requestAnimationFrame(render);
      };
      requestAnimationFrame(render);
    }

    initBuffers = (gl: any) => {
      const positionBuffer = gl.createBuffer();
      gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
      const positions = [
        // Elülső felület
        -1.0, -1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0,
        // Hátsó felület
        -1.0, -1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0,
```

```
            // Felső felület
            -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -1.0,
            // Alsó felület
            -1.0, -1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0,
            // Jobb oldali felület
            1.0, -1.0, -1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0, -1.0, 1.0,
            // Bal oldali felület
            -1.0, -1.0, -1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, -1.0, 1.0, -1.0,
        ];
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(positions),
gl.STATIC_DRAW);
        const faceColors = [
            [0.1, 0.1, 0.1, 0.5], // Elülső felület
            [0.2, 0.2, 0.2, 0.5], // Hátsó felület
            [0.3, 0.3, 0.3, 0.5], // Felső felület
            [0.4, 0.4, 0.4, 0.5], // Alsó felület
            [0.5, 0.5, 0.5, 0.5], // Jobb
            [0.6, 0.6, 0.6, 0.5], // Bal
        ];
        var colors: any = [];
        for (var j = 0; j < faceColors.length; ++j) {
            const c = faceColors[j];
            colors = colors.concat(c, c, c, c);
        }
        const colorBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colors),
gl.STATIC_DRAW);
        const indexBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, indexBuffer);
        const indices = [
            0, 1, 2, 0, 2, 3, 4, 5, 6, 4, 6, 7, 8, 9, 10, 8, 10, 11, 12, 13, 14, 12,
            14, 15, 16, 17, 18, 16, 18, 19, 20, 21, 22, 20, 22, 23,
        ];
        gl.bufferData(
            gl.ELEMENT_ARRAY_BUFFER,
            new Uint16Array(indices),
            gl.STATIC_DRAW
        );
        return {
            position: positionBuffer,
            color: colorBuffer,
            indices: indexBuffer,
        };
    };

    drawScene = (gl: any, programInfo: any, buffers: any, deltaTime: any) => {
        gl.clearColor(0.0, 0.0, 0.0, 1.0);
        gl.clearDepth(1.0);
```

```
    gl.enable(gl.DEPTH_TEST);
    gl.depthFunc(gl.LEQUAL);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    const fieldOfView = (45 * Math.PI) / 180,
       aspect = gl.canvas.clientWidth / gl.canvas.clientHeight,
       zNear = 0.1,
       zFar = 100.0,
       projectionMatrix = window.mat4?.create();
    window.mat4.perspective(projectionMatrix, fieldOfView, aspect, zNear, zFar);
    const modelViewMatrix = window.mat4.create();
    window.mat4.translate(modelViewMatrix, modelViewMatrix, [-0.0, 0.0, -6.0]);
    window.mat4.rotate(
       modelViewMatrix,
       modelViewMatrix,
       cubeRotation,
       [0, 0, 1]
    );
    window.mat4.rotate(
       modelViewMatrix,
       modelViewMatrix,
       cubeRotation * 0.7,
       [0, 1, 0]
    ); {
       const numComponents = 3,
          type = gl.FLOAT,
          normalize = false,
          stride = 0,
          offset = 0;
       gl.bindBuffer(gl.ARRAY_BUFFER, buffers.position);
       gl.vertexAttribPointer(
          programInfo.attribLocations.vertexPosition,
          numComponents,
          type,
          normalize,
          stride,
          offset
       );
       gl.enableVertexAttribArray(programInfo.attribLocations.vertexPosition);
    } {
       const numComponents = 4,
          type = gl.FLOAT,
          normalize = false,
          stride = 0,
          offset = 0;
       gl.bindBuffer(gl.ARRAY_BUFFER, buffers.color);
       gl.vertexAttribPointer(
          programInfo.attribLocations.vertexColor,
          numComponents,
          type,
```

```
            normalize,
            stride,
            offset
        );
        gl.enableVertexAttribArray(programInfo.attribLocations.vertexColor);
    }
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, buffers.indices);
    gl.useProgram(programInfo.program);
    gl.uniformMatrix4fv(
        programInfo.uniformLocations.projectionMatrix,
        false,
        projectionMatrix
    );
    gl.uniformMatrix4fv(
        programInfo.uniformLocations.modelViewMatrix,
        false,
        modelViewMatrix
    );

    {
        const vertexCount = 36,
            type = gl.UNSIGNED_SHORT,
            offset = 0;
        gl.drawElements(gl.TRIANGLES, vertexCount, type, offset);
    }
    cubeRotation += deltaTime;
};

initShaderProgram = (gl: any, vsSource: any, fsSource: any) => {
    const vertexShader = this.loadShader(gl, gl.VERTEX_SHADER, vsSource),
        fragmentShader = this.loadShader(gl, gl.FRAGMENT_SHADER, fsSource),
        shaderProgram = gl.createProgram();
    gl.attachShader(shaderProgram, vertexShader);
    gl.attachShader(shaderProgram, fragmentShader);
    gl.linkProgram(shaderProgram);
    if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
        alert(
            "Unable to initialize the shader program: " +
            gl.getProgramInfoLog(shaderProgram)
        );
        return null;
    }
    return shaderProgram;
};

loadShader = (gl: any, type: any, source: any) => {
    const shader = gl.createShader(type);
    gl.shaderSource(shader, source);
    gl.compileShader(shader);
```

```
        if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
            alert(
                "An error occurred compiling the shaders: " +
                gl.getShaderInfoLog(shader)
            );
            gl.deleteShader(shader);
            return null;
        }
        return shader;
    };
}
```

[Script 2.]
WebGPU token kód, melyet a WebGPU tesztelési időszakában kellett használnom.

```
<meta http-equiv="origin-trial"
content="AjfDXI8sBo8wbkUgbWi/LG14H4zMZt8pdEs18Cprk3SQhqVz0w5b0V4chBvDew
mAsFFfp0gsX+aMzWeGiLcuOQAAAABJeyJvcmlna.....==">
```

11.4. Ábrajegyzék

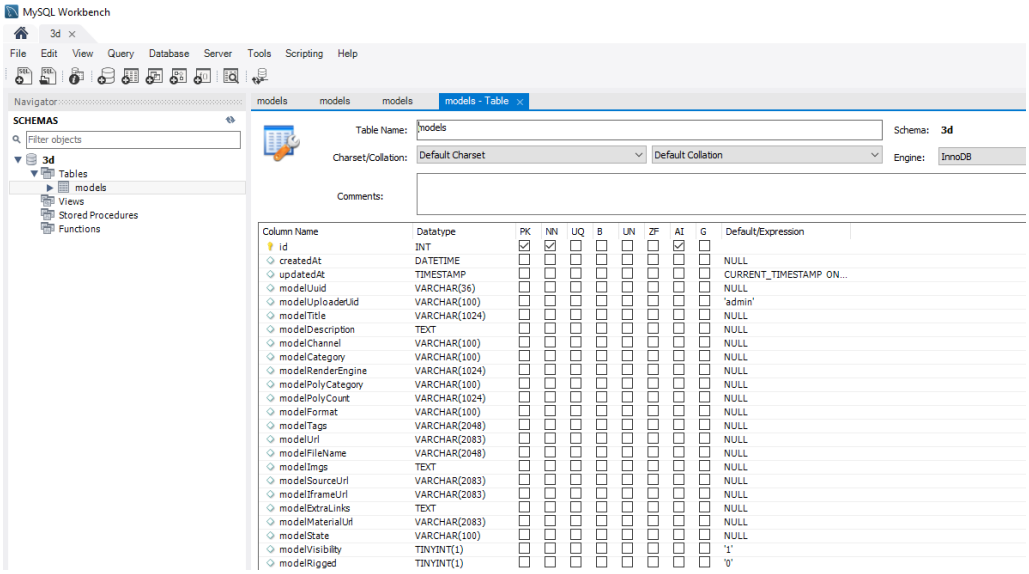
[Fig.Package] A package.json fájl alapverziója. Forrás: Balogh Áron, 2023.

```
{
  "compilerOptions": {
    "target": "es5",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "noFallthroughCasesInSwitch": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx"
  },
  "include": [
    "src"
  ]
}
```

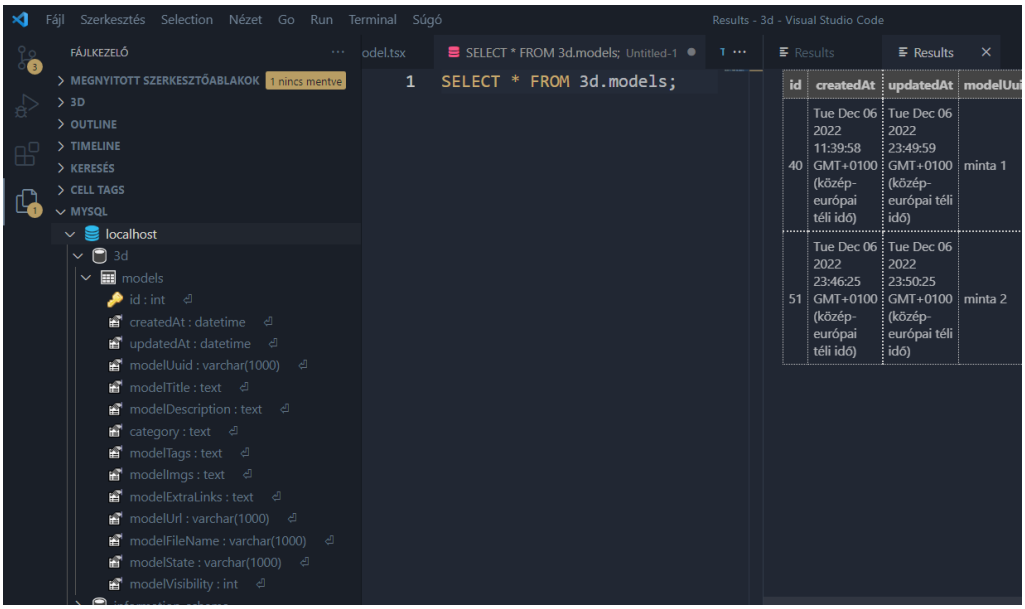
[Fig.TsConfig] A tsconfig.json fájl alapverziója. Forrás: Balogh Áron, 2023.

```
"env": {
  "browser": true,
  "es2021": true
},
"extends": ["plugin:react/recommended", "standard-with-typescript", "plugin:@typescript-eslint/recommended", "plugin:@typescript-eslint/recc
"overrides": [{ "files": ["*.ts", "*.tsx"] }], // added later { "files": ["*.ts", "*.tsx"] }
"parser": "@typescript-eslint/parser",
"parserOptions": {
  "project": ["tsconfig.json"],
  "ecmaVersion": "latest",
  "sourceType": "module"
},
"plugins": ["react", "@typescript-eslint"], // added later "@typescript-eslint"
"rules": {
  "semi": ["off"],
  "@typescript-eslint/semi": "off",      You, előző hónap • - …
  "no-unexpected-multiline": "error",
  "space-before-function-paren": "off",
  "@typescript-eslint/space-before-function-paren": "off",
  "@typescript-eslint/explicit-function-return-type": "off",
  "@typescript-eslint/no-floating-promises": "off",
  "@typescript-eslint/strict-boolean-expressions": "off"
}
```

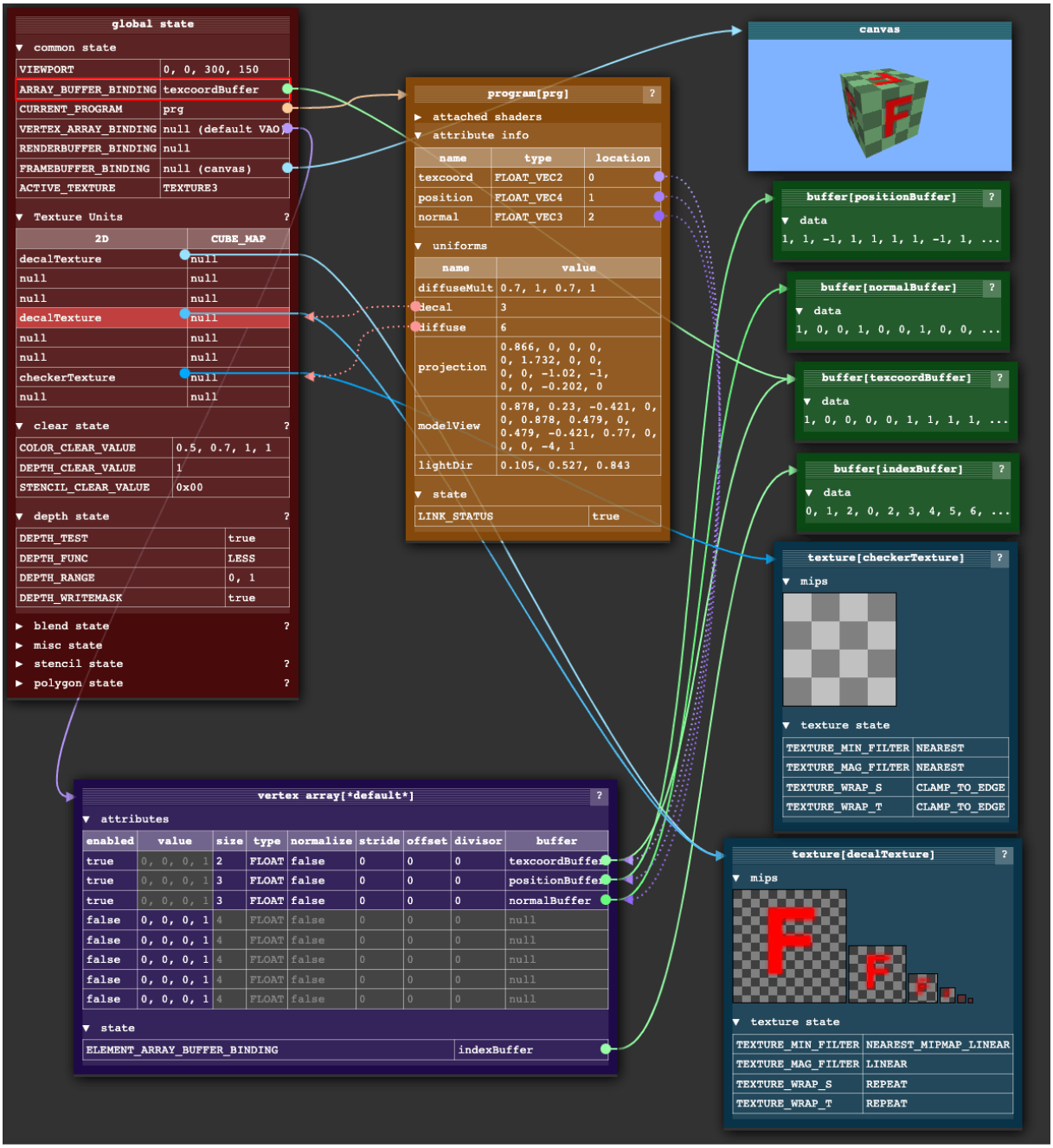
[Fig.Eslint] ESLint konfigurációs fájl. Forrás: Balogh Áron, 2023.



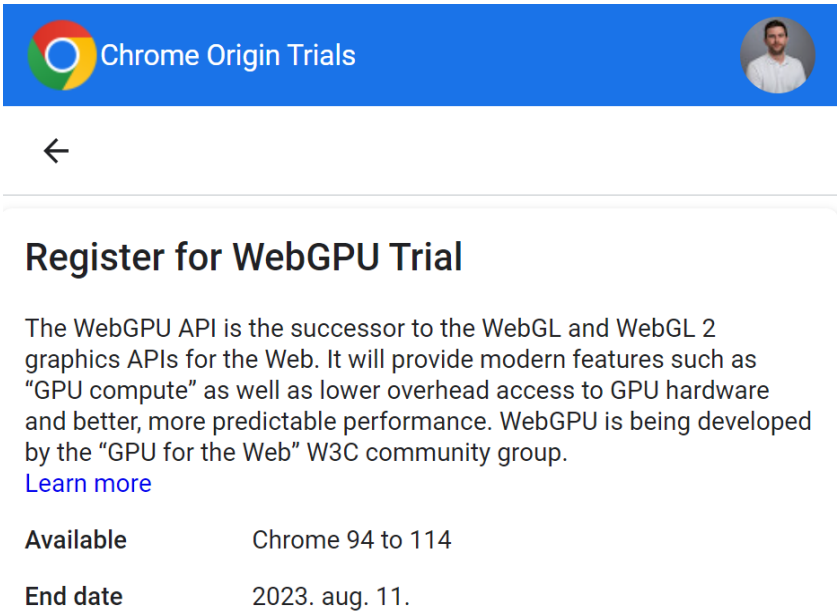
[Fig. MySqlWorkB] A MySQL Workbench kezelőfelülete. Forrás: Balogh Áron, 2023.



