Szegedi Tudományegyetem
Informatikai Intézet

DIPLOMAMUNKA

Galgóczki Norbert

2024

Szegedi Tudományegyetem

Informatikai Intézet

Kamera szimuláció fizikai jellemzők alapján

Diplomamunka

Készítette:

Galgóczki Norbert

programtervező informatikus MSc szakos hallgató

Témavezető:

Dr. Németh Gábor

egyetemi adjunktus

Szeged

2024

Feladatkiírás

A témát kiíró oktató neve: Németh Gábor

A témát meghirdető tanszék: Képfeldolgozás és Számítógépes Grafika Tanszék

Típus: diplomamunka

Ki jelentkezhet: 1 fő, Programtervező informatikus MSc szakos hallgató

A feladat rövid leírása:

A számítógépes grafikában egy virtuális világba képzelhetjük magunkat, akár animációs filmet is készíthetnénk. Felmerül a kérdés, hogy elő lehet-e állítani olyan látványt, amelyet egy adott kamera fizikai paraméterei (rekesznyílás, mélységélesség, záridő, objektív lencseátmérő, fókusztávolság) határoznak meg.
A jelentkező feladata egy ilyen kamera-modell előállítása változtatható paraméterekkel és a látvány leképezésének bemutatása a számítógépes grafika eszközeivel.

Előismeretek: nem szükségesek

Szakirodalom: angol és magyar nyelvű szakirodalom áll rendelkezésre

Tartalmi összefoglaló

- ***A téma megnevezése:***

  A választott téma, melyet diplomamunkámban kidolgoztam a Kamera szimuláció fizikai jellemzők alapján.

- ***A megadott feladat megfogalmazása:***

  A feladat kiírása alapján egy hétköznapi fényképezőgép használata során készített kép megalkotása során végbemenő folyamatokat kellett szimulálni egy számítógépes programban.

- ***A megoldási mód:***

  A feladat megoldásához a Unity játékmotort használtam a C# programnyelv alkalmazásával. A fényképezőgépek objektívének fizikai jellemzői alapján létrehozott képeket tanulmányoztam a rendelkezésemre álló szakirodalomban. Figyelembe vettem a képalkotása során az objektív különböző beállítási során létrehozott képek mélységélességét.

- ***Alkalmazott eszközök, módszerek:***

  A Unity játékmotor.

- ***Elért eredmények:***

  Az elkészített programban lehetőségek állnak rendelkezésre az objektív tulajdonságainak beállítását követően egy fénykép szimulálására. A programmal készített képek megegyeznek egy fényképezőgép által készített képnek.

- ***Kulcsszavak:***

Unity, képfeldolgozás, kamera, mélységélesség, fókusztávolság, rekesznyílás

Tartalom

Bevezetés

Az életem és a tanulmányim során mindig érdekelt, hogy az informatikai eszközök hogyan kommunikálnak egymással, milyen folyamatok zajlódnak le a digitális eszközökön ahhoz, hogy a valóságnak megfelelően érzékeljék, észleljék esetleg reprodukálni tudják a mindennapi életünk eseményeit. Mindig nagy érdeklődéssel figyeltem a mesterséges intelligencia fejlesztését, melyek segítséget nyújtanak az emberek hétköznapi életében. Napjainkban a mesterséges intelligencia már szinte minden használati eszközünkbe megjelenik. Így van ez a modern videókamerák, fényképezőgépekben is. Mindig szerettem volna megérteni, hogy a fényképezés során milyen folyamartok zajlódnak le a fénykép elkészítése és megjelenítése során. A fényképezésre a szép képek készítésére, mindig csodálattal tekintettem, szinte művészetnek tartottam. A fényképezés történelmében a számítástechnika fejlődésével megindult a digitalizáció, a digitális képalkotás. Napjainkban szinte már csak olyan kamerákat vásárolhatunk, melyek képalkotása digitálisan történik. A fényképezés, illetve a fénykép készítése során lezajlódó fizikai és kémiai folyamatokat mindig érdeklődéssel figyeltem. A fényképezés során nagyon sokat számít, ha ismerjük a kamera működési elveit és beállítási lehetőségeit. Ilyen például a rekeszérték állítása, illetve a fókusztávolság állítása, melyek nagyban befolyásolják az elkészített fényképek minőségét. Művészi szinten egy fényképész a kameráján keresztül az elkészített fotókon meg tudja mutatni a pillanatot, az érzelmet, illetve az élet szépségét, a természet csodáit. A mai felgyorsult világunkban már minden ember elérheti a digitális képalkotás lehetőségét, mert lassan minden ember zsebében ott lapul egy okos eszköz, egy mobiltelefon, egy miniszámítógép, melyekkel képeket, videókat lehet rögzíteni. Korábban ezeken az eszközökön csak rossz minőségű fényképeket, videófelvételeket lehetett elkészíteni, azonban a felgyorsult digitális világunkban, a robbanásszerűen fejlődő mestersége intelligencia által egyre jobb és jobb minőségű képek és videók készíthetők. Napról-napra jelennek meg újítások, új programok, fejlesztések, melyek a digitális életünket folyamatosan kihívások elé állítják. Manapság egy arcfelismerés, egy biometrikus azonosítás megtalálható a legolcsóbb mobileszközön is. A drágább eszközökbe szerelt kamerák és a kép alkotásához használt programok olyan fejlődésen mentek át a XXI. században, hogy manapság egy drága mobiltelefonnal készített kép olyan jó minőségben készül el, mint korábban egy professzionális fényképezőgéppel készített fotó.

Ezért nagyon kíváncsivá tett az Informatikai Intézet kiírásából a Kamera szimuláció feladatkiírás. Szerettem volna mindig megismerni azt, hogy egy számítástechnikai eszköz a digitálisan betáplált instrukciók alapján, hogyan alkotja meg a fényképet, illetve azt, hogy az

így elkészített kép mennyire felel meg a valódi emberi szem által látott világnak. Ezért választottam diplomamunkámnak az Informatikai Intézet kiírásából a Kamera szimulációt.

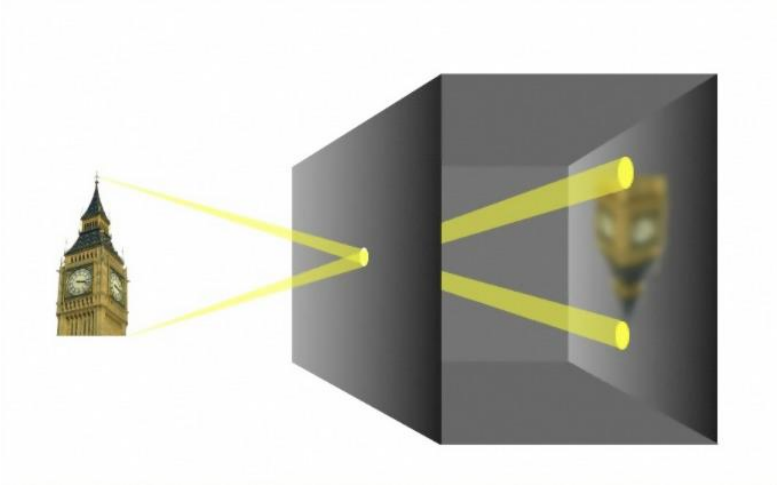
A kiválasztott téma nagyon elgondolkodtatott, többféle megoldás is eszembe jutott, de végül konzulens tanárommal arra a döntésre jutottunk, hogy a képalkotás szimulálásához a Unity keretprogramot fogom használni. A döntésemet az az indok vezérelte, hogy ne kelljen egy programot az alapoktól felépíteni, ezért egy keretrendszert egy játékmotort használtam. Az első gondolatom az volt, hogy ez a játékmotor segíteni fogja a munkámat és könnyebbé teszi a feladat megvalósítását. Azonban a Unity program leírása nem nyújtott teljeskörű támogatást a 3D objektumok tulajdonságainak megfelelő eléréséhez, illetve a kép megjelenítésében, ezért külön algoritmust kellett készítenem.