[Fig.VsMysql] A VsCode beépülő modulja (MySQL management tool). Forrás: Balogh Áron, 2023.

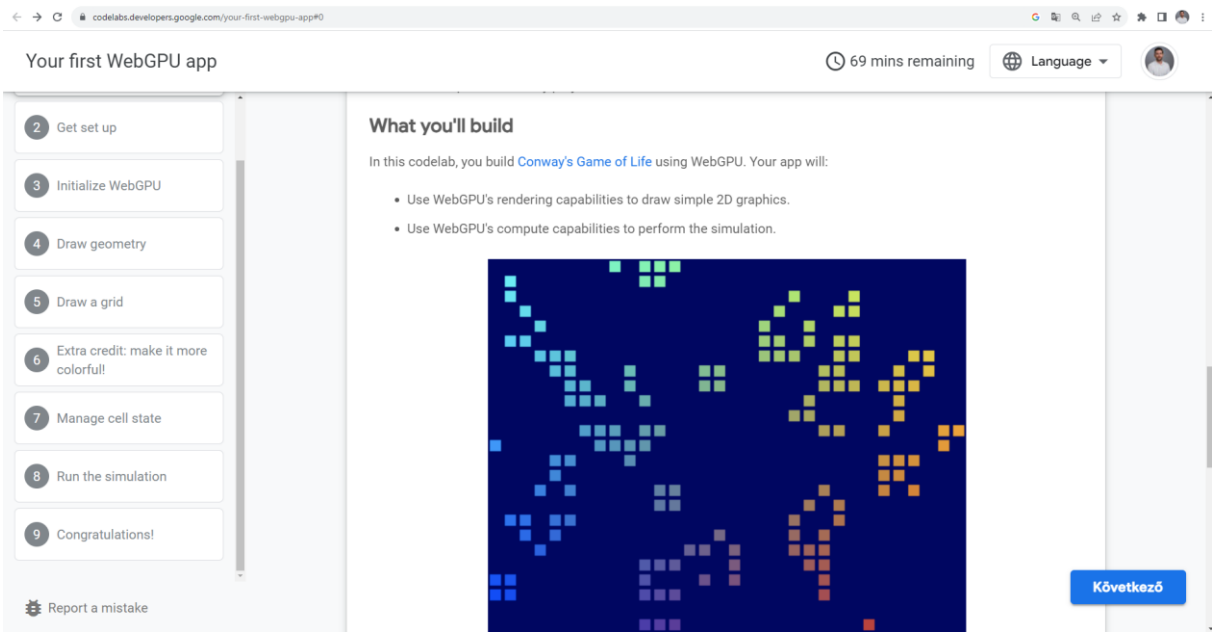


[Fig.WebGK_internalState] Kockafraktál WebGPU környezetben. Forrás: Bynens , Mathias: WebGPU — All of the cores, none of the canvas. Surma.dev, 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.

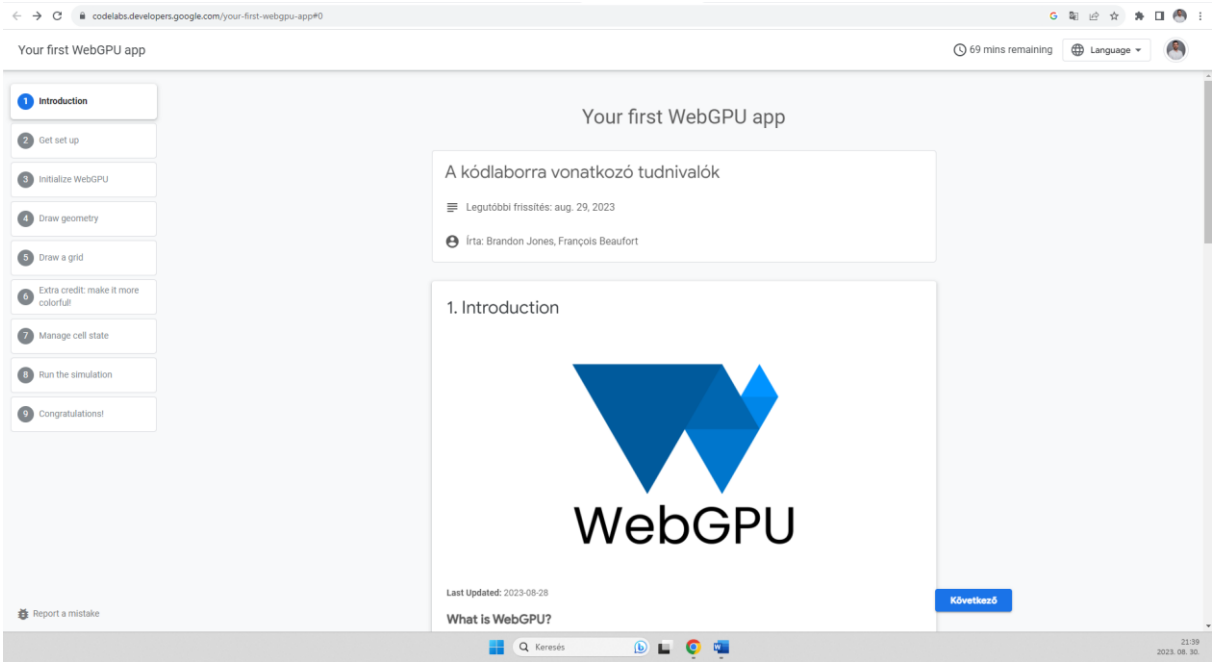


7

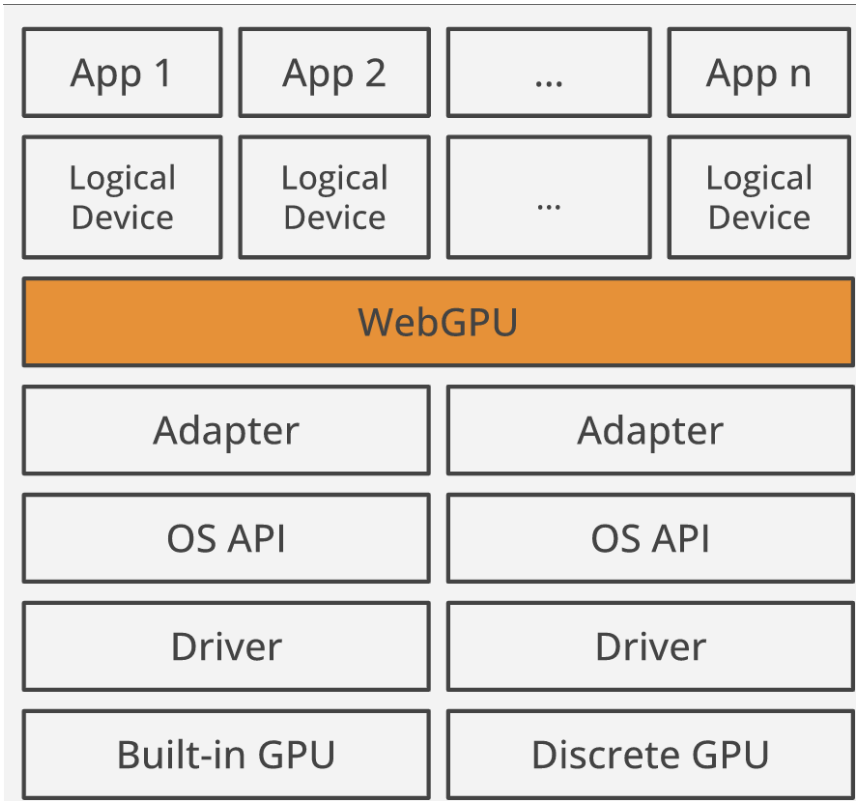
[Fig.Token1] WebGPU regisztrációs felület. Forrás: Balogh Áron, 2023.



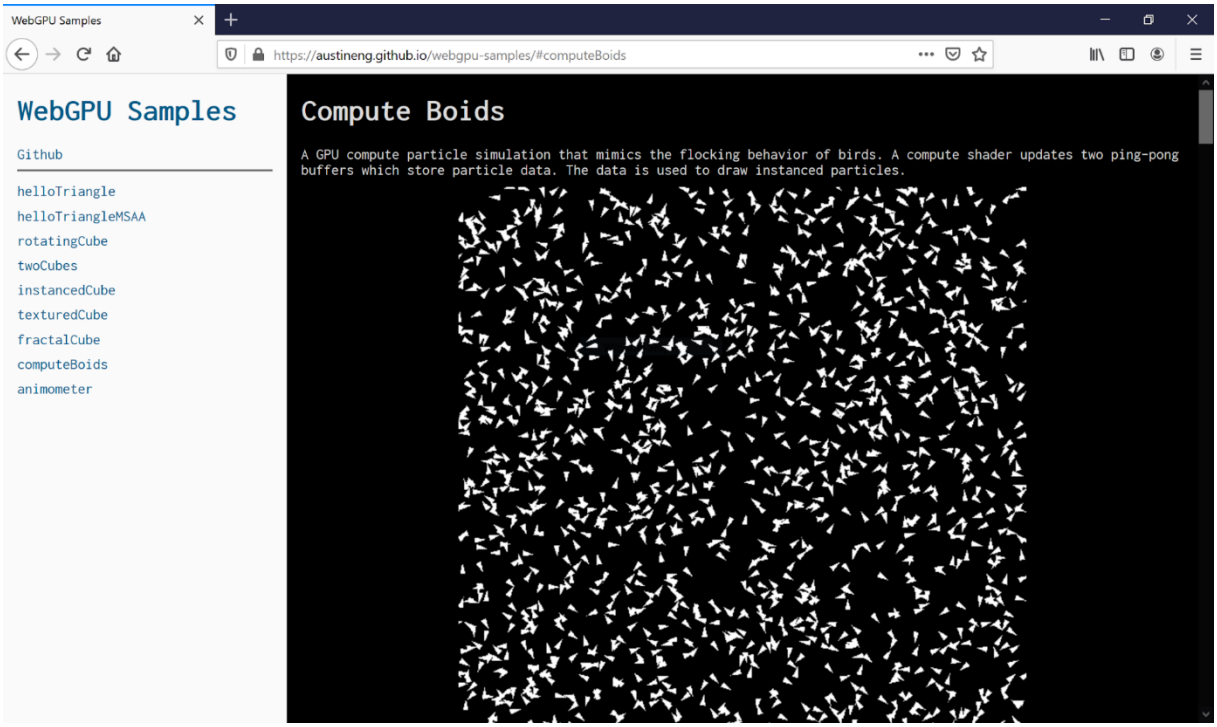
[Fig.WebGPU app 1.] A Google legelső WebGPU oktatóanyaga, mely John Conway, Életjátéka (1970) című alkotásának újraprogramozására biztatja a kísérletező fejlesztőket. Forrás: Google Developers, 2023.]



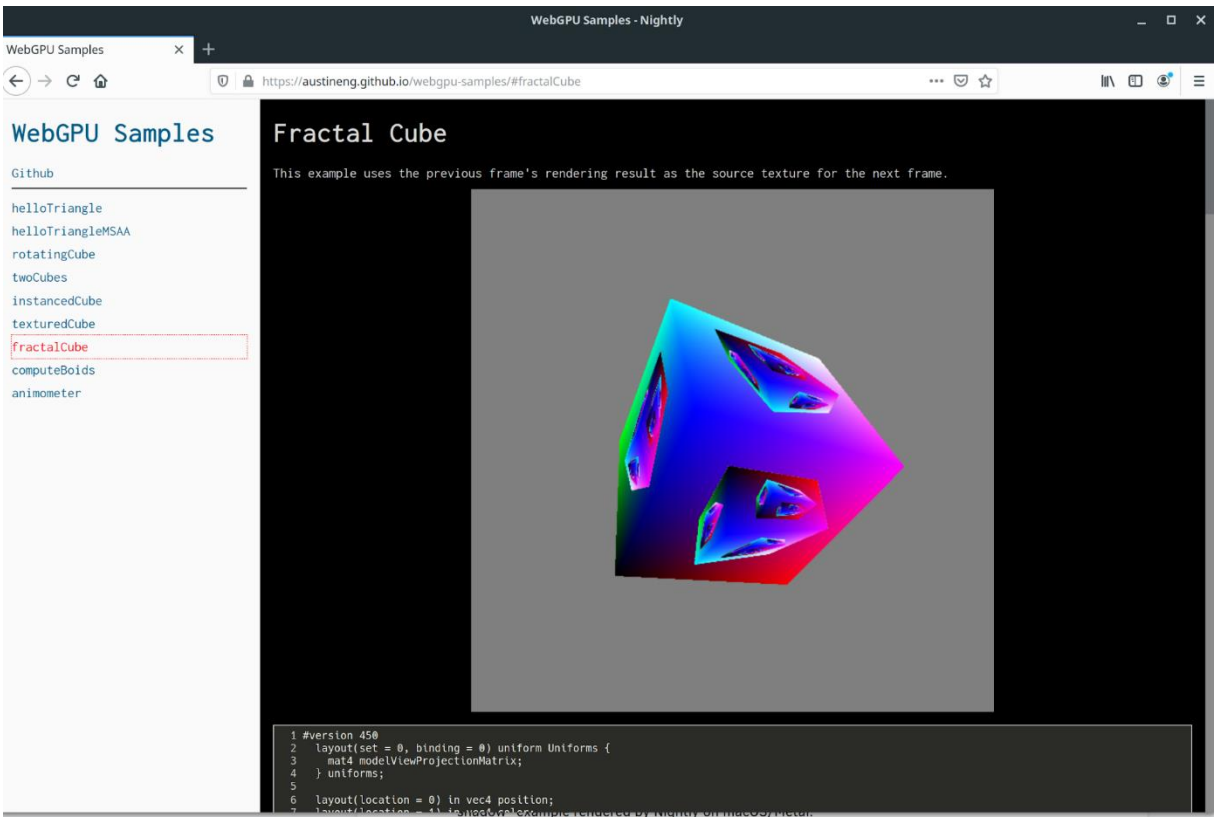
[Fig.WebGPU app 2. Az oktatóanyagot a publikus védésre szánt disszertáció leadási határideje előtt két nappal, 2023. 08. 29. tették mindenki számára elérhetővé. Forrás: Google Developers, 2023.08.29]



[Fig.WebGPUAbstraction] WebGPU absztrakciós rétegek, a GPU-kól a logikai eszközökig. Forrás: Bynens, Mathias: WebGPU — All of the cores, none of the canvas. Surma.dev, 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.



[Fig.WebGPU_particle1] Részecskeszimuláció látványképe. Forrás: Dzmitry Malyshau: A Taste of WebGPU in Firefox. Mozilla.org, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.