1 Kamerák

A munkámhoz szükségem volt megismerni a fényképezőgép kamerájának a működésesét. Azért, hogy az így megszerzett ismeret megfelelő alkalmazásával képes legyek a feladatban összeállított elvárásoknak megfelelő fénykép készítésére. A kameráknak rengeteg tulajdonságai vannak, amelyek egymásba fonódnak és kifinomult rendszert alkotnak, ilyenek a lencsék és a szenzorok távolsága, amellyel a fókusztávolság ismerhető meg. A rendszerek finom együttműködésének megértéséhez és szimulálásához az egyik elengedhetetlen ismeret volt a lyukkamerák ismerete. A kamera szó a latin camera obscura szóból ered, amelynek jelentése sötét kamra. Ez egy lencsenélküli optikai eszköz volt, amely a környezet vizuális leképezésére szolgált.

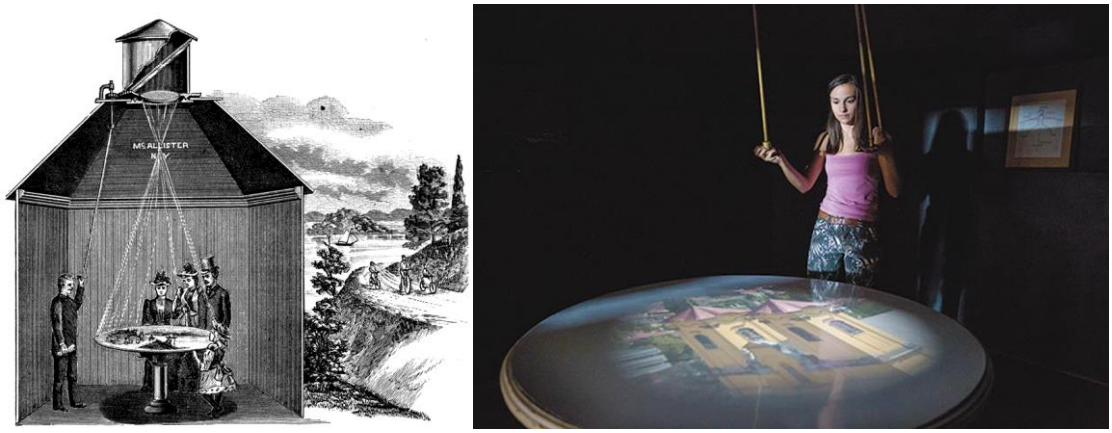
1.1 Lyukkamerák

A képalkotás legkoraibb típusú kamerája a lyukkamera volt. Ennek a kamerának a működési elve az volt, hogy egy teljesen lesötétített szóbában egyetlen fényforráson, vagyis egy lyukon keresztül érkezett a fény a külvilág tárgyairól visszaverődött sugarakból. A lyukon átszűrődő fény a falon fejjel lefelé megjelenő képet eredményezett. A mai fényképezőgépek működései is ennek az eszköznek a működési elvén alapulnak.



1.1.1. ábra Lyukkamera: [1]

A lyukkamerák a kivetített kép eltárolására alkalmatlanok voltak, de az embereket mindig is érdekelte az, hogy hogyan lehetne rögzíteni ezeket a képeket. A camera obscura-t a fényképezés előtti időkben rajzolási segédeszközként használták, a lyukkamera által kivetített kép segítette a rajzolót abban, hogy a képről vázlatot készítsen. Hazánkban is található egy még működő camera obscura az egri Líceumban, mely kamerát Hell Miksa bécsi csillagász tervezte.

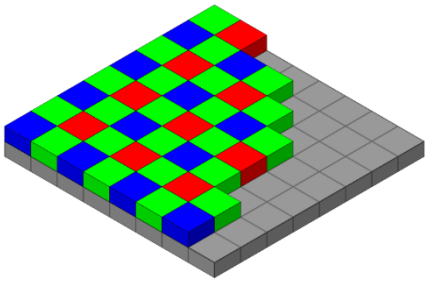


1.1.2. ábra. Egri Líceum lyukkamera.[2]

A Líceum egy kistermében helyezkedik el ez a camera obscura. A terem mennyezetén lévő kerek nyílásban helyezkedik el egy forgatható fémhenger, melyben egy síktükör és egy gyűjtőlencse található. A lencse az összegyűjtött fényt levetíti a terem közepén lévő asztallapra, ahol láthatóvá válik a város látképe.

1.2 Modern kamerák

A modern kamerák a lyukkamerával ellentétben sok fejlődésen mentek keresztül, ilyen például kezdetben a fényérzékeny anyagok használata. Egyes fémek és vegyi anyagok a fény hatására

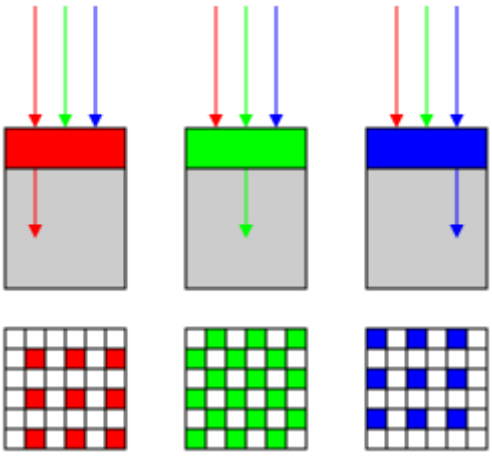


1.2.1. ábra: Bayer szűrő [3]

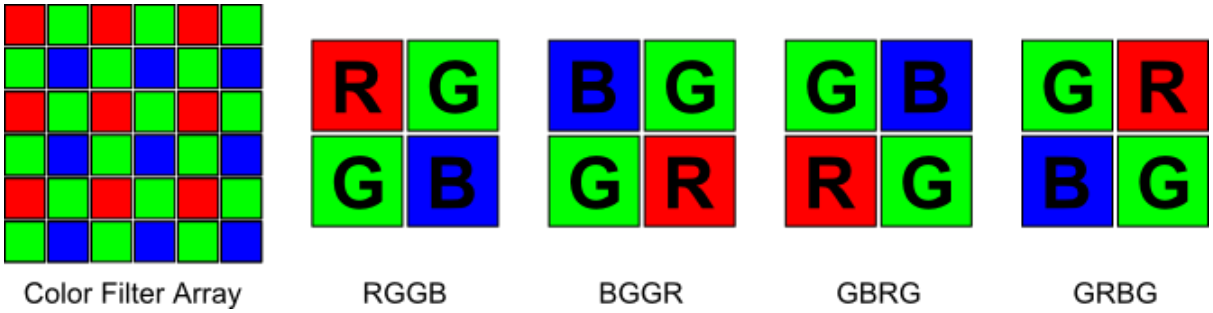
változáson mentek keresztül. A fénysugarak intenzitásának függvényében a felületük, áttetszőségük megváltozik és így megörökítik a képet. A fényképezőgép továbbfejlesztése volt a digitális fényképezőgép megalkotása. Ennél az eszköznél a beérkező fény megőrzését egy szenzor szolgálja, amely a beérkező fénysugarakat digitális adattá formálja. A digitális szenzorok egy Bayer szűrőt használnak.

1.2.1 Bayer szűrő

A Bayer szűrőnél az érzékelőbe érkezett fénysugarakat egy szűrő a piros, zöld vagy kék tartományra szűri. Egy pixel színének meghatározásához az adott pixel környezetében található különböző szenzorok által mért értékeket számoljuk egybe ahogy végül megkapjuk a végső színét egy adott pixelnek. A szemünk érzékenyebb a zöld szín komponens változására igy az egy pixelhez szín szükséges mérésénél négy különböző fényérzékelőt vesszük egybe és ezért a négy érzékelőből kettőt ennek a tartomány érzékelésére fordítanak.



1.2.1.1. ábra: Pixelek a Bayer szűrőben, beérkező fénysugarak egyes tartományának kiszürése (a képen BGGR látható) [3]



1.2.1.2. ábra: Bayer szűrő különböző eloszlásai [3]

1.2.2 Algoritmus

A kamera egy adott szenzorja a Bayer szűrő által átengedett fény erőségét érzékelve meghatározza az adott cellának az értékét és a szomszédos cellák értékéből kiszámítódik az általunk ismert szín érték, ami általában RGB, azaz piros zöld és kék értékek keveréke. Az RGB szín értékben a komponensek értéke általában 0-tól 255-ig terjed a leképzése, azonban ennek a végeredményét sok minden változtathatja, mint például tömörítés. Egy pixel RGB értékének meghatározásához általában a szenzor és a körülötte lévő szenzorokon lineáris interpolációt használunk, különleges esetekben kivétel előfordulhat, mint például hiba javítás vagy alternatív algoritmus használata. Lineáris interpoláció során az azonos az adót cella körüli 3×3-as területen az azonos szint tartományt vizsgáló szenzorok értékét átlagoljuk. Az átlagos két zöld és egy egy piros és kék szenzoros esetben három variáció fordulhat elő eltekintve attól az esetről amikor az érzékelő cella a szélén helyezkedik el a szenzornak. ha a 3×3-as érzékelő közepén egy zöld cella

helyezkedik el akkor a 3×3-as részben öt zöld, két piros és két kék helyezkedik el. Abban az esetben ha a 3×3-as rész közepén piros cella helyezkedik el akkor egy piros és négy-négy kék és zőld cella található a vizsgált részben. Amennyiben kékt cella helyezkedik el akkor forditott a helyzet az előzőhöz képest és csak a kék mintavételből van egy adat a 3×3-as terűleten.

1.2.2.1 ábra Vörös szenzor számitása (értékek fentről-lefele és jobbról-ballra: 200,200,150,255,255,50,250,150,150)

Az 1.2.2.1-es ábrán arra az esetre láthatunk egy példát hogy a középső cella a 3×3-as terűleten piros igy a körülötte lévő cellák kékes és zöldek. Ebből adodoan a piros maximális értékét azaz a 255-őt nem szükséges manipulálni és a körülötte lévő kék és zöld értékek betöltése után elég összeadni és ösztani négyel. mitán a lineáris interpolációt kiszámoltuk a tagokra már csak egyesiteni kell azokat hogy megkapjuk az RGB tagot. Az ábra a könnyebb megértés érdekében a számolást lebegőpontosan végeztem el, azonban ez a valoságban egész értékre van értelmezve, igy a számolás is egészekkel történik.

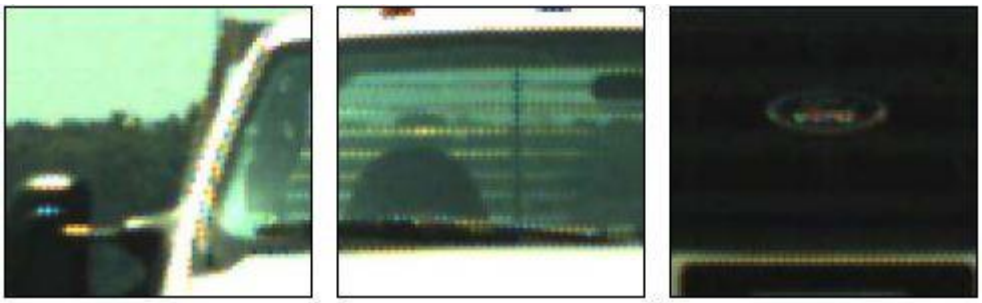
1.2.2.2 ábra zőld szenzor számitás (értékek fentről-lefele és jobbról-ballra: 255,255,255,255,255,100,50,100,50)

A következő példa az előzőhez hasonlóan a lineáris interpolációt demonstrálja, de arra az esetre az példát amikor a zöld komponens található a 3×3-as cella közepén. Az előzőhöz hasonlóan az egyszerüsités kedvéért az egészértéküséget az ábrában lecseréltem folytonos értékekre.

1.2.3 Hibák

Az előzőben tárgyalt lineáris interpolációs közelítésnél az algoritmus jól működik egyenletes változás esetén, de ha hirtelen változás történik a szintéren, akkor hibákhoz vezet ez a lineáris

interpoláció. Erre példák szélek esetén színben egy éles különbség van azonban a képen egy „simítás” tűnik fel, mivel két különböző szín között interpolálunk. Ez a jelentőség kétfajta hibát képes okozni egyik a „False color artifact”, a másik a „Zippering artifact” [3]. False color artifact az a jele amikor egy szél kőrű, ahol a lineáris interpoláció következtében a helytelen színt képez az algoritmus.



1.2.3.1 ábra: False color artifact-re példák, az autó ablaka körül szín eltolódás minta [3]

Zippering artifact az a jelenség, amikor az él mentén elmosodás történik, amely egy jellegzetes zig-zag mintázatot alkot. Az előzővel ellentétben ez a hiba főleg a szélek mentén jelennek meg, míg az előzőnél a szélek körül jelennek meg főként.

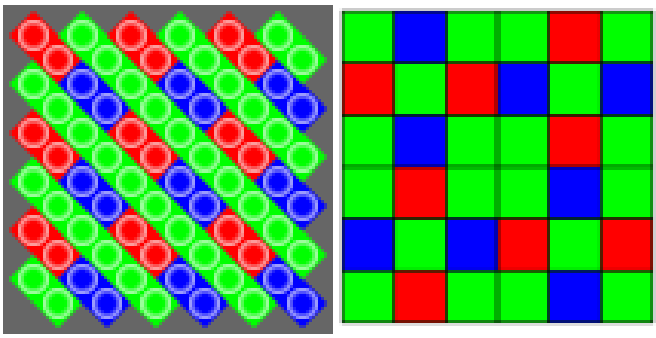
1.2.3.2 ábra: Zippering artifact példa [3]

Mind a két hibára létezik algoritmus, amellyel csökkenthető, esetlenként akár el is tüntethetőek ezek. Ezeknél az algoritmusoknál fontos tulajdonság, hogy képesek a szélek észlelésére és ezek alapján pontosabb számítást végezni.