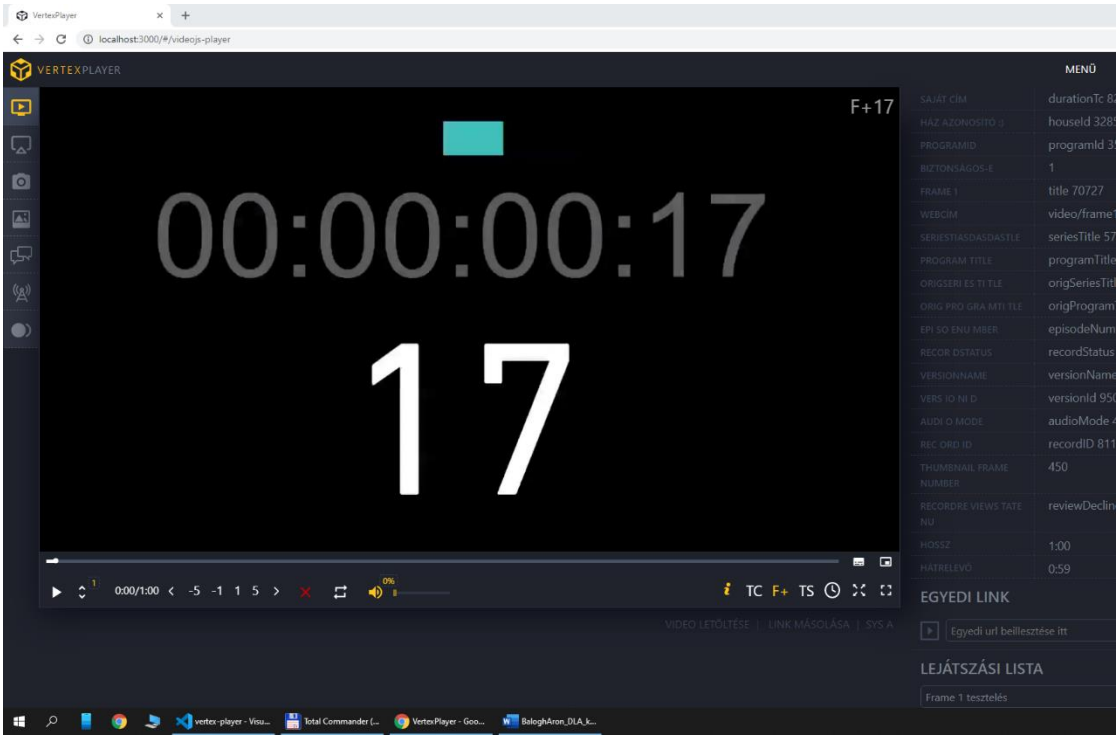


[Fig.WebGPU_cube1] Kockafraktál WebGPU környezetben. Forrás: Dzmitry Malyshau: A Taste of WebGPU in Firefox. Mozilla.org, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.

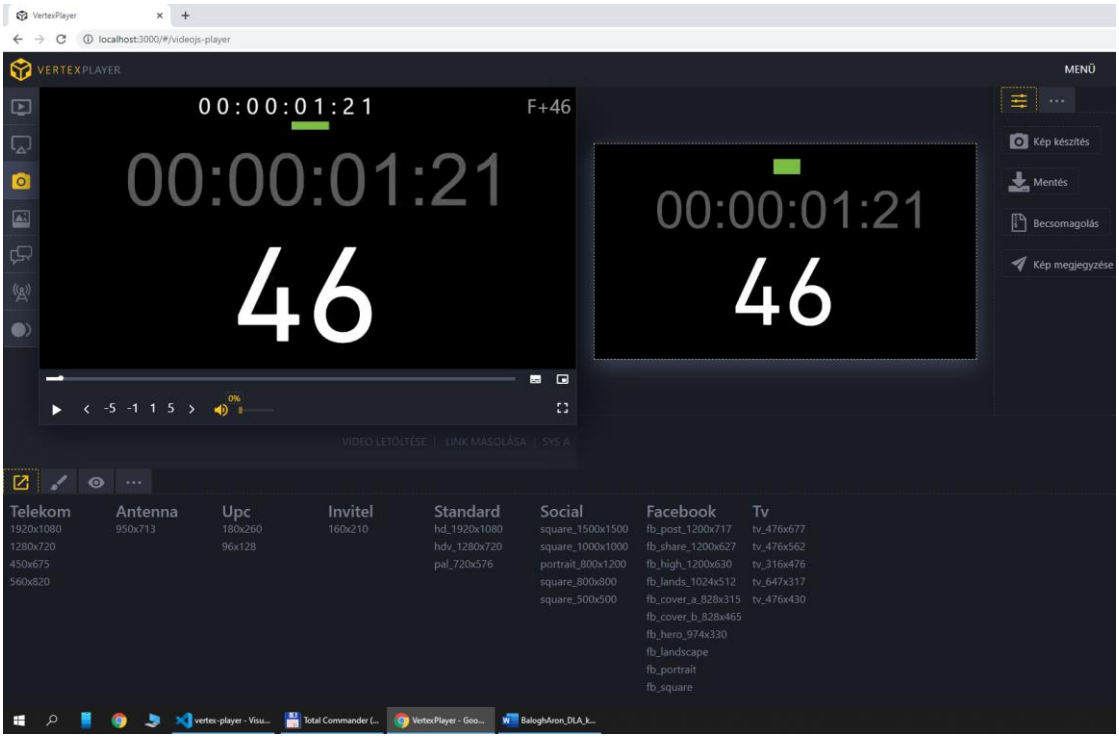
[Fig.WebGPU_Shades1] Shading WebGPU környezetben. Forrás: Seguin Damien: Graphics on the Web and Beyond with WebGPU. Medium, 2020. nyár (https://dmnsgn.medium.com/graphics-on-the-web-and-beyond-with-webgpu-13c4ba049039) Utolsó letöltés: 2023.04.02.

***VertexPlayer** – online megtekintő rendszer*

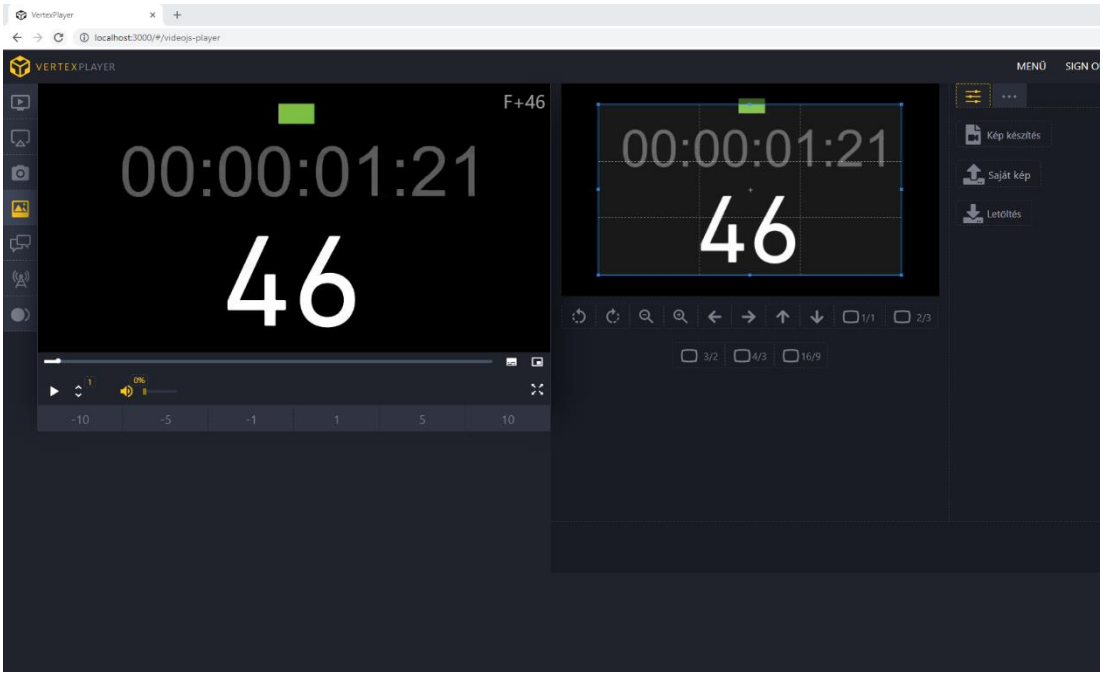
A DLA disszertációra készített teljesen egyedi fejlesztésű multimédia lejátszóm, mely 3D tartalmak megjelenítése is képes.



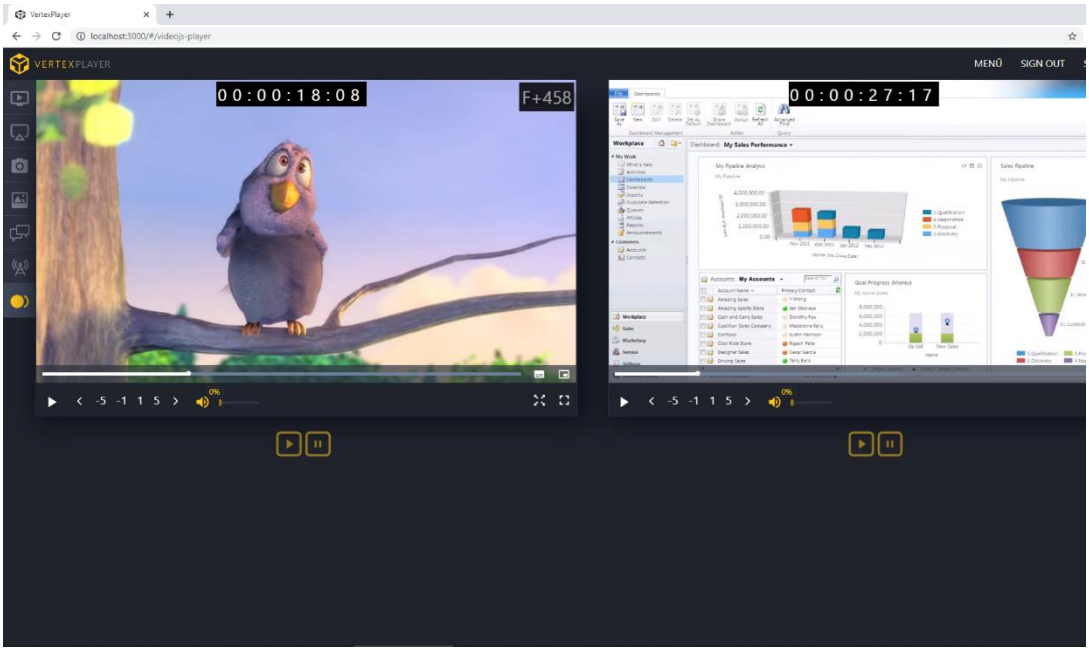
[Fig.Vtx1] VertexPlayer - Videómegtekintő, képkocka megjelenítéssel. Forrás: Balogh Áron, 2023.



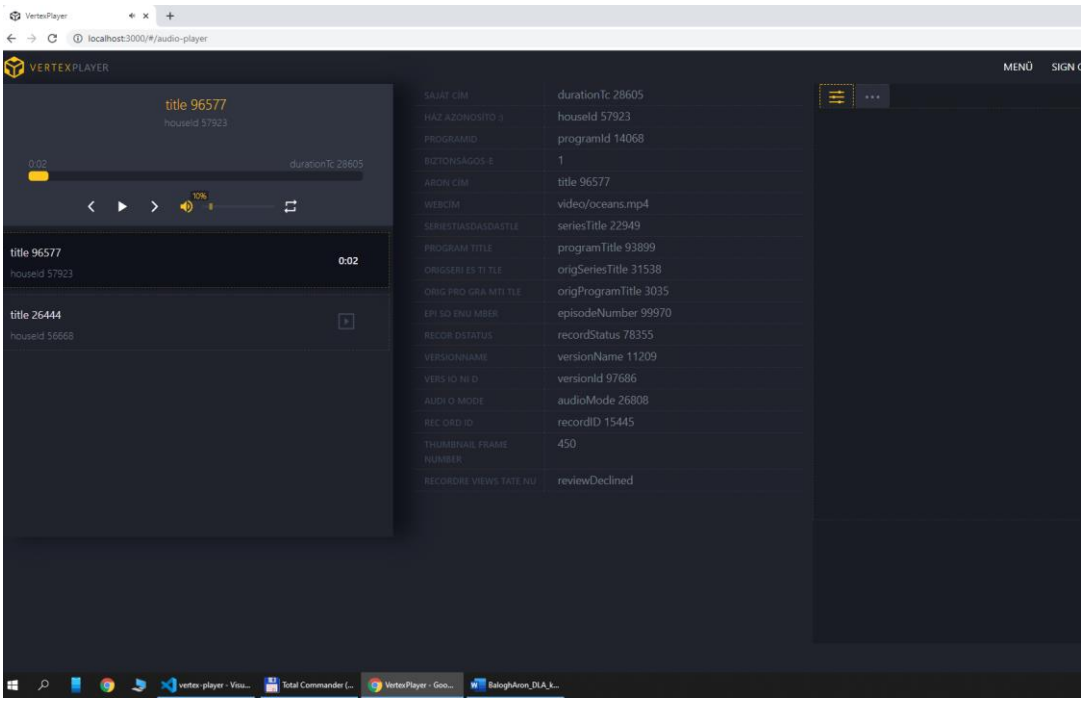
[Fig.Vtx2] VertexPlayer - Pillanatkép készítés videóból. Forrás: Balogh Áron, 2023.



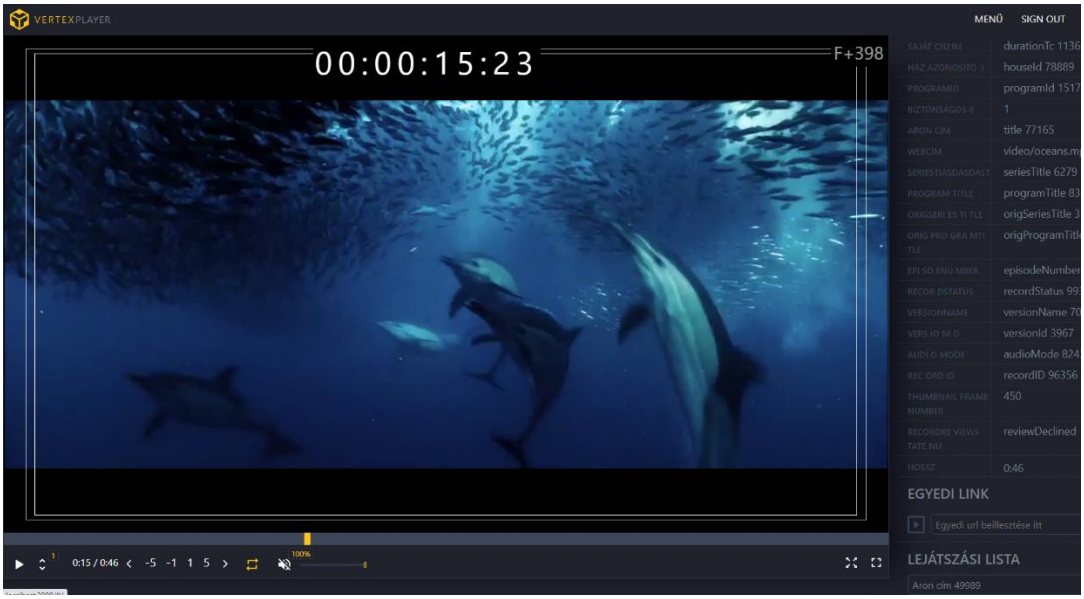
[Fig.Vtx3] VertexPlayer - Pillanatkép editálása. Forrás: Balogh Áron, 2023.



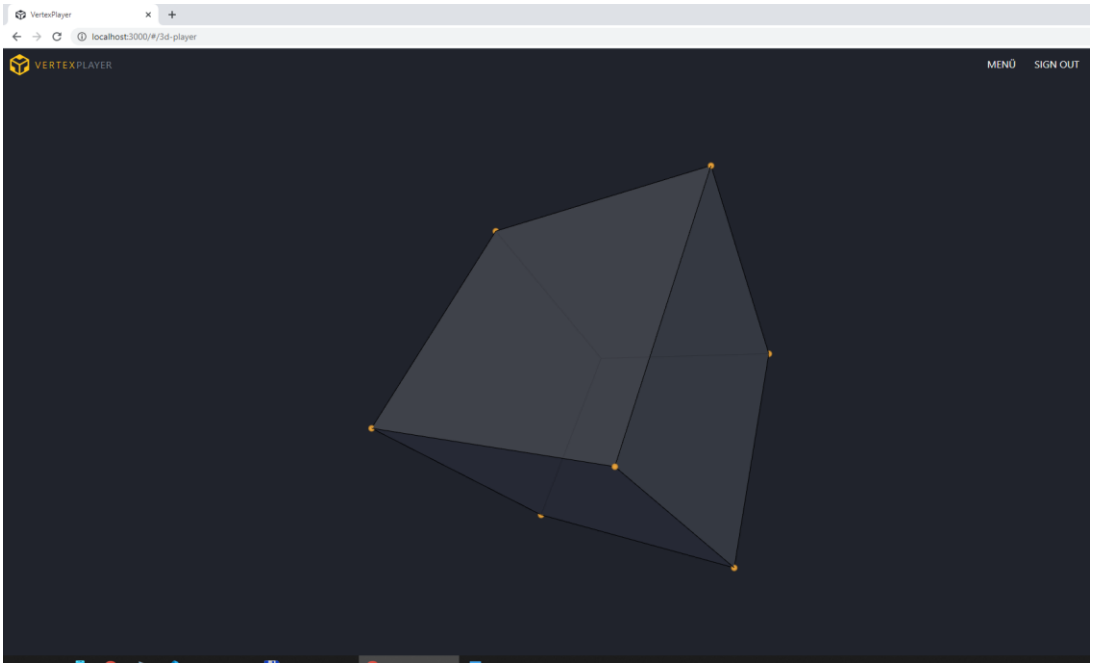
[Fig.Vtx4] VertexPlayer - Multivewer megtekintőrendszer. Forrás: Balogh Áron, 2023.