1.2.4 Változtatások az általános Bayer szűrőn

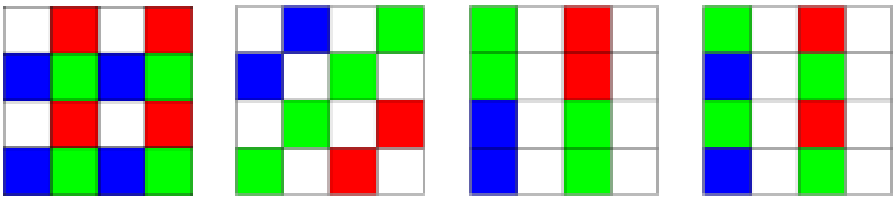
Három különböző csoportra bonthatóak azok a fajta szűrök, amelyeket találtam. Az első csoportam olyan szűrök tartoznak, amelyek maradtak a Bayer szűrő piros kék zöld színű szűrő kompozíciónál, azonban változtattak a különböző szint érzékelő szenzorok elhelyezkedésén.

Az első a Fujifilm "EXR" néven található és a másik meg Fujifilm "X-Trans". A Fujifilmnek az "EXR" szűrnél két egymás mellet azonos színű szűrő helyezkedik el és a zöld egy sorban folyamatosan helyezkedik el. Fujifilm "X-Trans" egy a piros, zöld és kék szűröket egy 6×6-os elrendezésben a zöld szűrök egy X alakzatban helyezkednek el és innen kapta a nevét ez a szűrő.



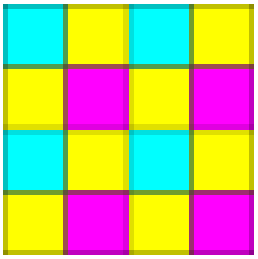
1.2.4.1 ábra: Fujifilm szűrő minták [3]

A Második csoportnál az piros, zöld és kék színek mellé felvettek egy negyediket a fehéret, ami a kettő zöld szűrő egyiket helyettesiti. Ennél a csoportnál még újítás, hogy különböző elrendezést is használtak, amely az eddigi 2×2-es elrendezés helyett 4×4-est használ és igy különböző mintákkal kísérleteztek.



1.2.4.2 ábra: alternatív szűrő konfigurációk [3]

A harmadik a második csoport kiegészítésének tekinthető, ahol a piros, zöld és kék helyett cián, sárga és magenta színeket használ, amit már akár a nyomtatók világából már ismerhetünk. Ezek a színek a fehérrel együtt különböző konfigurációkban előfordulnak, azonban nem találtam összehasonlításokat a különféle szűrők egymáshoz képest milyen előnyökkel és hátrányokkal rendelkeznek.



1.2.4.3 ábra: RGGB helyett CYYM szűrő [3]

1.3 Objektívek

A kamera az objektív segítségével képes összegyűjteni a visszaverődő fényt és azt fókuszálni a fényérzékeny filmre. Ezek az objektívek képesek a kamerába érkező fénysugarak szögét, a látószöget, illetve a fókusztávolságot megváltoztatni. A látószög segítségével változtatni lehet a kamera által látott képet, melynek segítségével nagyítani és kicsinyíteni lehet a képet. Az objektíveknek több fajtáját különböztetjük meg. Különböző stílusú képek elkészítésére más és más fajta objektíveket kell választani. (pl.: portré kép, tájkép, macrokép)

objektívek fajtái:

- fix objektív
- zoom objektív
- nagylátószögű objektív
- halszem objektív
- tele objektív
- macro objektív
- tükör objektív

Ezeken belül megkülönböztethetünk automata fókuszos és manuális fókuszos objektíveket. Készítenek még stabilizátorral felszerelt objektíveket és stabilizátor nélküli objektíveket is.

1.3.1 Fix objektív

Azok az objektívek tartoznak ide, amelyek gyújtó távolsága nem változtatható. Előnye az, hogy ezekkel az objektívekkel nagyobb fényértékkel dolgozhatunk. Az objektíven belül nincsenek mechanikus elemek, ezáltal érhető el a tisztább fényérték. A használata portré fotózáshoz ajánlott.

1.3.2 Zoom objektív

Azok az objektívek tartoznak ebbe a kategóriába, melyeknek a fókusztávolságát manuálisan tudjuk állitani.

1.3.3 Nagy látószögű objektív

Ezeknek az objektíveknek az előnyei, hogy a nagy látószögük miatt közelebbről fotózhatjuk a témánkat, illetve nagyobb lesz a mélységélesség. A nagy látószögű objektív hátránya az, hogy a kép torzítását eredményezi.

1.3.4 Halszem objektív

Azokat a nagy látószögű objektíveket soroljuk ide, melyek látószöge 180°-os vagy annál nagyobb. Ezekkel az eszközökkel különleges képi hatásokat lehet elérni.

1.3.4.1 ábra. Halszem objektív

1.3.5 Teleobjektív

A 60 mm gyújtótávolság feletti objektíveket nevezzük így.

1.3.5.1 ábra teleobjektív.

1.3.6 Macro objektív

Úgy van korrigálva, hogy leképezzen torzítás nélkül 1:1 arányban is.

1.3.6.1 ábra Macro objektív [4]

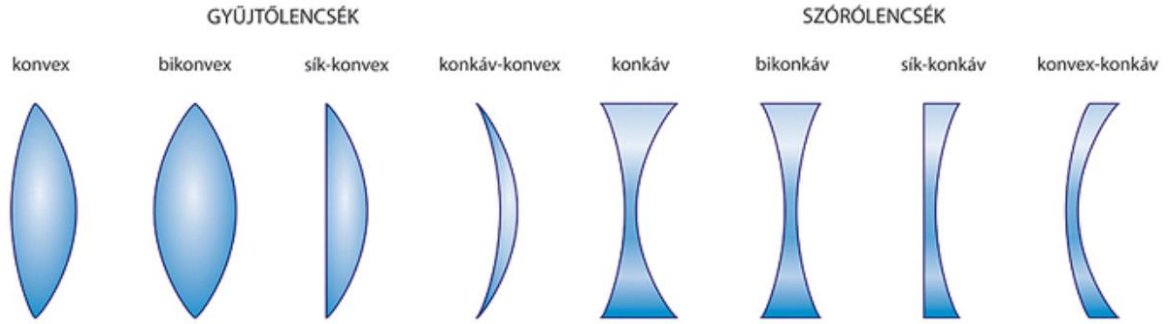
1.3.7 Tükör objektív

Ezek az objektívek 500 és 100 mm gyújtótávolságúak. Ezekben az objektívekben a sok lencse helyett több ponton tükröket helyeztek el és ezekkel a tükrökkel törik meg a fény útját. Nagy hátrányuk, hogy kis fényerővel rendelkeznek, ezért életlenek a színek.