[Fig.Vtx5] VertexPlayer - Audio rendszer. Forrás: Balogh Áron, 2023.

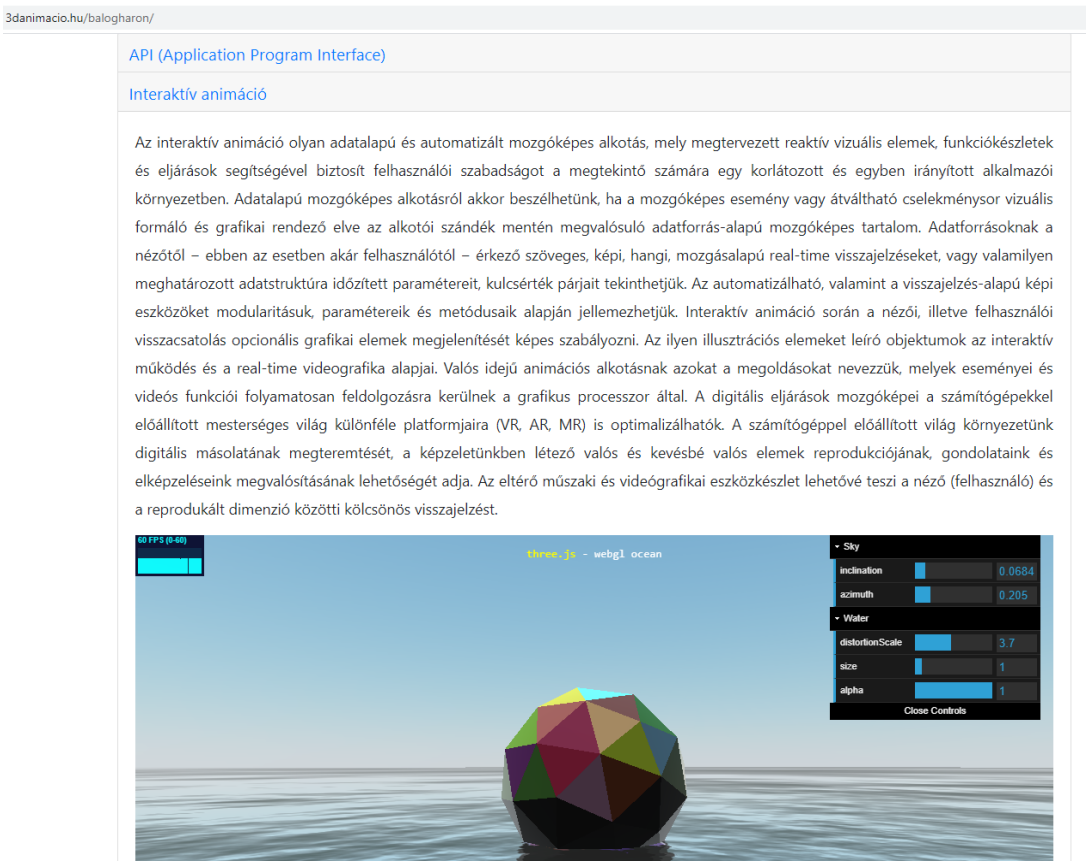


[Fig.Vtx6] VertexPlayer - Információs panel. Forrás: Balogh Áron, 2023.



[Fig.Vtx7] VertexPlayer - Interaktív 3D képmegjelenítő. Forrás: Balogh Áron, 2023.

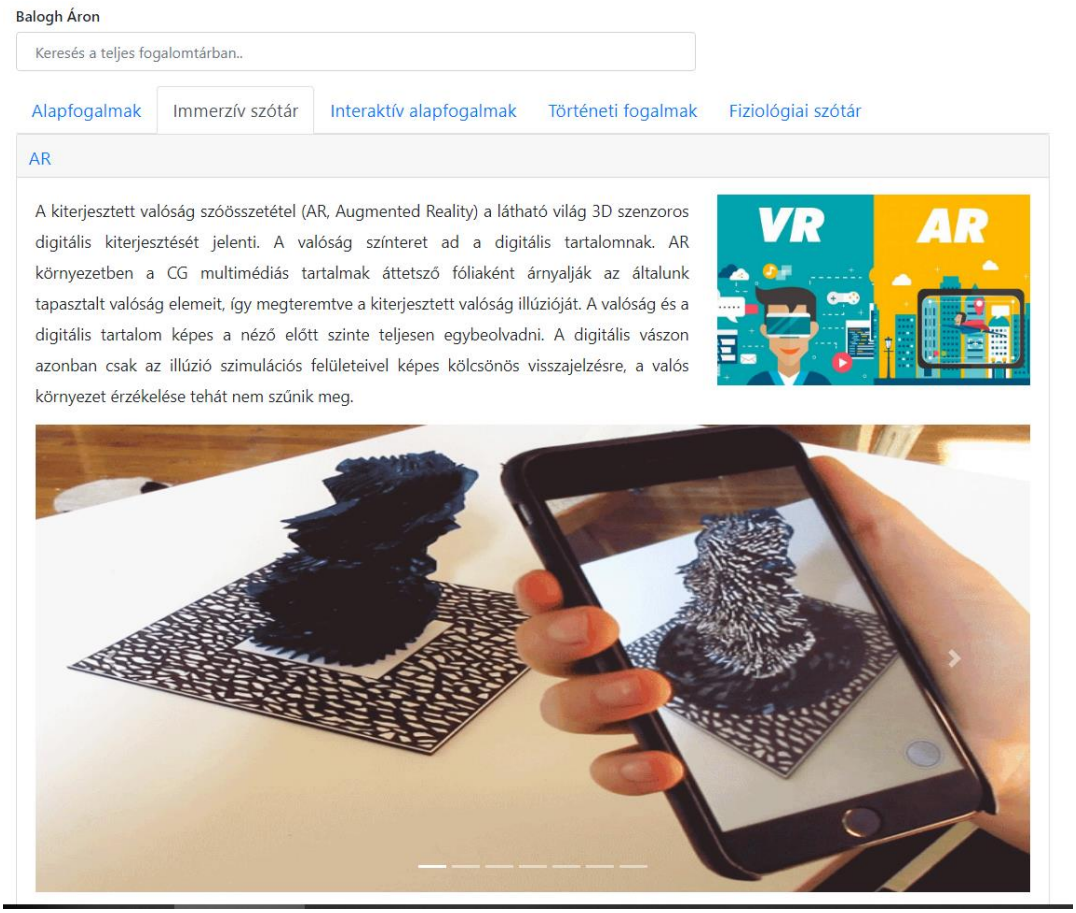
[Fig.Fogalomtart1] Saját fejlesztésű, kereshető fogalomtár. Forrás: Balogh Áron, 2023.



[Fig.Fogalomtart2] Saját fejlesztésű, kereshető fogalomtár. Forrás: Balogh Áron, 2023.

[Fig.Fogalomtart3] Saját fejlesztésű, kereshető fogalomtár. Forrás: Balogh Áron, 2023.

[Fig.Fogalomtart4] Saját fejlesztésű, kereshető fogalomtár. Forrás: Balogh Áron, 2023.



[Fig.Fogalomtart5] Saját fejlesztésű, kereshető fogalomtár. Forrás: Balogh Áron, 2023.

12. Glosszárium

3D

A háromdimenziós, vektorgrafikus objektumok olyan síkból kiemelkedő entitások, amelyek három koordinátával (x, y, z) rendelkeznek. A 3D animáció területén ez a kifejezés utal a térbeli tervezőművészeti munkafolyamatokra és az ebben a kontextusban létrehozott grafikai illusztrációkra. Fontos megjegyezni, hogy a jelen szócikkben tárgyalt 3D fogalom és a kétdimenziós képekből készített sztereoszkópikus 3D vizualizáció eltérő jelentéseket hordoz, bár gyakran (tévesen) azonos jelöléssel illetik őket. A 3D kifejezés tehát egyrészt utalhat a 3D modellezéssel létrehozott grafikai ábrázolásokra, másrészt - kissé félrevezető módon - a moziélményként ismert 3D (szemüveges sztereoszkóp) képhatásra is.

3D animáció

A 3D animáció egy technikai eljárás, amely a háromdimenziós modellek mozgatását és megjelenítését teszi lehetővé. A 3D animációk létrehozása összetett folyamat, amely több lépést is magában foglal, mint például a 3D modellek létrehozása, a karakterek (csont-) vázrendszerének (rigging) kialakítása, az animációs szekvenciák megtervezése, valamint a szükséges fények, kameramozgások és kamerapozíciók definiálása.

3D polarizáció

A 3D polarizáció passzív sztereó technika, amely speciális projektorokat és vetítővásznakat használ a térbeli képmegjelenítéshez. A technika a két szem számára eltérő tartalmat vetít, amit polarizációs szemüvegek transzformálnak a megfelelő szemekhez.

Alpha csatorna

Egy képpont RGB színösszetevők esetén egyenként 8 biten ábrázolhatók (32 bit részletesség esetén). A fennmaradó 8 bit az alfa csatorna, mely az átlátszóságra, transzparenciára vonatkozó információkat tartalmazza.

Aktív szemüveges technika

Sztereoszkóp technika, mely során a projektor 120 Hz-es képfrissítés mellett felváltva jeleníti meg a képet (60-60 képkocka) a két szem számára. A néző aktív szemüvegében a monokróm LCD kijelző, a shutter, rövid időre elzárja a fény útját.

Aliassing

A számítógépes monitoron előállított kép legkisebb egysége a pixel. A háromdimenziós térben azonban létezhet a pixelnél kisebb elem, amit a monitor nem tud kirajzolni. A leírt problémára az anti-aliassing eljárás jelent megoldást, mely az egyes pixelek vagy poligonok, valamint élek és vertexek újraszámításáról, utólagos manipulációjáról szól.

Anaglif kép

Sztereoszkóp képtechnika, mely az eltérő (vörös és cián, vagy sárga és kék, zöld és bíbor) színszűrők és lencsék használatán alapszik. Ugyanazon kép egyidejűleg tartalmazza a szemek számára eltérő képi információt. A képtér eltérő színezete mélységérzetet teremt.

Anyagszerkezet

A háromdimenziós formákat, színek helyett anyagtulajdonságaik segítségével határozzuk meg. Az anyag kifejezés azon tulajdonságokra vonatkozik, amelyek meghatározzák egy felület vizuális megjelenését (fényvisszaverődés, a fényáteresztő képesség és a felületi színezet), továbbá meghatározza, hogy hogyan lép kölcsönhatásba a felületet érő fényhatásokkal. A 3D anyagszerkezet a 3D objektumok textúráját és megjelenését határozza meg. Megfelelő anyagszerkezet beállításával a 3D objektumoknak teljesen valósághű megjelenést is adhatunk.

API (Application Program Interface)

Az angol rövidítés alkalmazásprogramozási felületet jelent, ami egyfajta technikai kommunikációs interfész szoftverek és/vagy eszközök között. Broadcast fejlesztői környezetben például az animációs template-ek, illetve az adatalapú grafikák kapcsolódhatnak a vezérlőszoftver API-jához.

AR (Augmented Reality)

A ’kiterjesztett valóság’ szóösszetétel a látható világ 3D szenzoros digitális kiterjesztését jelenti. A valóság színteret ad a digitális tartalomnak.

Back-end stack

Szerveroldali fejlesztői eszközök, melyek a kliensoldali futtatókörnyezetet biztosítják

Binokuláris diszparitás
A binokuláris fúzió során az agy a két szem által látott képet egybeolvasztja. A szemek eltérő pozíciója miatt létrejön a parallaxis effektus, mely a pontos mélységészlelést definiálja.

Blender
Nyílt forráskódú 3D grafikai program.

Bootstrap
A világ egyik legismertebb, nyílt forráskódú HTML-, JavaScript -, CSS keretrendszere.
(JS) Build eszközök
JavaScript könyvtárak, programkódok telepítési folyamatához.

Bump mapping
3D tervezői eszköz, mely a felület fényvisszaverődési szögét szabályozza az egyenetlenségi térkép segítségével. A bump map segítségével a felületek beesési szöge eltér a visszaverődési szögtől, így a felület egyenetlenségének látszatát valósíthatjuk meg.

CAD (Computer-Aided Design)
Mérnöki 3D tervező szoftverek

Canvas
A böngésző vászon eleme, amely grafikus-tárolóként a vizuális képi elemek megjelenítését biztosítja.

CG (Computer Graphics)
számítógépes grafika. Általában 3D-s környezetben használjuk.

CGI (Computer Generated Imaginary) számítógépes grafikát, számítógéppel létrehozott 3D alkotást jelent. A kifejezés olyan képi környezetre utal, melyet háromdimenziós tervezőszoftverek alkalmazásával (többnyire képszámítás segítségével) állítanak elő. Egyaránt jelenthet álló- és mozgóképtartalmat, továbbá a kiterjesztett valóság interaktív színterein is utal az előre beágyazott animációs tartalmakra.

CLI (Command Line Interface)

Parancssori program

CORS (Cross-Origin Resource Sharing)
Olyan mechanizmus a webböngészőkben, amely engedélyezi, vagy korlátozza egy másik weboldal, illtetve alkalmazás számára a hozzáférést az erőforrásaihoz.

CPU (Central processing unit)
A számítógép központi feldolgozó egysége.

CRA (Create React App)
Előzetes konfiguráció nélkül használható integrált React fejlesztői eszközlánc.

CRUD
A Create, Read, Update, Delete metódusok rövidítése.

CSS (Cascading Style Sheets) Egymásba ágyazott stíluslapok.

Edge
Két vertexet összekötő szakasz.

ESLint
A forráskód szintaktikai elemzését az ESLint bővítmény biztosítja. A szintaktikai ellenőrzési módszerek definiálása a .eslitnrc.js(on) fájl segítségével történik.

Exporter modul
A digitális jelenet paraméterezhető konverzióját, tömörítését és mentését leíró folyamat.

Face
Legalább három vertexből álló felület.

Front-End fejlesztő
A kliensoldali fejlesztésekért *(pl. felhasználói felületek, vizuális megjelenítés)* felelős szakember.

Front-End Stack
Kliensoldali fejlesztői eszközlista, amely a kódíráshoz szükséges programokat, platformokat és kiegészítőket jelenti.

Full-Stack
Komplett alkalmazás, amely Front-end (kliens-) és Back-end (szerver-) oldali összetevőkből áll

Git
Nyílt forráskódú verziókezelő rendszer.

GitHub
A forráskódok tárolására és azok verziókezelésére használt online platform.

Git repo
Repo-k, más néven git tárolók, melyek egy szoftver életciklusának verzió-elemeit lokális adatbázisban tárolják.

Gitignore fájl
A git-ben nem verziókezelt, azaz nem nyomon követett elemek szöveges listája.

GUI (Graphical User Interface)
Grafikus felhasználói felület.

GPU (Graphics processing unit)
A GPU (Graphics Processing Unit) a grafikus vezérlőkártya központi egysége, amely a grafikai tervezés és megjelenítés magasszintű feladatait végzi el. Az ilyen processzorok optimalizálva vannak a képek és 3D objektumok gyors feldolgozására és renderelésére.

HTML (Hypertext Markup Language)
A Hypertext Markup Language (HTML) olyan jelölőnyelv, amelyet a weboldalak és az internetes tartalmak megjelenítésére használnak.

IDL (Interface Definition Language)

interfészleíró nyelv, melyet a WebGPU IDL szócikként részletezek.