1.4 Lencsék

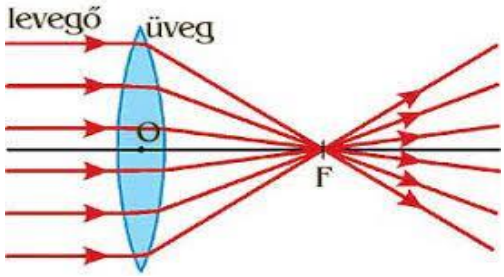
Az objektívek lencsékből épülnek fel, amelyek két gömbfelületből állnak és átlátszó anyagból, optikai üvegből készülnek. A lencse ívét egy gömbközéppont és egy gömbsugár határozza

meg. A gömbnek a gömbközéppontja nem egyezik meg a lencse a gyújtópontjával.Kétfajta lencsetípust különböztetünk meg, gyűjtőlencsét és szórólencsét.



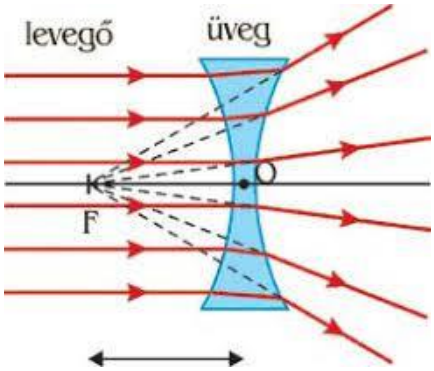
1.4.1 ábra Lencsék.

A domború lencsék (gyűjtőlencsék) azok, amelyek középen vastagabbak, mint a szélüknél. Ide tartoznak a kétszeresen domború (bikonvex) lencsék, sík-domború (plankonvex) lencsék, illetve a homorúan domború (konkáv-konvex) lencsék.A homorú lencsék (szórólencsék) a szélükön vastagabbak, mint középen. Ide tartoznak a kétszeresen homorú (bikonkáv) lencsék, sík-homorú (plankonkáv) lencsék illetve a domborúan homorú (konvex-konkáv) lencsék.A gyűjtőlencsére (domború lencse) párhuzamosan érkező fénysugarak, a lencsén áthaladva megtörnek és egy ponton haladnak át.



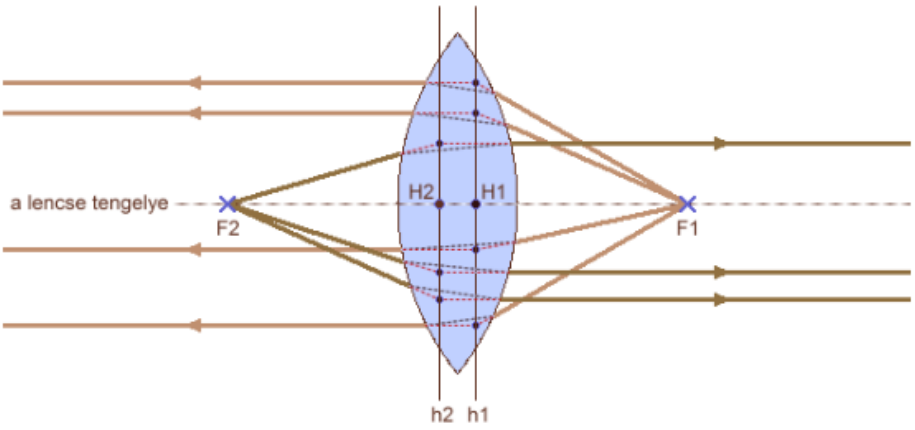
1.4.2 ábra gyűjtőlencse

A szórólencsén (homorú lencse) a párhuzamosan érkező fénysugarak megtörnek és széttartókká válnak.



1.4.3 ábra szórólencse.

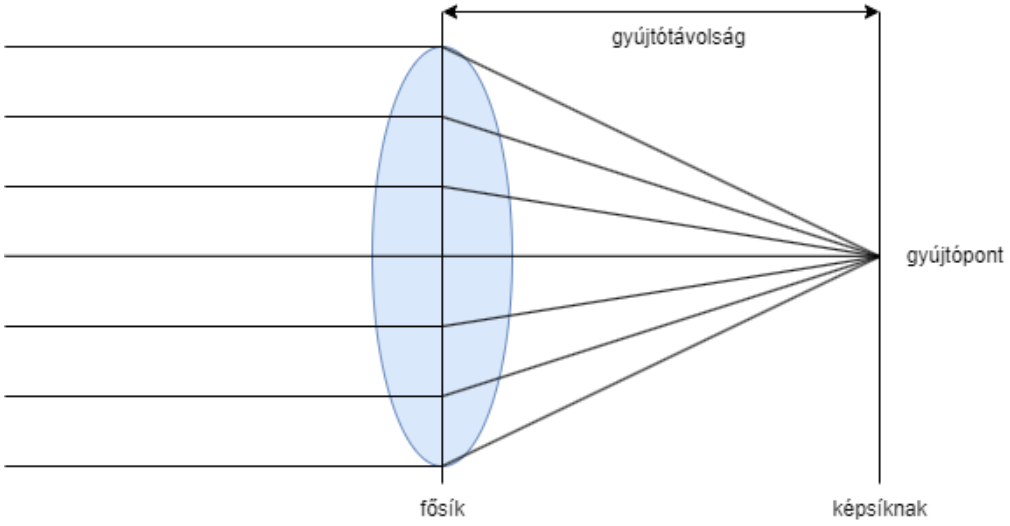
Optikai tengelynek nevezzük a lencsének a két gömbközéppontjának a síkját.



1.4.4 ábra. A lencse tengelyvonal. [5]

1.5 Gyújtótávolság

A lencsék egy fő tulajdonsága. A gyűjtőlencsénél a gyújtótávolság meghatározásához a következő módszerrel végzik. Lencsébe érkező párhuzamos fényt fénysugarak keresztezési pontját vesszük, amit a lencse gyújtópontjának nevezzük és a beérkező fényhez képest a lencse másik oldalán helyezkedik el. A gyújtópont meghatározása után a lencséből ki és be haladó sugarak meghosszabbítása után a sugarak metszeténél kialakuló tengelyt a fősíknak nevezzük. Az adott fősík és gyújtópont távolsága a gyújtotávolság rögzített lencseátméretnél a gyújtótávolság és a látószög fordítottan arányosak. Minél közelebb van a gyújtopont és a fősík annál szélesebb a látószög.



1.5.1 ábra Gyújtótávolság

1.6 Rekeszérték

A rekesz az objektívek belsejében található. Ez a szerkezet „lamellákból” áll, amely egy változtatható méretű nyílással szabályozza azt, hogy az objektíven keresztül mekkora mennyiségű fény érkezzen a fényképezőgép érzékelőjébe. A rekesz átmerőjének változtatásával tudjuk a fény mennyiségét szabályozni. Vagyis, ha adott idő alatt nagyobb nyíláson engedjük be a fényt akkor több, ha kisebb nyíláson, akkor kevesebb fény jut be a kamera érzékelőjébe.

A rekesz értéket f-számokkal jelöljük, melyek lehetnek f/1.4, f/2, f/2.8, f/4, f/5.6, f8, f/11, f/16, f/22, f/32. Minél kisseb az f-szám annál nagyobb a rekesz nyílása.

1.6.1. ábra. rekeszérték. [6]

Az objektíveknek két fajtája van, az egyik a fix rekeszértékű, a másik az állítható rekeszértékű objektív.

A nem fix, azaz állítható rekeszértékű objektíveken a rekeszállító gyűrű elforgatásával vagyunk képesek a rekesznyílást nyitni és csukni. A modern fényképezőgépeken a rekeszértéket az eszköz automatikusan képes beállítani. A rekesznek ismert neve a blende.

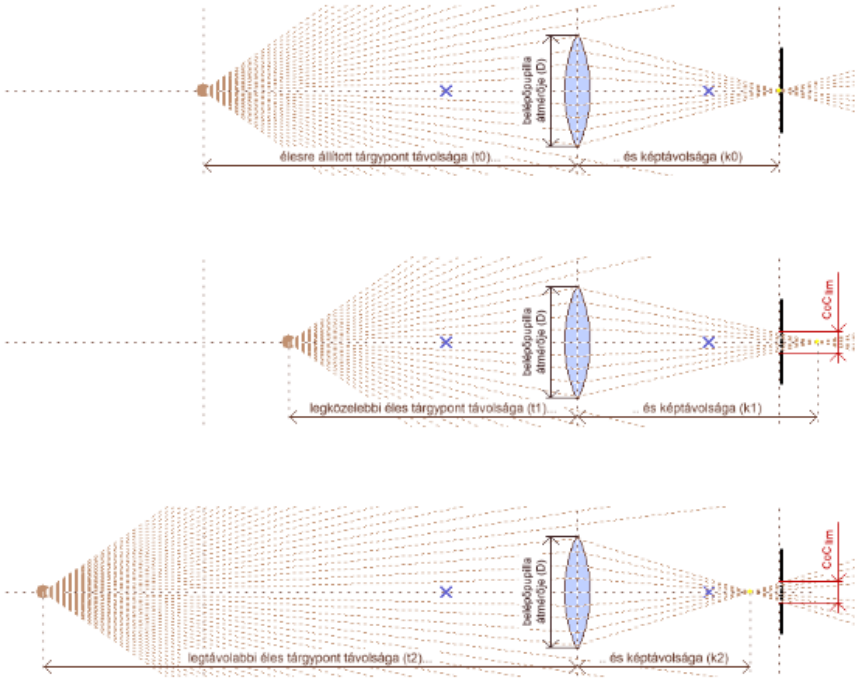
1.7 Záridő

A záridő azt jelenti, hogy mennyi ideig éri a fény a fényérzékeny szenzort, vagyis milyen hosszan exponál a fényképezőgép. Az exponálás értékét úgy vagyunk képesek értelmezni, hogy az adott szám fordítottját, azaz reciprokát vesszük. Vagyis, ha a szám érték 20, akkor az azt jelenti, hogy a kamera érzékelőjét 1/20 másodpercig éri a fény. Tehát a kamera zárja 0.05 másodpercig volt nyitva. A záridő hatással van a kép élességére vagy éppen az elmosódására. Attól függően, hogy mi a célunk a kép megjelenítésével, attól függően állítjuk be a záridőt. Például egy csillagászati fotó esetén minél kisebb számot állítunk be, azaz minél hosszabb ideig exponálunk, akár másodpercek, perceken keresztül, annál élesebb lesz az elkészített fotón a csillagok képe. Ellentétben egy sportrendezvényen, ahol minél nagyobb záridőszámot alkalmazunk annak érdekében, hogy az emberek gyors mozgását megörökítsük. Azonban, ha egy mozgó tárgy esetében kicsi záridőszám kerül beállításra, akkor a mozgó részek elmosódnak a fényképünkön.

1.7.1 ábra kis záridő beállítása [7]

1.8 Szóródási kör

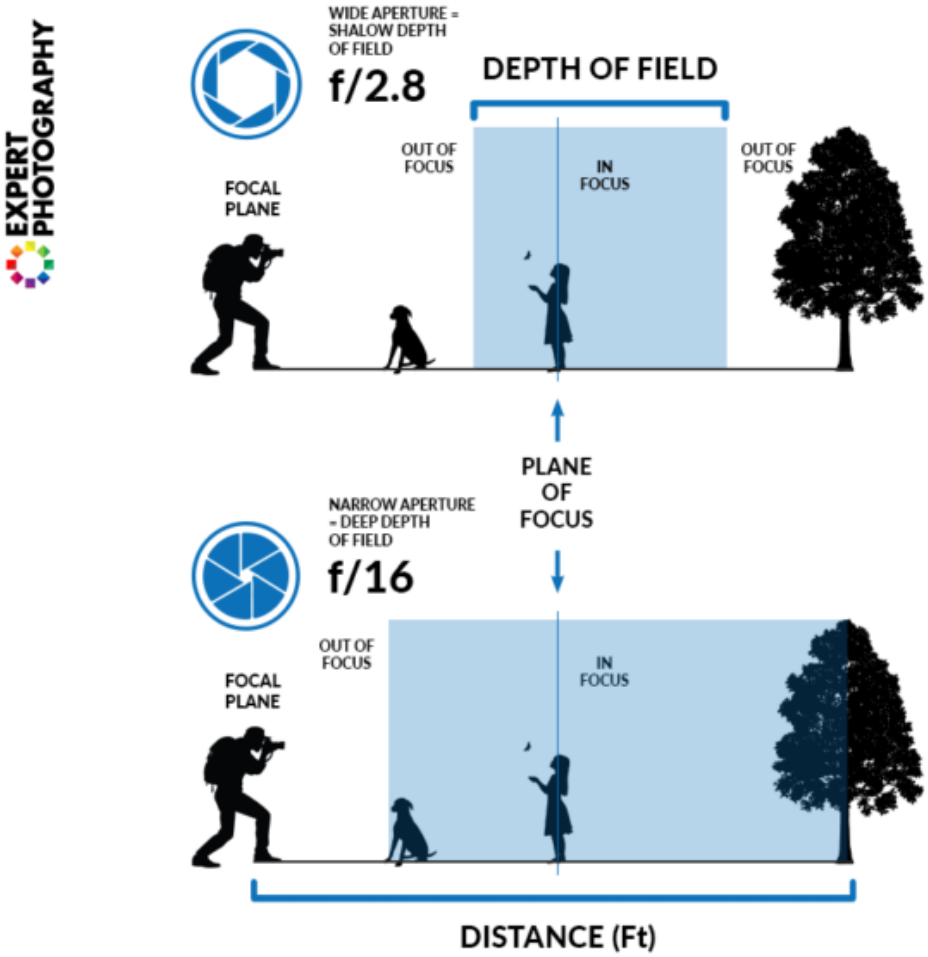
A tárgy egy pontjából kiinduló fénysugarak a lencsén áthaladva a szenzoron nem egy pontba érkeznek meg. Azt a felületet, amelyre a fénysugarak szétterülnek, szóródási körnek nevezzük. A legélesebb lehetőségnél a pontnak a képe a szenzoron is egy pontban összpontosul. Ellenkező esetben a pontnak a képe a szenzoron a fókusztávolságtól távolodva egyre nagyobb kiterjedésű kört képez. A magyar irodalomban a szóródási kör fogalma nem elterjedt, általában az angol terminológiából a Circle of Confusion kifejezést szokták használni. A „Circle of Confusion”-nek a leggyakoribb alakja a CoC, tehát a rövidítése a kifejezésnek.



1.8.1 ábra. Szóródási kör [5]

1.9 Mélységélesség

Mélységélességnek nevezzük azt a távolságtartományt, ahol az objektívhez legközelebb, illetve a legtávolabb levő tárgyak képpontjai még élesen jelennek meg a fényérzékeny film felületén. Leegyszerűsítve a mélységélesség az a távolság a fókuszponttól, ahonnan még éles képet kapunk a környezetről. Azonban a mélységélesség mértéke nem minden esetben egyforma a fókuszponttól vett távolsággal.