Interaktív animáció

Az interaktív animáció olyan adatalapú és automatizált mozgóképes alkotás, mely megtervezett reaktív vizuális elemek, funkciókészletek és eljárások segítségével biztosít felhasználói szabadságot a megtekintő számára egy korlátozott és egyben irányított alkalmazói környezetben. Adatalapú mozgóképes alkotásról akkor beszélhetünk, ha a mozgóképes esemény vagy átváltható cselekménysor vizuális formáló és grafikai rendező elve az alkotói szándék mentén megvalósuló adatforrás-alapú mozgóképes tartalom. Adatforrásoknak a nézőtől − ebben az esetben akár felhasználótól − érkező szöveges, képi, hangi, mozgásalapú real-time visszajelzéseket, vagy valamilyen meghatározott adatstruktúra időzített paramétereit, kulcs-érték párjait tekinthetjük. Az automatizálható, valamint a visszajelzés-alapú képi eszközöket modularitásuk, paramétereik és metódusaik alapján jellemezhetjük. Interaktív animáció során a nézői, illetve felhasználói visszacsatolás opcionális grafikai elemek megjelenítését képes szabályozni. Az ilyen illusztrációs elemeket leíró objektumok az interaktív működés és a real-time videografika alapjai. Valós idejű animációs alkotásnak azokat a megoldásokat nevezzük, melyek eseményei és videós funkciói folyamatosan feldolgozásra kerülnek a grafikus processzor által. A digitális eljárások mozgóképei a számítógépekkel előállított mesterséges világ különféle platformjaira (VR, AR, MR) is optimalizálhatók. A számítógéppel előállított világ környezetünk digitális másolatának megteremtését, a képzeletünkben létező valós és kevésbé valós elemek reprodukciójának, gondolataink és elképzeléseink megvalósításának lehetőségét adja. Az eltérő műszaki és videógrafikai eszközkészlet lehetővé teszi a néző (felhasználó) és a reprodukált dimenzió közötti kölcsönös visszajelzést.

JavaScript

Prototípus alapú, interpretált szkript- vagy parancsnyelv, amely a böngészőkön, és webszervereken fut.

JSDoc

Olyan dokumentációs eszköz, amely segítségével JavaScript kódot lehet dokumentálni.

JSX

Beágyazható XML-jellegű JavaScript szintaxis.

Kulcskocka

A kulcskockák az animációs (vagy mozgóképes) folyamat során egy mozdulatsor meghatározott időkeretének kiinduló- és végpontjait jelölik. A kulcskockák segítségével az animációk irányítása valósítható meg. A jelölőeszköz előre definiált paraméterek kulcs-érték párjaiban rögzíti a változást a megformált képi idő adott pontjaiban. Egy grafikai idősíkon (időszalagon) elhelyezett önálló kulcskocka csak akkor jelent animációs értékváltozást, amennyiben ezt valamilyen szkript lefutása biztosítja. (Ezt a jelenséget kulcskocka nélküli animációnak is nevezhetjük.)

Localhost

Saját szerver. Olyan fejlesztői környezet, melyen a böngésző (kliens), és a webkiszolgáló (szerver) egy időben fut, és a köztes kommunikáció ugyanezen a gépen történik.

LOD (Level of detail)

azaz a textúrák részletességének értéke. A kameratávolságtól számított textúrák méretét a mipmap eljárás határozza meg. A módszer által generált szintváltásokat a LOD értéke definiálja.

Mixin

Újrahasználható stíluselemek, gyűjteménye.

Modellezés

A térbeli tervező és animációs szoftverek jelentősen leegyszerűsítik a 3D képalkotást. A testek drótváza (wireframe) úgy jön létre, hogy a rajzoló program a három koordináta adott pontjai között vektorokat húz. Számos módon hozhatunk létre rendkívül komplex térbeli alkotásokat. 3D modellezés során a legegyszerűbb geometriai formákból indulunk ki, ezeket szerkesztjük, pozícionáljuk a háromdimenziós térben. A legkisebb építőelemeket (pontok, vonalszakaszok, háromszögek) együttes nevükön geometriai primitíveknek nevezzük. Az összetett jelenetek általában ezekből az egyszerű grafikai komponensekből állnak össze.

MR (Mixed Reality)

A ’kevert valóság’ a kiterjesztett valóság definíciójához képest jelent változást. A létező tartalom és a szintetikus trükk képes visszahatni egymásra, vagyis interaktívan kapcsolódni tud eredeti forráskörnyezetének elemeihez. A valóság és a virtuálisan megalkotott tartalom bonyolultabb interakcióira is képes, elmosva a virtuális- és valós környezet határvonalát. A valóság kevert illúziója így előre meghatározott keretek között, valós eszközök felhasználásával G multimédiás tartalmak áttetsző fóliaként árnyalják az általunk tapasztalt valóság elemeit, így megteremtve a kiterjesztett valóság illúzióját. A valóság és a digitális tartalom képes a néző előtt szinte teljesen egybeolvadni. A digitális vászon azonban csak az illúzió szimulációs felületeivel képes kölcsönös visszajelzésre, a valós környezet érzékelése tehát nem szűnik meg.

MVC (Model-View-Controllers)
Szoftverfejlesztési módszer, amely az (Adat-)Modell-nézet-vezérlő programtervezési minta épül.

MySQL / MariaDB
Nyílt forráskódú relációs adatbázis-kezelő rendszer.

Node.js
JavaScript alapú ingyenes szerver- és, futtatókörnyezet.

Nodemon
Node.js alkalmazások automatikus (változtatás alapú) újraindításához használt segédalkalmazás.

Npx
Az npx egy CLI-eszköz, amely az npm csomagok futtatását támogatja.

Npm (Node Package Manager)
A Yarn-hoz hasonló tulajdonságokkal rendelkezik.

ORM (Object Relational Mapping)
Objektum-relációs leképezés.

Pixel
Az adott kép vagy a monitoron látható kép legkisebb megkülönböztethető alkotóeleme.

Prettier
VsCode Extension, a Visual Studio Code beépülő kódformázója.

React.js
JavaScript könyvtár felhasználói felületek létrehozásához.

(3D) Render
A 3D vektorgrafikus modelltér objektumainak raszteres képen való megjelenítéséhez szükséges képszámítási folyamat. A renderelés az az eljárás, mely során a grafikát a képernyőre rajzolja egy rendszer. (A képszámítást végző rendszerben általában két chip található. A központi feldolgozóegység (CPU) mely az alkalmazáslogikai folyamtok futtatásáért felel, továbbá a grafikus feldolgozóegység (GPU), mely kifejezetten a grafikus folyamatokat számításait gyorsítja. Általánosságban a CPU-n lévő alkalmazáslogika építi fel a renderelési utasítások listáját, majd ezeket a továbbítja a GPU felé. A CPU-kat úgy tervezik, hogy hatékonyan futtassanak soros utasításokat, ami inkább az alkalmazáskódokra jellemző, míg a GPU-k úgy vannak tervezve, hogy hatékonyan futtassanak párhuzamos utasításokat. Utóbbi jellemzően rendereléskor érvényesül. Míg a CPU is alkalmas a rajzolás folyamatát ellátni, ez általában lassabb és időigényesebb, így nem érvényesíthető valósidejű 3D alkalmazói környezetben. Real-time környezetben a GPU-t használják, mely kifejezetten ezt a célt szolgálja.)

RESTful API (Representational State Transfer)
Olyan programozási interfész, amely a webes alkalmazások közötti kommunikációra szolgál.

Percepció
Pszichológiai folyamat, mely az érzékelésből és a hozzá tartozó megismerési és gondolkodási (vagy kognitív) folyamatból áll.
PMA (phpMyAdmin)
MySQL adatbázis menedzselésére alkalmazható, nyílt forráskódú PHP alapú adminisztrációs felület.

Plasztikus film

Az 1950-es évek magyar sztereoszkóp filmes eljárása, melyet Bodrossy Félix operatőr talált fel. 1952. július 25-én mutatták be az első plasztikus filmet. A Bajcsy Zsilinszky úti Toldiban nagy érdeklődés kísérte Bodrossy Félix és Győrffy József Állatkerti séta című művének vetítését. A rendező és alkotótársának első 3D-s rövidfilmjét további nagy népszerűségnek örvendő vetítések is követték. A fekete fehér Artistavizsga, a Sztereó híradó is folyamatosan a moziműsoron szerepelt. A plasztikusfilmes alkotások olyan filmtörténeti és filmtechnikai emlékeket őriznek, melyek egyedülállóak a magyar mozgókép-gyártás történetében.

Plugin

Kiegészítő, beépülő modulok, melyet egy szoftverhez lehet hozzáadni.

Poligon

A térbeli tárgyakat felszínhálójuk sokszög modelljeivel reprezentáljuk. A 3D felszínhálót síkidomokból, jellemzően háromszögekből, vagy négyszögből építjük fel.

Promise API

A meghatározhatatlan időt igénylő, de véges feladatokat, folyamatokat fázisonként (Pending, Fulfilled / Resolved, Rejected) kezelő osztály API-ja.

Rasztergrafika

A rasztergrafika a képek digitális megjelenítésének egyik módja, amelyben a képet különböző méretű és színű képpontokból, vagy pixelekből álló rácsra, vagy raszterre bontják. A rasztergrafikák többnyire fotók és digitális képek esetében használhatók, és eltérnek a vektorgrafikáktól, amelyek matematikai modellek segítségével írják le a képeket. A rasztergrafikák fájlmérete nagyobb, mint a vektorgrafikáké, és a nagyításkor a kép minősége romlik, mivel a pixelek növekedése miatt a kép elveszíti élességét.

Render engine

A render engine-ek számítógépes hardverekből állnak, melyek a 3D grafikus műveletek képszámításait végzik.

Renderelés

Egy geometriai modellekből álló jelenet képszámítási folyamata. A képalkotás során az objektumokra ható fények, felület-, anyag- és árnyékmintázata jelenik meg vizuálisan értelmezhető (szekvenciális fázisok) rajzolatként. Egy kétdimenziós kép elkészítéséhez a térbeli jelenetet három dimenzióból két dimenzióba kell projektálni. A renderelés tehát a jelenet fényképezésének felel meg. A kivetítés (vagyis a néző) pozícióját a virtuális kamerák segítségével a digitális térben bárhol elhelyezhetjük. A folyamat azon részét, amikor a pixelekhez rendeljük az adott színezetet raszterezésnek nevezzük, míg a képkészítés teljes folyamatát renderelésnek. A grafikus motorok képszámítás során elemzik és véglegesítik az adott tér textúráját, fényelését és minden olyan tulajdonságát, amelyek a 3D objektumok és képek megjelenését befolyásolják.

SASS (Syntactically Awesome Style Sheets)
A CSS kiterjesztése. Olyan CSS előfeldolgozó-, ill. fordító-nyelv, amely változók, mixinek, függvények használatát biztosítja egy CSS-sel megegyező szintaxis-környezetben.

SBS (Side by side) technológia
Két egész képet tartalmazó sztereoszkóp technikával készült kép.

Shading nyelv
A shading nyelv olyan programozási nyelv, amely lehetővé teszi a 3D grafikai elemek megjelenítéséhez használt árnyékoló algoritmusok megírását. Ezek az algoritmusok határozzák meg, hogyan jelenik meg egy adott modell a képernyőn, milyen színekkel, árnyalatokkal és árnyékokkal. A WebGL és a WebGPU is tartalmaznak saját shading nyelveket. A WebGL a GLSL (OpenGL Shading Language) nyelvet használja, míg a WebGPU a WGSL (WebGPU Shading Language) nyelvet használja. Ezek a nyelvek lehetővé teszik a fejlesztők számára, hogy pontosan meghatározzák, hogyan jelenjenek meg a 3D objektumok a webes alkalmazásukban.

SVG (Scalable Vector Graphics)
Egy XML alapú leíró nyelv, kétdimenziós, statikus és mozgó vektorgrafikák meghatározására.

Szignál transzdukció

Az érzékelés biológiai alapfolyamata, mely során a külvilág fizikai jeleit érzékszervek receptorai ingerület, vagyis neurális impulzusok fogmájában továbbítják a központi idegrendszer felé.

Sztereoszkópia
A sztereoszkópia az a jelenség, mely során a kétdimenziós képeket egyedileg pozícionálva állítják elő úgy, hogy figyelembe veszik a két szem nyújtotta lehetőséget. A nézőszemüveges technológia a térhatás érzetét kelti a viselő számára. A kétlencsés sztereókamerákkal történő rögzítés mellett ma már széles körű 3D-s technikai (utómunka) módszerek is rendelkezésre állnak a térbeli vízió érzetének megteremtésében.

Three.js, babylon.js
Ingyenes, nyílt forráskódú, JavaScript nyelven írt 3D könyvtárak.

Texel
A textúra legkisebb alkotóeleme.

Textúra
A texelekből felépülő bitkép, mely a poligonokra feszítve alkot rajzolatot. Az anyagok egyik legfontosabb tulajdonsága a textúra. Általánosságban elmondható, hogy a textúrázás az anyagtulajdonságok megváltoztatásának módja egy felületen, pontról pontra festve. A kép felvihető egy felületre úgy, hogy a kép úgy néz ki, mintha festve lenne rá. A 3D textúrák általában tehát képekből állnak, amelyeket a 3D objektumok felületére szerkesztenek.

Transzformáció
Egy 3D alkotás létrehozásakor a tér és elemeinek minden pontja három koordinátával definiálható (x,y,z). A tervezés során komponensekbe szervezett alakzatok transzformációs adatait egységesen is beállíthatjuk. A geometriai transzformáció átméretezéssel, forgatással és pozícionálással paraméterezhető. A transzformáció értékeit a világ-, és az objektum koordináta-rendszer segítségével is leírhatjuk. A transzformáció során az grafikus könnyen megvalósíthatja a 3D objektumok és képek méretezését, forgatását, eltolását és tükrözését a háromdimenziós térben.

TSX

A JSX TypeScript megfelelője.

Typescript
Nyílt forráskódú, objektum-orientált, statikusan típusos, szkript nyelven kialakított fejlesztői keretrendszer

UI (User Interface)
Alkalmazások felhasználói felülete.

Vektorgrafika
A vektorgrafika olyan grafikai formátum, amely a képeket matematikai modellek segítségével írja le, invektorok és egyéb geometriai alakzatok segítségével. A vektorgrafikák kiválóan alkalmazhatók a grafikai modellek átméretezéskor, mivel a matematikai modellek alkalmazásával a kép újra rajzolódik a nagyított méretben. A vektorgrafikák általában kisebb fájlméretűek, mint a raszteres képek, és könnyen módosíthatók, mivel minden egyes elemét külön-külön lehet szerkeszteni. A kép raszteres megjelenítéshez szükséges keretpufferben való tárolás sokkal több memóriát igényel, ezért a vektoros képmegjelenítés lényegesen gyorsabb lehet.

Vertex
A háromdimenziós térben elhelyezkedő poligonok végpontjait jelölő legkisebb sarokpontok.

VR (Virtual Reality)
A ’virtuális valóság’ a 3D-s szimulációs környezet egyik digitális és multimédiás formátuma. A VR olyan elképzelt (vagy éppen valóságos) környezetet replikál, mely minden elemében digitális referenciákkal rendelkezik. A CG VR (Computer Generated Virtual Reality) kifejezés olyan technológiai dimenzióra utal, melyet teljes egészében a számítógép által generált tartalom alkot. Az immerzív VR azt jelenti, hogy a néző számára a környezet (valóság-) érzékelését teljes egészében számítógép által vezérelt digitális dimenzió váltja fel. A fizikai tér (audio)vizuális érzékelése is teljesen megváltozik. A látómező háromszázhatvan fokos kiszélesítésével a passzív néző résztvevővé válik, a digitális adatok érzékelése és értelmezése a multiszenzoros technológia segítségével interaktívan, valós időben történik. A panorámavászon képes az immerzivitás érzését kelteni a nézőben, vagyis azt a perceptuális és pszichológiai érzetet váltja ki, hogy a virtuális világban vagyunk. A háromdimenziós virtuális

vászon újfajta képi eszközt jelent a filmes és a színházi szakembereknek, a 3D-s grafikus tervezőművészeknek.

VR 180
Videóformátum, melyben a néző 180 fokos szögben tekintheti meg a képet, azaz oldalirányban szabadon forgathatja a fejét. A 360 fokos videókkal ellentétben a projekció így a mozgókép egyik felében látható, míg a másik fél láthatatlan marad.