1.9.1 ábra. Mélységélesség [8]

A mélységélességet több változó módosítja, ezek a következők:

- a rekesznyílás
- a tárgy távolsága
- a fókusztávolság

Nagy mélységélesség esetén kicsi a fókusztávolság és /vagy nagy a tárgytávolság és /vagy kicsi a rekesznyílás. Kis mélységélesség esetén nagy a fókusztávolság és /vagy kicsi a tárgytávolság és /vagy nagy a rekesznyílás.A magyar irodalomban a mélységélesség fogalma, nem nyíltan használt, általában az angol terminológiából a Depth of Field kifejezést szokták használni. A „Depth of Field”-nek a leggyakoribb alakja a DoF tehát a rövidítése a kifejezésnek. A kis mélységélességet a portré, illetve a makró fotózásnál lehet kihasználni, a nagy mélységélességet leginkább a tájképek fotózásánál szokták használni.

1.9.2 ábra. Kicsi és nagy mélységéleség. [9]

1.10 Bokeh

A bokeh japán eredetű szó, jelentése homály. Ez az effekt a fókusz területén kívül eső részeken jelentkezik. Általában körkörös formában jelenik meg, ezt a jelenséget az objektívnek a zár alakzata befolyásolja. Ezért igyekeznek a professzionális fotósok minél több lamellás objektívet használni a tökéletes kör alakzatok létrehozására, illetve azok széleinek elsimítására. Az effekt befolyásolására az objektív elé különböző alakzatú „zárakat" is helyezhetnek, melyek megváltoztatják az effektus mintáját, például szív vagy csillag forma.

1.10.1 ábra. Bokeh alakzatok

2 Keretrendszer

Keretrendszernek a Unity játékmotort használtam, amelyben C# programnyelv használatával készült a programom. Unity-ben a beépített kameramodult használtam referencia képnek a játéktérről. A kameramodul alapbeállításban nem tartalmaz semmilyen after effect-et, mint például a mélységélesség. Így a kameramodulnak a szimulációjának az eredményén megfelelően látható lesz a változás.

A játékmotornak a 2022-es utolsó frissített stabil változatát választottam, mivel ebben a verzióban már az új program változtatások szerepelnek, amelyeket a későbbiekben a program javításához és gyorsításához használtam.

3 Használt technika

A szimulációhoz a ray tracing működésének elvéből indultam ki. A ray tracking működésének lényege az, hogy életszerűen modellezi le a fény visszaverődését a digitális térben. Az alap ötlet, hogy a kamerába érkező fénysugarat visszakövetve képesek az érintett tárgyak tulajdonságait felhasználva élethűbb hatást elérni. Ezt az ötletet megváltoztatva csak az első érintett tárgyat vizsgáljuk meg, ami elég szabadságot tud nyújtani számunkra a szimulációhoz. Nehézséget okozott a GPU kártya programozásának tapasztalathiánya, ezért nem volt lehetőségem a grafikus kártya használatára a szimulációm során. Az általam elkészített algoritmus a videókártya használata nélkül lassabb lett. Ezt a CPU-n használatának párhuzamosításával igyekeztem felgyorsítani, amire a Unity újabb változata lehetőséget adott. A keretrendszerben a fizikai lekérdezések során vagyok képes az információ kinyerésére, amelyre az algoritmusomhoz szükségem van. A programban ezek a lekérések azok, amelyek a programom lassításhoz vezetnek. A lekérdezések gyorsabb végrehajtására adhat lehetőséget későbbiekben a technologi javítása, vagy a GPU használatának megoldása, azért, hogy a programomban gyorsabb és pontosabb szimulációt érhessek el.

A Unity `raycast` hívása segítségével vagyok képes a játék színterében szereplő elemekből kinyerni az algoritmushoz szükséges információkat. Az így megkapott adatokból kiszámíthatom, hogy a szimulált képnek a fókusztávolsága hol található és milyen mélységélességgel készüljön a kép. Ezek alapján képes vagyok az elkészült adatokon a megfelelő átalakításokkal elérni azt a hatást, amely hasonlít a valóságban készített kép jellemzőire.

4 Algoritmus megtervezése

Az alkalmazás és a dolgozat elkészítéséhez szükséges volt számomra az algoritmus elkészítése, mellyel képes lettem megvalósítani a feladatot.

Az algoritmusom két részre bontható, melyek a következők.

- Az első a kép generálása a szimulált térből a kamera látószögének megfelelően.
- A második pedig az, hogy az első lépésben megkapott képen a mélységélesség szimulálása megtörténjen, mellyel kialakul a végső kép.

A kép generálása során a játéktérben a kamera pozíciójából sugarak segítségével megkaphatóak a pontok, amelyeket a kamera láthat. A látószög függvényében képes a program beállítani a kamerából kifelé mutató sugarak irányultságát, hogy reprezentálni tudja egy valóságos fényképezőgépnek az objektíven át beszűrődő fényt, amelyet aztán az érzékelők felfognak. A kamera pozíciója, az irány amerre a szimulált kamera áll és a látószög arányában képesek vagyunk meghatározni, hogy az egyes sugarak honnan- hová tartanak. Ezek segítségével képezzük a szimulált világban készült képet. Majd miután megkaptuk a sugarak használatához szükséges adatokat, azután a szimulációban kiértékeljük az összes sugarat. Ezekből megkapjuk, hogy az elkészülő képnél az adott pixelnél egy lyukkamera esetén, melyik objektumról érkezne vissza a fénysugár. Miután megkaptuk, hogy az adott sugárral melyik objektumot érintettük, más szóval az adott szögből, melyik pixelhez, melyik objektumról érkező fény jut el, azután az objektum reprezentációja szerint vissza kell keresnünk, hogy az adott pontba az objektum mely pontját érintettük. Az érintett pontból megkaphatjuk azt, hogy az objektum érintett pontjához melyik színérték tartozik.

A második részben a kapott képet, amelyet egy lyukkamera modelljével készítettünk, arra kiszámoljuk a paraméterekből adódóan, hogy a szimulált kamerának a fókuszpontja és fókusztávolsága hol található. Ez alapján képes a program módosítást elvégezni a kapott képen. A második részben szükségem van továbbá arra, hogy az első lépésben kapott képpontok pontjai milyen távol voltak a kamerától. Ezekkel az adatokkal képes a program szimulálni a mélységélességet, amely annak a jelenségnek a következménye, amely során a fókusztávolságon kívül eső pontok nem egy érzékelőn összpontosulnak, hanem eloszlanak több érzékelő között, így képelmosódást eredményeznek. Az elmosódás a fókuszpont

távolságától vett távolsággal arányos. Minél nagyobb a távolság a fókuszponttól, annál elmosódottabb egy pont a képen és minél közelebb van annál élesebb. A lyukkamera modell esetén a pont távolsága nem számít a kép élességében.