VR 360
Videóformátum, mely a néző számára 360 fokos szögben jeleníti meg a képet. A megtekintés során teljes körben, 360 fokban láhatóvá válik a mozgóképes tartalom.

VSCode
Microsoft Visual Code
nyílt forráskódú kódszerkesztő program. Az alkalmazás online verziója ezen a linken érhető el
https://vscode.dev/

W3C (World Wide Web Consortium)
Webes szabványozásért felelős nemzetközi szervezet

WebAR (Web-based Augmented Reality)
A böngészők grafikai vászontulajdonságával elérhető, platformfüggetlen, HTML5 alapú AR technológia, mely a (mobil)készülék kameráját felhasználva valós idejű 3D tartalom projekciójára képes. A kiterjesztett valóság térbeli vizualizációs eszköze a háromdimenziós tervezés és megjelenítés egyik legújabb iránya. Segítségével már a felvételkészítés folyamata során fóliaszerű rétegezésben illeszthetők a digitális grafikai elemek a valós térre.

WebGL (Web-based Graphics Library)
A webalapú grafikus könyvtár elnevezés egy JavaScript API, mely a böngészőmotor vászontulajdonságán keresztül futtatható 3D-s vizualizációs eszközkészletét és interaktív tulajdonságcsoportját jelenti. A grafikus programkönyvtár a HTML szkript és JavaScript programozási nyelven keresztül böngésző- és platformfüggetlenül biztosítja az interaktív 3D-s grafikai megjelenítést. A WebGL a HTML5 Canvas elemében fut, így a DOM API-n

keresztül számos nyelvbe beágyazható. Az OpenGL webes implementációjának is tekinthető rendszer (OpenGL ES 3.0) 2D és 3D API-val is rendelkezik, képes kihasználni a GLSL-t (az OpenGL shading nyelvet) és engedélyezi a grafika (GPU) hardveres gyorsítását is. Igaz azonban, hogy ez utóbbi az egyszálú feldolgozás koncepciójához alkalmazkodik. A dolgozat munkanaplójában részletezett módon a WebGL keretrendszerek segítségével (babylon.js, tree.js) a meglévő 3D-s jelenetek transzplantálhatók lesznek az immerzív interaktív környezet irányába is. A legkésőbbi WebGL 2.0 szabvány 2017-es bevezetése óta a 3D tervezőművészek számára is megfelelőbb minőségű kompozíció- és objektumtovábbítást tesz lehetővé.

WebGPU
A legmodernebb böngészőalapú HTML5 Canvas elemre épülő 3D-s grafikai renderelési technika, mely a párhuzamosított működésre optimalizált modern GPU számításokat támogatja. A WebGPU a modern számítógépes grafikus API-k, például a Vulkan, a DirectX 12 és a Metal webes architektúrájának verziója, melyek nem kapcsolhatók a WebGL sztenderdhez. A GPU alapú webes 3D-s megjelenítés jelenleg legdinamikusabban fejlődő iránya. Olyan szabványos JavaScript API, mely a WebGL-nél lényegesen hatékonyabban támogatja a grafikus processzorban lévő lehetőségeket. Nagyobb számítási hatásfokkal, jobb 3D-s grafikus képességekkel rendelkezik elődjénél, azonban jelenleg még csak egyetlen böngésző támogatásával rendelkezik.

WebGPU IDL (Web Graphics Processing Unit Interface Definition Language)
Interfészleíró nyelv, amelyet az alkalmazások használnak a WebGPU API-val való kommunikáláshoz.

Webpack
A Webpack.js nyílt forrású csomagkezelő rendszer (más néven modulcsomagoló), amely a függőségeket tartalmazó js állományokat statikus fájlok előállítására való. A Webpack mellett számos build tool elérhető. A Vite, Gulp, Parcel, stb. megfelelő alternatívát jelenthet.

WebVR
Böngészőre épülő VR technológia

WebXR

Az immerzív web legújabb szabványrendszere, mely a VR, AR, MR 3D renderelést támogatja.

XAMPP
Nyílt forráskódú, platformfüggetlen helyi webszerver kialakítására használható program.

XR
Az XR gyűjtőfogalom, mely a virtuális-, a kevert- és a kiterjesztett valóság immerzív fejlesztési technikáit és meghatározásainak fogalomkörét összesíti. A rövidítés első betűje spektrumot jelöl, mely a teljes (fizikai) valóságtól az immerzív képi dimenziókig terjed. A tulajdonságegyüttesek skálájának meghatározásait a kiterjesztett valóság színterei szakaszolják. A meghatározás arra utal, hogy a valóság audiovizuális élménye és a digitális tartalom egy időben, folyamatos interakcióban létezik. Az XR elméleti terminusain túl egy létező, új technológiai módszertan is.

Yarn, npm
CLI eszközök (yarn, npm), vagy programsoros felület, amely a különféle csomagok és függőségeik telepítését biztosítja

Z-buffer
Mélységpuffer eljárás, mely során az egymást takaró, azonos térbeli mélységi pozícióval rendelkező poligonok közül a látható elem (legkisebb automatikus z-index kiosztással) kiválasztása kerül.

13. Bibliográfia

Anagnost, Andrew: Maya documentation.
Autodesk Inc. 2019. (https://knowledge.autodesk.com/support/maya/getting-started/caas/simplecontent/content/maya-documentation.html) Utolsó letöltés 2019.04.20.

Angel, Edward - Shreiner, Dave: *Interactive Computer Graphics: A Top-Down Approach with Webgl.* Cambridge, Pearson Kiadó, 2014.

Armour, Theo - Cabello, Ricardo - Masson, Paul: three.js. *Github Inc.*, 2018. (https://threejs.org/docs/index.html#manual/introduction/Creating-a-scene) Utolsó letöltés 2018.04.27.

Axel Rauschmayer: Exploring ECMAScript 6: Upgrade to the Next Version of JavaScript, O'Reilly *Media*, 2015. (https://exploringjs.com/es6/) Utolsó letöltés: 2019.02.14.

Ács Tibor Adrián: Háromdimenziós üzenetek a múltból. *Múlt-kor*, 2009. tél (https://mult-kor.hu/20090217_haromdimenzios_uzenetek_a_multbol?print=1) Utolsó letöltés: 2018.04.08.

Barson, Michael: André De Toth. i*n: Encyclopædia Britannica*, 2013. (https://www.britannica.com/biography/Andre-De-Toth) Utolsó letöltés: 2022.01.14.

Barson, Michael: George Pal. i*n: Encyclopaedia Britannica*, 2013. (https://www.britannica.com/biography/George-Pal) Utolsó letöltés: 2022.01.14.

Basarat Ali Syed: TypeScript Deep Dive. *GitBook*, 2022. tél (https://basarat.gitbook.io/typescript/) Utolsó letöltés: 2022.01.14.

Bártfai Andrea: Az interaktív film. *Médiakutató,* 2011. tél (https://mediakutato.hu/cikk/2011_04_tel/05_interaktiv_film) Utolsó letöltés 2023.04.07.

Bouaziz Sofien - Valentin Julien: Introducing TensorFlow Graphics: Computer Graphics Meets Deep Learning. TensorFlow Blog, 2019. nyár

(https://blog.tensorflow.org/2019/05/introducing-tensorflow-graphics_9.html) Utolsó letötlés: 2023.04.08.

Bodó Barna: Mi a holográfia? *Megyei Tükör*, 1971. tél (https://adtplus.arcanum.hu/hu/view/HaromszekMegyeiTukor_1971_02/?query=Mi%20a%20hologr%C3%A1fia%3F&pg=75&layout=s) Utolsó letöltés: 2022.01.20.

Bodrossy Félix: A plasztikus film. Budapest, *Művelt Nép Könyvkiadó,* 1953.

Bodrossy Félix: Volt egyszer egy plasztikus film… in: *Az Élet és Tudomány Kalendáriuma* 1981. /1 270-272.

Bódy Gábor: *Filmiskola*. Budapest, Palatinus-könyvek, 1998.

Bódy Gábor: *Végtelen kép. Bódy Gábor írásai.* Budapest, Pesti Szalon Könyvkiadó, 1996.

Brandon Jones, Colin MacKenzie: glMatrix API (https://glmatrix.net/). Utolsó letöltés: 2021.12.06.

Brandon Jones, Donovan Hutchence, Jaume Sánchez, Takahiro Aoyagi: WebGL and WebGPU Updates. *Khronos*, 2022. tél (https://www.khronos.org/developers/linkto/webgl-webgpu-updates-january-2022) Utolsó letöltés: 2022.03.14.

Brandon Jones, François Beaufort: Your first WebGPU app. *Google Developers*, 2023. nyár (https://codelabs.developers.google.com/your-first-webgpu-app#0) Utolsó letöltés: 2023.08.30.

Bynens, Mathias: WebGPU — All of the cores, none of the canvas. *Surma.dev,* 2023. tavasz (https://surma.dev/things/webgpu/index.html) Utolsó letöltés: 2023.04.05.

Cantor, Diego: *WebGL Beginner's Guide*. Birmingham, Packt Publishing, 2012.

Catuhe, David - Rousset, David - Vandenberghe, Sebastien: babylon.JS. *Github Inc.* 2018. (https://doc.babylonjs.com/) Utolsó letöltés 2018.04.27.

Csaplár Vilmos: Bódy Gábor titkos története. *NOL*, 2005. tél (http://nol.hu/archivum/archiv-385435-198090) Utolsó letöltés: 2019.06.20.
Csákvári Géza: Üzenetek a jövőnek, 1973-ból, 3D-ben.
Nol, 2017. nyár (http://nol.hu/kultura/20120725-uzenetek_a_jovonek-1320811) Utolsó letöltés: 2022.01.05.

Cservény József: A televízió jövője. *Korunk*, 1957. tavasz (https://adtplus.arcanum.hu/hu/view/Korunk_1957/?query=P.V.%20Smakov%20professzor&pg=1151&layout=s) Utolsó letöltés: 2022.01.20.

Crawford, C. HTML5 Canvas: The Basics. *IBM*, 2023. (https://www.ibm.com/developerworks/library/wa-html5-canvas-basics/) Utolsó letöltés: 2023.02.03.

Darizsh, Dreakhshani: *Maya 3D modellezés és animáció.* Budapest, Perfact kiadó,2009.

Damján Szabolcs: Manipulating video in a browser. *Medium*, 2022. tél (https://medium.com/docler-engineering/manipulating-video-in-a-browser-5b37f8149d9b) Utolsó letöltés 2023.02.10.

Damján Szabolcs: Real-time WebGL video manipulation. *Medium*, 2022. tél (https://medium.com/docler-engineering/webgl-video-manipulation-8d0892b565b6) Utolsó letöltés 2023.02.10.

Dinur, Eran: *The Filmmaker's Guide to Visual Effects: The Art and Techniques of VFX for Directors, Producers, Editors and Cinematographers.* Oxford, Focal Press, 2015.

Dirksen, Jos: *Learning Three.js: The JavaScript 3D Library for WebGL.* Birmingham, Packt Publishing, 2015.

Dorin, Alan: Moving Pictures. Monash.edu, 2009. ősz (https://users.monash.edu/~cema/courses/FIT3084/lectureNotes/lect23.html) Utolsó letöltés: 2023.08.30.

Dragon Zoltán: A film a digitalizáció korában – bevezető. *Apertúra*, 2011. tavasz (https://www.apertura.hu/2011/tavasz/dragon-film-digitalizacio-koraban/) Utolsó letöltés: 2022.06.03.

Dragon Zoltán: A szoftver és a film: a film helye a digitális kultúrában. *Apertúra*, 2009. tél (http://uj.apertura.hu/2009/tel/dragon/) Utolsó letöltés: 2022.06.03.

Dragon Zoltán: Újmozi, avagy adatbázis az egész világ. Válasz Sághy Miklós A film jövője: adatbázis és/vagy (interaktív) narratíva? című reflexiójára. *Apertúra*, 2006. tél (http://uj.apertura.hu/2011/nyar/dragon-ujmozi-avagy-adatbazis-az-egesz-vilag/) Utolsó letöltés: 2019.05.10.

Dzmitry Malyshau: A Taste of WebGPU in Firefox. *Mozilla.org*, 2020. tavasz (https://hacks.mozilla.org/2020/04/experimental-webgpu-in-firefox/) Utolsó letöltés: 2022.01.06.

Dzmitry Malyshau: Point of WebGPU on native. *Github*, 2020. nyár (https://kvark.github.io/web/gpu/native/2020/05/03/point-of-webgpu-native) Utolsó letöltés: 2022.01.10.

Eiler Emil - Karácsonyi Géza: *Sztereo térfényképezés – térlátás.* Budapest, Műszaki Könyvkiadó, 1959.

Eck, David J.: Introduction to Computer Graphics. *Hws.edu.* 2021, nyár (https://math.hws.edu/graphicsbook/index.html) Utolsó letöltés: 2023.04.05.

Dr. Fekete Róbert Tamás - Dr. Tamás Péter - Dr. Antal Ákos - Décsei-Paróczi Annamária: 3D megjelenítési technikák. *BME-MOGI*, 2014. (http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_3d_megjelenitesi_technikak/ch06s03.html) Utolsó letöltés: 2023.11.10.

François Beaufort & Corentin Wallez: Access modern GPU features with WebGPU. *Web.dev*, 2023. (https://web.dev/gpu/) Utolsó letöltés: 2023.03.07.

Fredrik, Näslund - Munch, Torben: Viz Artist User's Guide. *Vizrt Inc.*, 2017. (https://documentation.vizrt.com/viz-artist-guide/3.9.1/Viz_Artist.html) Utolsó letöltés 2018.04.27.

Fredrik, Näslund - Munch, Torben: Viz Mosart User Guide. *Vizrt Inc.*, 2018. (https://documentation.vizrt.com/viz-engine-guide/3.9.1/Viz_Engine.html) Utolsó letöltés 2018.04.27.

Fredrik, Näslund - Munch, Torben: Viz Pilot User Guide. *Vizrt Inc.*, 2018. (https://documentation.vizrt.com/viz-pilotguide/8.2/Viz_Pilot_User_Guide.html) Utolsó letöltés 2018.04.27.

Freeman, A.: AngularJS Directives Fundamentals. *Pluralsight*, 2021. tél (https://app.pluralsight.com/library/courses/angularjs-directives-fundamentals/table-of-contents) Utolsó letöltés: 2021.04.10.

Füzi Izabella: A képernyő és a filmvászon hibridizációja: a videóchat és a videószelfi játékfilmes idézése (Remélem, legközelebb sikerül meghalnod, FOMO – Megosztod és uralkodsz, Szép csendben). *Aperatúra*, 2021. nyár (https://www.apertura.hu/2021/nyar/fuzi-a-kepernyo-es-a-filmvaszon-hibridizacioja-a-videochat-es-a-videoszelfi-jatekfilmes-idezese/) Utolsó letöltés 2021.10.21.

Füzi Izabella: Néző és közönség − Mozitörténeti vázlat. *Apertúra,* 2018. tavasz (http://uj.apertura.hu/2018/tavasz/fuzi-nezo-es-kozonseg-mozitorteneti-vazlat/) Utolsó letöltés: 2019.05.10.

Gaál Gábor: A holográfia: dokumentum és fikció. *Korunk*, 1977. 8.szám. (https://adtplus.arcanum.hu/hu/view/Korunk_1977/?query=dokumentum%20%C3%A9s%20fikci%C3%B3&pg=733&layout=s) Utolsó letöltés: 2022.01.15.

Galvan, Alain: Raw WebGPU. *Github*, 2023. tél, (https://alain.xyz/blog/raw-webgpu) Utolsó letöltés: 2023.01.02.

Gantner Ilona: A múlt, a jelen és a jövő filmje Moszkvában. *Népszava*, 1979. nyár (https://adtplus.arcanum.hu/hu/view/Nepszava_1979_08/?query=%3A%20a%20m%C3%BAlt%2C%20a%20jelen%20%C3%A9s%20a%20j%C3%B6v%C5%91%20filmje%20moszkv%C3%A1ban&pg=191&layout=s) Utolsó letöltés: 2022.01.12.

Gelencsér Gábor: Forgatott könyvek. Adaptációk az 1945 utáni magyar filmben. *Apertúra*, 2006. tél. (http://www.apertura.hu/2006/tel/gelencser/index03.htm) Utolsó letöltés: 2022.01.10.

Gerencsér Péter: A net art az net art az net art. A médiumspecifikusság és a posztmédia dichotómiája az internetes művészetben. *Aperatúra*, 2016. nyár (https://www.apertura.hu/2019/tel/simons_az-iphone-es-a-youtube-kozott-a-filmek-mozgasban/) Utolsó letöltés 2022.05.10.

Gerencsér Péter: Bevezetés a web 2.0 definícióiba és ideológiáiba. *Aperatúra*, 2019. tél (https://www.apertura.hu/2019/tel/gerencser-bevezetes-a-web-2-0-definicioiba-es-ideologiaiba/) Utolsó letöltés 2023.01.06.

Gorácz Anikó: 3D revolúció. A 3D múltja, jelene és jövője. Térélmény. *Filmvilág*, 2009 ősz (https://www.filmvilag.hu/xista_frame.php?cikk_id=9878) Utolsó letöltés: 2024.08.23.

Greguss Ferenc: Térhatású tv - hologram a képernyőn közvetítés lézersugárral. *Magyarország*, 1969. nyár (https://adtplus.arcanum.hu/hu/view/MagyarorszagUj_1969_2/?query=hologram%20a%20k%C3%A9perny%C5%91n%20k%C3%B6zvet%C3%ADt%C3%A9s%20l%C3%A9zersug%C3%A1rral&pg=438&layout=s) Utolsó letöltés: 2022.01.20.

Greguss Pál: Holográfia - az információk rögzítésének új útja. *Haditechnika - Haditechnikai szemle* 2., 1968, 4. szám. (https://adtplus.arcanum.hu/hu/view/Haditechnika_1968/?query=Greguss%20p%C3%A1l&pg=139&layout=s) Utolsó letöltés: 2022.01.23.

Greguss Pál: A térbeli fényképezés: A holográfia. *Népszabadság*, 1968. nyár (https://adtplus.arcanum.hu/hu/view/Nepszabadsag_1968_06/?query=A%20t%C3%A9rbeli%20f%C3%A9nyk%C3%A9pez%C3%A9s&pg=202&layout=s) Utolsó letöltés: 2022.01.23.

Gullen, Ashley: A brief history of graphics on the web and WebGPU. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/webgl-webgpu-construct-1519) Utolsó letöltés: 2022-07-03.

Gullen, Ashley: From WebGL to WebGPU in construct. *Construct*, 2020. tavasz (https://www.construct.net/en/blogs/ashleys-blog-2/webgl-webgpu-construct-1519) Utolsó letöltés: 2022-07-03.

Gyímesi Ferenc: Látványholográfia, holografikus méréstechnika és digitális holográfia. *Magyar Tudomány*, 2005 / 12. (Https://adtplus.arcanum.hu/hu/view/akademiaiertesito_matud_2005/?query=holografikus%20film&pg=1562&layout=s) Utolsó letöltés: 2022.12.05.

Hartai László: A zék, az alfák és a filmoktatás. *Apertúra*, 2018. tél (https://www.apertura.hu/2018/tel/hartai-a-zek-az-alfak-es-a-filmoktatas/) Utolsó letöltés: 2019.05.10.

Henke Gustavo: Concurrently - Run commands concurrently. *GitHub*, 2023. tavasz (https://github.com/open-cli-tools/concurrently) Utolsó letöltés: 2023.04.07.

Herb A. Lightman: Behind the Scenes on Westworld: AC Talks to Writer-Director Michael Crichton. *American Cinematographer*, 2020. tavasz (https://theasc.com/articles/behind-the-scenes-on-westworld) Utolsó letöltés: 2023.08.28.

Horváth Annamária: A jövő televíziója. *Népszava*, 1979. tavasz (https://adtplus.arcanum.hu/hu/view/Nepszava_1979_03/?query=A%20j%C3%B6v%C5%91%20telev%C3%ADzi%C3%B3ja&pg=255&layout=s) Utolsó letöltés: 2022.01.15.

Jánossy Mihály: A holográfia története. *Akadémiai Értesítő - Magyar Tudomány*, 1972. 7-8.szám. (https://adtplus.arcanum.hu/hu/view/AkademiaiErtesito_MATUD_1972/?query=hologr%C3%A1fia%20t%C3%B6rt%C3%A9nete%20&pg=546&layout=s) Utolsó letöltés: 2022.01.15.

Jackson, Simon: *Unity 3D UI Essentials.* Birmingham, Packt Publishing, 2015.

Jan Simons: Az iPhone és a YouTube között: a filmek mozgásban. *Aperatúra*, 2019. tél (https://www.apertura.hu/2019/tel/simons_az-iphone-es-a-youtube-kozott-a-filmek-mozgasban/) Utolsó letöltés 2021.10.18.

Jason, Torchinsky: Computer animation was a drive down a highway in 1961. Here's a look. *The Autopian*, 2022. tél (https://www.theautopian.com/the-first-realistic-computer-animation-was-a-drive-down-a-highway-in-1961/) Utolsó letöltés: 2023.08.24.

Johns, P.: WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. *Addison-Wesley Professional*, 2014. (http://uniguld.dk/wp-content/guld/DTU/webgrafik/0321902920_WebGL.pdf) Utolsó letöltés: 2022.10.21.

Jones, Brandon: Efficiently rendering glTF models - A WebGPU Case Study. *Github*, 2023. (https://toji.github.io/webgpu-gltf-case-study/) Utolsó letöltés: 2023.03.21.

Judson, Rosebush: A chronology of animation history. A history of computer animation. Chron4.doc, 1989. (https://studylib.net/doc/8170363/chapter-4--a-history-of-computer-animation---a) Utolsó letöltés: 2023.08.18.

Juhász Imre: *OpenGL*. mobiDIÁK Könyvtár. Debrecen, 2003. (http://193.6.8.43/segedlet/dokumentumok/OpenGL/OpenGL.php) Utolsó letöltés: 2023.03.02.

Juho Vepsäläinen: SurviveJS – Webpack 5. *SurviveJS*, 2023. (https://survivejs.com/webpack/) Utolsó letöltés: 2023.04.03.

Julien, Moreau-Mathis: *Babylon.js Essentials.* Packt Publishing, 2016, Birmingham.

Kelemen Zsolt: Arcade Fire 2.0 és az adatbázis-logika.
Apertúra, 2011. tavasz (https://www.apertura.hu/2011/tavasz/kelemen-zsolt-arcade-fire-adatbazis-logika/) Utolsó letöltés: 2019.05.10.

Kenyeres Ágnes: Bodrossy Félix. *Magyar életrajzi lexikon,* 1994. (https://www.arcanum.hu/hu/online-kiadvanyok/Lexikonok-magyar-eletrajzi-lexikon-7428D/b-74700/bodrossy-felix-74C8C/) Utolsó letöltés: 2022.01.14.

Kerlow, V. Isaac: *The Art of 3D. Computer Animation an Effects*. Hoboken, John Wiley&Sons, Inc., 2004. p16.

Kiricsi Zoltán: A robot, aki utált mosogatni. Mikrobi 1. *Comment:com,* 2006. tél (https://comment.blog.hu/2006/12/14/mikrobi_1) Utolsó letöltés: 2018.05.26.

Kispéter Miklós: *A győzelmes film - Film, tudomány, művészet.* Budapest, Királyi Magyar Egyetemi Nyomda, 1938.

Kis Sándor: Bevezető a Magyar Televízió épületében tartott sajtóbemutatóhoz. *Filmrestor*, 2009. tél (http://filmrestor.uw.hu/lap-3D.html) Utolsó letöltés: 2018.05.26.

Kis Sándor: Különleges sajtóvetítés. *MTV Intra*, 2009. tél (http://intra.mtv.hu/filmrestor/lap3D-bodrossy.asp) Utolsó letöltés: 2018.05.25.

Kiss Miklós: „A legapróbb részletekre kiterjedő víziók” Rendezőportrék: Bódy Gábor. *Filmtett*, 2002. ősz (
https://www.filmtett.ro/cikk/1731/rendezoportrek-body-gabor) Utolsó letöltés: 2023.01.20.

Kitching, Alan: The Antics computer animation system. Atlas, 1975. (https://www.chilton-computing.org.uk/acl/applications/graphics/p003.htm) Utolsó letöltés: 2023.08.28.

Kopülov Tacskov: *Televízió és holográfia.* Műszaki Könyvkiadó, Budapest, 1978.

Kurutz Márton: Mozilexikon - Budapest V., Bajcsy-Zsilinszky út 36-38. City filmszínház, Szittya-filmszínház, Úttörő mozi, Toldi plasztikus filmszínház, Toldi mozi. *Hangosfilm*, 2011. (http://www.hangosfilm.hu/mozilexikon/budapest-v-city-szittya-uttoro-toldi) Utolsó letöltés: 2022.01.14.

László Ibolya: Magyar világszabadalom - Új lehetőség a színes rajzfilmgyártásban. *Tolna Megyei Népújság,* 1972. tél. (https://library.hungaricana.hu/hu/view/TolnaMegyeiNepujsag_1972_02/?query=SZO%3D(h%C3%A1rf%C3%A1s)&pg=125&layout=s) Utolsó letöltés: 2022.01.14.

Lev Manovich: Az adatbázis, mint szimbolikus forma. *Aperatúra*, 2009. ősz(https://www.apertura.hu/2009/osz/manovich/) Utolsó letöltés 2022.05.10.

Lev Manovich: Mi a film?. *Aperatúra*, 2009. ősz(https://www.apertura.hu/2009/osz/manovich-3/) Utolsó letöltés 2022.05.10.

Mardan, A.: Node.js 16: The Complete Guide. *Udemy*, 2022. (https://www.udemy.com/course/nodejs-the-complete-guide/) Utolsó letöltés: 2023.11.01.

Marrs, Tom: JSON at Work - Practical Data Integration for the Web. Kalifornia, O'Reilly Media, 2017.

Molnár Aurél: Mi a holográfia? *Tükör*, 1972. 14. szám. (https://adtplus.arcanum.hu/hu/view/Tukor_1972_04-06/?query=%20hologr%C3%A1fia&pg=13&layout=s) Utolsó letöltés: 2022.01.15.

M Tóth Éva, Kiss Melinda: *Animációs mozgóképtörténet I.* Budapest, Typotex Kiadó, 2014.

M Tóth Éva, Kiss Melinda: *Animációs mozgóképtörténet II.* Budapest, Typotex Kiadó, 2014.

Munro, I. (2018). WebGPU - A New Graphics API for the Web. *Dev.to*, 2022. (https://dev.to/azure/webgpu-a-new-graphics-api-for-the-web-25om) Utolsó letöltés: 2022.10.21.

Murray, Scott: *Interactive Data Visualization for the Web, An Introduction to Designing with D3. Kalifornia*, O'Reilly Media, 2017.

Nagy Eszter, Politzer Péter: Variációk a harmadik dimenzióra. *Filmvilág folyóirat,* 1997/01 (http://filmvilag.hu/xereses_frame.php?cikk_id=1379) Utolsó letöltés: 2022.01.14.

Ninomiya, Kai - Wallez, Corentin - Malyshau, Dzmitry: WebGPU Explainer. Draft Community Group Report. *GPU for the Web Community Group,* 2023. tavasz (https://gpuweb.github.io/gpuweb/explainer/). Utolsó letöltés: 2023.04.04.

Okun, Michael - Zwerman, Susan: *The VES Handbook of Visual Effects.* Oxford, Focal Press, 2015.

Ondrejcsik Kálmán: Térhatású sztorik. *A Hét*,1978. 13. szám. (https://adtplus.arcanum.hu/hu/view/AHetMarosvasarhely_1978_01-52/?query=T%C3%A9rhat%C3%A1s%C3%BA%20%20sztorik&pg=141&layout=s) Utolsó letöltés: 2022.01.15.

Orosz Márton: Magyarok a komputerművészet korai történetében. in: Beke László, Orosz Márton, Peternák Miklós: *Magyar művészek és a számítógép*. Budapest, HUNGART, 2016. p16.

Ozogany Ernő: A térhatású kép | 2. *A Hét,* 1978, 9. szám. (https://adtplus.arcanum.hu/hu/view/Ahet_1978_1/?query=t%C3%A9rhat%C3%A1s%C3%BA%20k%C3%A9p%20%7C%202&pg=185&layout=s) Utolsó letöltés: 2022.01.25.

Ozogány Ernő: A térhatású kép | 3. *A Hét,* 1978. 10. szám. (https://adtplus.arcanum.hu/hu/view/Ahet_1978_1/?query=A%20t%C3%A9rhat%C3%A1s%C3%BA%20k%C3%A9p%20%2F%203&pg=209&layout=s) Utolsó letöltés: 2022.01.15.

Parisi, Tony: *WebGL: Up and Running: Building 3D Graphics for the Web.* Kalifornia, O'Reilly Media, 2012.

Parkinson, David: Douglas Trumbull, visual effects visionary behind 2001 and Blade Runner, 1942 to 2022. BFI, 2022. tél (https://www.bfi.org.uk/news/douglas-trumbull-1942-2022) Utolsó letöltés: 2023.08.30.

Patterson, Richard: *Convergence or interaxis.* American Cinematographer, 1983. nyár

Peternák Miklós: *Képháromszög*. Ráció Kiadó, Budapest, 2007. p. 97-101.

Peternák Miklós: *Filmutópia*. A következő száz év. *Filmvilág folyóirat,* 1997/01 (https://filmvilag.hu/xereses_aktcikk_c.php?cikk_id=74) Utolsó letöltés: 2023.06.26.
Peternák Miklós: Szellem az anyagban – az érzéki világ kiterjesztése és a technikai médiumok – 1. rész. *Balkon* 2015/5. 14–22.

Peternák Miklós: Szellem az anyagban – az érzéki világ kiterjesztése és a technikai médiumok – 2. rész. Világkiállítások. Mit jelent kiállítani? *Balkon* 2015/6. 26–30.

Peternák Miklós: Szellem az anyagban – az érzéki világ kiterjesztése és a technikai médiumok – 3. rész. Látni a láthatatlant: látható semmi – láthatatlan valami. *Balkon* 2015/7–8. 26–30.

Peternák Miklós: Szellem az anyagban – az érzéki világ kiterjesztése és a technikai médiumok – 4. rész. Coda: Csontváry, Moholy-Nagy, a technikai médiumok. *Balkon* 2015/9. 20–23.

P. Szabó Dénes: Térhatás most és régen. *Filmvilág*, 2012. nyár (http://filmvilag.blog.hu/2012/07/16/terhatas_most_es_regen) Utolsó letöltés: 2018.05.10.

Sághy Miklós: A film jövője: adatbázis és/vagy (interaktív) narratíva? Válasz Dragon Zoltánnak A film a digitalizáció korában című írásában kifejtett felvetéseire *Apertúra*, 2011. nyár (http://uj.apertura.hu/2011/nyar/saghy-a-film-jovoje-adatbazis-esvagy-interaktiv-narrativa/) Utolsó letöltés: 2022.05.10.

Schedeen, Jesse: The history of 3D movie tech. *Ign*, 2010. (http://www.ign.com/articles/2010/04/23/the-history-of-3d-movie-tech) Utolsó letöltés: 2017.01.14.

Schreiber András: 3D idehaza - Magyar Dimenzió. Bodrossy Félix háromdimenziós mozgóképei. Balogh Tibor holotévéje. 3D – Made in Hungary. *Filmvilág*, 2006 nyár. (http://www.filmvilag.hu/xista_frame.php?cikk_id=8665) Utolsó letöltés: 2017.12

Seguin Damien: Graphics on the Web and Beyond with WebGPU. Medium, 2020. nyár (https://dmnsgn.medium.com/graphics-on-the-web-and-beyond-with-webgpu-13c4ba049039) Utolsó letöltés: 2023.04.02.
Simpson, K.: Understanding React: A Guide for Frontend Developers. *Envato Tuts+*, 2021. (https://webdesign.tutsplus.com/series/understanding-react--cms-1519) Utolsó letöltés: 2023.02.12.

Tanács Attila: *Three.js jegyzet Számítógépes grafika gyakorlathoz.* Szegedi Tudományegyetem, 2022. (http://www.inf.u-szeged.hu/~tanacs/threejs/index.html) Utolsó letöltés: 2022.07.09.

Tari Balázs: Programozás és algoritmizálás JavaScript nyelven. *Educatio Társadalmi Szolgáltató*, 2019 nyár (http://www.inf.u-szeged.hu/~tarib/adatlap.html#tema). Utolsó letöltés: 2023.01.05.

Teszler Tamás: Mozipest. Toldi Mozi. Kísérleti láncolat. *Filmvilág*, 2007/11. (http://www.filmvilag.hu/xista_frame.php?cikk_id=9169) Utolsó letöltés: 2022.01.14.

Tjurin, Sz. Oszipov: Plasztikus Filmszínház született a Szovjetunióban. in: *Élet és tudomány folyóirat*, 1947. január / II. évf. 2. p. 61.

Tóth Edit: Hitchcock Szédülése és Kepes György fényművészete a háború utáni nagyvárosban 1. rész. *Balkon*, 2017. ősz

(http://epa.oszk.hu/03000/03057/00099/pdf/EPA03057_balkon-2017-10_021-025.pdf) Utolsó letöltés: 2023.08.26.

Vajda Péter: A háromdimenziós portré. *Népszabadság*, 1971. 269. szám. https://adtplus.arcanum.hu/hu/view/Nepszabadsag_1971_11/?query=A%20h%C3%A1romdimenzi%C3%B3s%20portr%C3%A9&pg=151&layout=s) Utolsó letöltés: 2022.01.15.

Vajdovich Györgyi: A magyar film az 1950-es években. *Filmhu*, 2014. nyár. (http://archiv.magyar.film.hu/filmtortenet//korszakelemzesek/a-magyar-film-az-1950-es-evekben-filmtortenet-korszakelemzes.html)
Utolsó letöltés: 2022.01.17.

Valuska László: Bodrossy és a 3D - A polárszűrő, az Agy és 2 füstölgő szem. *Filmhu,* 2006. tavasz (http://magyar.film.hu/filmhu/magazin/bodrossy-es-a-3d-beszamolo-szakma) Utolsó letöltés: 2017.12.26.

Varga Erzsébet: A művészet és a tudományos-műszaki forradalom. *A Hét,* 1977. 18. szám (https://adtplus.arcanum.hu/hu/view/Ahet_1977_1/?query=%20%C3%A9s%20a%20tudom%C3%A1nyos-m%C5%B1szaki%20forradalom&pg=422&layout=s) Utolsó letöltés: 2022.01.15.

Varga Zsuzsa: Fényképezés lézersugárral. *Pajtás*, 1975. nyár (https://adtplus.arcanum.hu/hu/view/Pajtas_1975_2/?pg=659&layout=s&query=f%C3%A9nyk%C3%A9pez%C3%A9s) Utolsó letöltés: 2022.01.15.

Varga Zoltán: *A magyar animációs film: intézmény- és formatörténeti közelítések.* Szeged, Pompeji Alapítvány, 2016.

Várhelyi Tamás: A lézersugár fényes jövője. *Népszava*, 1970. tél (https://adtplus.arcanum.hu/hu/view/Nepszava_1970_12/?query=A%20l%C3%A9zersug%C3%A1r%20f%C3%A9nyes%20j%C3%B6v%C5%91je%20&pg=230&layout=s) Utolsó letöltés: 2022.01.15.

Várhelyi Tamás: Dinamikus holográfia. *Népszava*, 1975. tél (https://adtplus.arcanum.hu/hu/view/Nepszava_1975_12/?query=Dinamikus%20hologr%C3%A1fia&pg=205&layout=s) Utolsó letöltés: 2022.01.15.

Wolf, Max - Gregor, Sebastian – VVVV. *Vvvv group, Inc*. 2019. (https://vvvv.org/documentation/getting-started) Utolsó letöltés: 2023.03.01.

Xu, Jack: Practical WebGPU Graphics: Creating Advanced Graphics on the Web Using WebGPU. Google Books, *Apress*, 2022. (https://books.google.hu/books?id=Qwo9zgEACAAJ.) Utolsó letöltés: 2023-04-04.

Zay László: A látható tudás - ismeretterjesztő filmekről. *Magyar Nemzet*, 1979. tavasz (https://adtplus.arcanum.hu/hu/view/MagyarNemzet_1979_03/?query=A%20l%C3%A1that%C3%B3%20tud%C3%A1s%20%20zay&pg=174&layout=s) Utolsó letöltés: 2022.01.15.

Z. Kiss Endre: A mozi-vászon mélységei. *in: Élet és tudomány folyóirat,* 1947. november / II. évf. 26. p. 61.

Technikai dokumentációk

Adobe Inc.: Everything you need to know about physically based rendering. Adobe Systems, 2022. (https://www.adobe.com/products/substance3d/discover/pbr.html) Utolsó letöltés: 2022.07.12.

Axios: Axios documentation. GitHub, 2023. (https://axios-http.com/docs/api_intro) Utolsó letöltés: 2022.07.12.

Babylon.js: Babylon.js documentation. Github, 2023. (https://doc.babylonjs.com/) Utolsó letöltés: 2023.04.01.

Babylon.js: Video As A Texture. Github, 2023. (https://doc.babylonjs.com/features/featuresDeepDive/materials/using/videoTexture) Utolsó letöltés: 2023.04.01.

Bgfx: Cross-platform rendering library. Github, 2023. (https://github.com/bkaradzic/bgfx) Utolsó letöltés: 2023.04.01.

Blender Foundation: Blender documentation – Eevee. Doc.Blender, 2023. (https://docs.blender.org/manual/en/latest/render/eevee/materials/settings.html) Utolsó letöltés: 2023.04.05.

Chromium: CEF – Chromium Embedded Framework. Github, 2023. (https://github.com/chromiumembedded/cef) Utolsó letöltés: 2023.04.05.

Diego Marcos, Don McCurdy, Kevin Ngo: A-Frame: A web framework for building virtual reality experiences. Aframe.io, 2023. (https://aframe.io/) Utolsó letöltés: 2023.04.04.

Google Inc.: Draco 3D data compression. Github, 2023. (https://google.github.io/draco/) Utolsó letöltés: 2023.04.05.

Ecma International: Language Specification. ECMAScript, 2021. (https://www.ecma-international.org/publications/standards/Ecma-262.htm) Utolsó letöltés: 2023.04.03.

Epic Games, Inc: Unreal Engine documentation. UnrealEngine, 2023. (https://docs.unrealengine.com/en-us/) Utolsó letöltés: 2023.04.03.

Facebook Inc.: Bootstrap documentation. GitHub, 2022. (https://getbootstrap.com/docs/5.3/getting-started/introduction/) Utolsó letöltés: 2022.10.10.

Facebook Inc.: Create React App. GitHub, 2022. (https://github.com/facebook/create-react-app) Utolsó letöltés: 2023.04.03.

Facebook Inc.: Create React App. ReactJs, 2022. (https://reactjs.org/docs/create-a-new-react-app.html) Utolsó letöltés: 2023.04.03.

Facebook Inc.: React dokumentáció. ReactJs, 2022. (https://reactjs.org/) Utolsó letöltés: 2023.04.03.

Facebook Inc.: React 360 documentation. GitHub, 2022. tavasz. (https://facebook.github.io/react-360/) Utolsó letöltés: 2022.09.03.

Formium, Inc.: Formik documentation. Formik, 2023. (https://facebook.github.io/react-360/) Utolsó letöltés: 2022.11.24.

Formidable Labs, LLC: Formidable documentation. Github, 2023. (https://github.com/node-formidable/formidable /) Utolsó letöltés: 2022.11.24.

Git Community: Git documentation. GitHub, 2022. tavasz. (https://facebook.github.io/react-360/) Utolsó letöltés: 2022.09.03.

Google LLC: Dawn, a WebGPU implementation. Googlesource, 2020. (https://dawn.googlesource.com/dawn/+/refs/heads/chromium-gpu-experimental/README.md) Utolsó letöltés: 2022.12.10.

Google LLC: TensorFlow. Googlesource, 2023. (https://www.tensorflow.org/) Utolsó letöltés: 2023.04.07.

Google LLC: WebGPU API Specification. Chrome Platform Status. Google, 2022. (https://chromestatus.com/feature/6213121689518080) Utolsó letöltés: 2023.04.04.

JSDoc: JSDoc Official documentation. JsDocApp, 2020. (https://jsdoc.app/) Utolsó letöltés: 2022.10.20.

Khronos Group: GLTF documentation. OpenGL.org, 2023. tavasz. (https://www.khronos.org/gltf/) Utolsó letöltés: 2023.02.10.

Khronos Group: OpenGL API documentation. OpenGL.org, 2023. tavasz. (https://www.opengl.org/wiki/Introduction_to_OpenGL) Utolsó letöltés: 2022.12.03.

Khronos Group: Introduction to OpenGL. OpenGL.org, 2023. tavasz. (https://www.opengl.org/wiki/Introduction_to_OpenGL) Utolsó letöltés: 2022.12.27.

Khronos Group Inc.: Vulkan. Khronos, 2023. (https://www.vulkan.org/) Utolsó letöltés: 2023.03.21.

Khronos Group Inc.: Vulkan Porting Layers. Khronos, 2022. (https://www.vulkan.org/porting) Utolsó letöltés: 2023.03.21.

Microsoft Corporation: TypeScript documentation. Microsoft, 2022. (https://www.typescriptlang.org/docs/) Utolsó letöltés: 2023.03.20.

Microsoft Corporation: JSX. Microsoft 2021. (https://www.typescriptlang.org/docs/handbook/jsx.html) Utolsó letöltés: 2023.03.23.

Microsoft Docs: Model-View-Controller (MVC) overview. Microsoft 2021. nyár (https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0) Utolsó letöltés: 2023.02.10.

Microsoft Learn: Oktatóanyag: Az első WebXR-alkalmazás létrehozása a Babylon.js használatával. Microsoft 2023. tavasz (https://learn.microsoft.com/hu-hu/windows/mixed-reality/develop/javascript/tutorials/babylonjs-webxr-helloworld/introduction-01) Utolsó letöltés: 2023.04.08.

Mozilla Developer Network (MDN): CORS. Mozilla Corporation, 2023. (https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS) Utolsó letöltés: 2023.04.03.

Mozilla Developer Network (MDN): ECMAScript. Mozilla Corporation, 2023. (https://developer.mozilla.org/en-US/docs/Web/JavaScript /Language_Resources) Utolsó letöltés: 2023.04.03.

Mozilla Developer Network (MDN): Mozilla VR: Getting Started with WebVR. Mozilla Corporation, 2022. (https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API/Using_the_WebVR_API) Utolsó letöltés: 2022.10.21.

Mozilla Developer Network (MDN): WebGL API. Mozilla Developer Network (MDN), 2023. (https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API) Utolsó letöltés: 2022.10.21.

Mozilla Developer Network (MDN): WebRTC API. Mozilla Corporation, 2023. (https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API) Utolsó letöltés: 2023.04.07.

Mutler: Mutler documentation. Github, 2023. (https://github.com/expressjs/multer) Utolsó letöltés: 2023.01.21.

Node.js: Node.js v16.10.0 documentation. Node.js, 2022. (https://nodejs.org/docs/latest-v16.x/api/) Utolsó letöltés: 2022.03.27.

Node.js Foundation. GitHub. 2022. (https://github.com/nodejs/node) Utolsó letöltés: 2022.03.14.

OpenCollective: ESLint documentation. Eslint.org, 2023. (https://eslint.org/docs/latest/) Utolsó letöltés: 2023.02.18.

OpenCollective: Sequelize documentation. Sequelize.org, 2023. (https://sequelize.org/docs/v7/) Utolsó letöltés: 2023.03.13.

OpenJS Foundation: ESLint documentation. OpenJs, 2023. (https://eslint.org/docs/user-guide/getting-started). Utolsó letöltés: 2023.04.05.

Oracle Corporation: MySQL documentation. MySQL, 2021. (https://dev.mysql.com/doc/) Utolsó letöltés: 2023.04.03.
ReactJS.org: Create React App. Facebook, 2021. (https://create-react-app.dev/docs/getting-started/) Utolsó letöltés: 2023.02.12.

Oracle Corporation: MySQL Workbench documentation. Oracle, 2023. (https://dev.mysql.com/doc/workbench/en/) Utolsó letöltés: 2023.04.03.

Mutler: Mutler documentation. Github, 2023. (https://github.com/expressjs/multer) Utolsó letöltés: 2023.01.21.

Poimandres: Drei. Github, 2023. (https://github.com/pmndrs/react-xr#readme) Utolsó letöltés 2023.04.07.

Poimandres: React Three Fiber. Github, 2023. (https://docs.pmnd.rs/react-three-fiber/getting-started/introduction), Utolsó letöltés: 2023.04.05.

Poimandres: React-XR. Github, 2023. (https://github.com/pmndrs/react-xr#readme) Utolsó letöltés 2023.04.07.

React-Bootstrap: React-Bootstrap documentation. Github, 2023. (https://github.com/react-bootstrap/react-bootstrap) Utolsó letöltés: 2023.04.05.

Sequelizedocs contributors: Sequelize. Readthedocs, 2022 tavasz (https://sequelize.org/master/) Utolsó letöltés: 2023.04.03.

Sokol: Simple STB-style cross-platform libraries for C and C++. Github, 2023. (https://github.com/floooh/sokol) Utolsó letöltés: 2023.04.01.

StrongLoop - IBM: Express documentation. ExpressJs, 2023. (https://expressjs.com/) Utolsó letöltés: 2023.04.05.

Three.js: Three.js documentation. Github, 2023. (https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene) Utolsó letöltés: 2023.04.01.

Three.js: VideoTexture. Github, 2023. (https://threejs.org/docs/#api/en/textures/VideoTexture) Utolsó letöltés: 2023.04.01.

Unity Technologies: Unity Engine. Unity3D, 2023. (https://docs.unity3d.com/2023.2/Documentation/Manual/webgl.html) Utolsó letöltés: 2023.04.01.

Unreal Engine: Unreal Engine 5 Documentation. UnrealDocs, 2023. (https://docs.unrealengine.com/5.0/en-US/) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: Browser plugin. Vizrt, 2023. (https://documentation.vizrt.com/viz-plugins-user-guide/5.0/Browser.html) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: DataPool. Vizrt, 2023. (https://documentation.vizrt.com/datapool-guide-2.13.pdf) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: Viz Artist. Vizrt, 2023. (https://documentation.vizrt.com/viz-artist-guide-5.0.pdf) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: Artist webinars: Vizrt. 2023. (https://www.vizrt.com/webinars/search?keyword=viz%20artist) Utolsó letöltés: 2023.04.05.

Vizrt Inc.: Viz Engine: Vizrt. 2023. (https://documentation.vizrt.com/viz-engine-guide-5.0.pdf) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: Viz Pilot: Vizrt. 2023. (https://documentation.vizrt.com/viz-pilot-guide-8.9.pdf) Utolsó letöltés: 2023.04.01.

Vizrt Inc.: Viz Pilot: Vizrt. 2023. (https://documentation.vizrt.com/viz-trio-guide-4.1.pdf) Utolsó letöltés: 2023.04.01.

W3C Community Group: GPU for the Web. WebGPU API Specification. W3C, 2021. (https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2023.04.05.

W3C: World Wide Web Consortium. W3.org, 2023. (https://www.w3.org/) Utolsó letöltés: 2023.04.03.

W3C: WebXR Device API. Github, 2022. (https://immersive-web.github.io/webxr/) Utolsó letöltés: 2023.01.20.

WebGPU Working Group: WebGPU Specification. GitHub, 2023.
(https://gpuweb.github.io/gpuweb/) Utolsó letöltés: 2022.10.21.

Webpack Group: Webpack documentation - Getting Started. WebpackJs, 2023.
(https://webpack.js.org/guides/getting-started/) Last accessed: 2023.04.03.

WebRTC: Real-time communication for the web. Google Developers, 2023.
(https://webrtc.org/) Utolsó letöltés 2023.02.10.

Yarn: Yarn Package Manager. Yarn documentation, 2020.
(https://classic.yarnpkg.com/en/docs/) Utolsó letöltés: 2022.11.24.