5 Alkalmazás megtervezése

Ebben a részben az alkalmazás felépítésének, megtervezésének menetét írom le, mely szükséges volt a működő programom elkészítéséhez. Az alkalmazást a Unity játékmotor segítségével készítettem el. A játékmotor adja a programom keretrendszerét. Erre terveztem meg a rendszerem felépítését, azonban ezek minimális változtatással más játékmotorra is kiterjeszthetők, mint például Unreal és Godot. Ezek a játékmotorok olyan programok, amelyek támogatják a háromdimenziós objektumok importálását és szimulálását az általuk használt térben. Ezek lehetőséget engednek az objektumoknak az UV map-jéhez hozzáférést és a szimulációban szerepelt objektumok vertex-jeinek elérésében. Ezek segítségével szimulálni tudjuk az alkalmazás működését. Azonban elképzelhető, hogy nekünk kell megvalósítani olyan funkciókat, mint a raycast, mely megnehezíti az alkalmazás megvalósítását. Az alkalmazás a program algoritmusa szempontjából két fő részegységre bontható. Szükséges még hozzá venni az alkalmazás tervezéséhez a segédfunkciókat és a keretrendszert, melyek segíthetik a program alkalmazását. Az alkalmazásban a kamera szimulációja mellett szükséges volt egy bemeneti rendszer kialakítására, melynek segítségével irányítható a képalkotó algoritmus indítása, illetve a kamera pozíciója is állítható legyen. Erre azért is szükségem volt, mert az alkalmazás tesztelése és használata ennek a rendszernek a segítségével könnyebbé vált.

Az alkalmazás a Unity terminológiája alapján egy színteret tartalmaz, mely egy szimulált tér a digitális világban, ahol az alkalmazásunk futni fog. Nagy projekteknél segítséget jelent a több színtér alkalmazása, mert több színtér esetén is egyszerre csak egy, az adott aktív színtér van betöltve és a többi színtér elemeit a Unity program felszabadítja. Ez a lehetőség csökkenti a számítógép terhelését és átláthatóbbá teszi a projekteket. A mi általunk készített projektnél a hangsúly a szimuláción volt, ezáltal elég volt egy színtér alkalmazása.

Ezen a téren az elkészített projektemet lehetséges továbbfejleszteni és a továbbfejlesztés által egy nagyobb szabású projekt kezelésére is alkalmassá lehet tenni a projektem. Mivel az alkalmazás a fizika szimulációján keresztül éri el az algoritmushoz szükséges adatokat, ezáltal több színtér alkalmazásával csökkenthető a fizikában szereplő objektumok száma. A bemeneti rendszer, melyet használtam a Unity-nek a nemrégiben

megjelent és támogatott rendszerét használta. A rendszer célja, hogy egy egységes input rendszert hozzon létre, amely könnyen bővíthető. A bővítés után már csak az algoritmus megtervezése során készített terveket kell összekötni a keretrendszerben lévő részekkel. Itt szükséges a sugaraknak megadni a függvény híváshoz szükséges paramétereket, ezek a pozíció, irány és szükséges hozzá még megadni egy maximáslis távolságot, melynek segítségével lehetőség nyílik az algoritmusom futási idejének gyorsítására.

6 Szoftverfejlesztés életciklus

Az alkalmazás készítése során a vízesés modellt használtam, azaz először az elvégzendő feladatok alapján elkészítettem az alkalmazás és az algoritmus terveit, majd elkészítettem az alkalmazást. Végül pedig az elkészült alkalmazást leteszteltem. Megvizsgáltam, hogy a program megfelelőképpen működik, majd pedig megvizsgáltam azt, hogy a feladatban kiírt elvárásoknak a programom megfelel-e. A programom fejlesztését ennek a modell alapján hajtottam végre és készítettem el.

7 Architektúra összefoglalás

A programhoz való tervek készítésénél fontos szempont volt számomra az, hogy az általam megtervezett alapprogram bármikor fejleszthető, bővíthető és kiterjeszthető legyen. Ehhez a programozás során végig ragaszkodtam. Igyekeztem modulárisan felépíteni az alkalmazást ezáltal támogatva a későbbi programfejlesztéseket, illetve figyelembe vettem, hogy a programom alrendszerei újra felhasználhatóak legyenek. Az alkalmazás ezáltal lett felbontva két fő logikai részre, azaz a kép előállítására és a kép utófeldolgozására. Ez által még egy egység behelyezése vagy egy alrendszer kicserélése lehetséges nagyobb rendszer újra dolgozása nélkül, ezzel elősegítve a produktivitást. Az egyes rendszerelemek könnyen cserélhetőségének biztosítása elősegíti az egyes részek könnyebb mérését, javítását és továbbfejlesztését is.

8 Unity raycast

Unity-ben a fizikával foglalkozó motor része a `raycast`, amely a fizikai tér reprezentálásával és azok számolásával foglalkozik. A Unity-nek ezen részén keresztül voltam képes a játéktérben részvevő elemekről lekérni az algoritmus számára szükséges tulajdonságokat. Ezek voltak az objektum azonosítója, amellyel el tudom érni az objektum különböző tulajdonságait, az UV koordinátát, ahol a raycast elérte az objektumot és ez alapján el tudtam érni a színét az adott pontban és a távolságát a kiinduló pontból, amely a mélységélesség szimulálásához szükséges.

```
Texture2D tex = rend.material.mainTexture as Texture2D;
Vector2 pixelUV = _raycastHits[i * sizeOfTheFilterVertical +
j].textureCoord;
pixelUV.x *= tex.width;
pixelUV.y *= tex.height;
rawColorImage[i * sizeOfTheFilterVertical + j] =
tex.GetPixel((int)pixelUV.x, (int)pixelUV.y);
depthImage[i * sizeOfTheFilterVertical + j] = (_raycastHits[i *
sizeOfTheFilterVertical + j].distance)*1000;
```

8.1 kódrészlet adatkinyerés egy raycast-ból

Unity-ben több `raycast` hívása nagy idő felhasználású és ezért amikor több `raycast` használata szükséges bevezették a `RaycastCommand`-ot, amely segítségével képesek vagyunk több `raycast`-ot beállítani és a Unity megpróbálja elosztani a CPU-k között a terhelést a `raycast`-okhoz, ezzel gyorsítva az eredmény megkapását.

```
Ray ray = cam.ScreenPointToRay(new Vector3(Mathf.Round(cam.pixelWidth /
2), Mathf.Round(cam.pixelHeight / 2), 0));
Ray rayOrigine = cam.ScreenPointToRay(Input.mousePosition);
for (int i = 0; i < sizeOfTheFilterVertical; i++) {
    float yOffset = ((sizeOfTheFilterHorizontal - 1) / 2f) - i;
    for (int j = 0; j < sizeOfTheFilterHorizontal; j++) {
        float xOffset = j - ((sizeOfTheFilterVertical - 1) / 2f);
        Vector3 position = ray.origin;
        Quaternion angleRight =
Quaternion.AngleAxis((angleOfViewHorizontal / sizeOfTheFilterHorizontal)
* xOffset, transform.up);
        Quaternion angleDown = Quaternion.AngleAxis((angleOfViewVertical
/ sizeOfTheFilterVertical) * yOffset, transform.right);
        Vector3 direction = angleRight * (angleDown * ray.direction);
        _raycastCommands[(i * sizeOfTheFilterVertical) + j] = new
RaycastCommand(position, direction, queryParameters, viewMaxDistance);
        }
    }
int commandsPerJob = (int)Mathf.Floor(Mathf.Max((sizeOfTheFilterVertical
* sizeOfTheFilterHorizontal) / JobsUtility.JobWorkerCount, 1));
_jobHandle = RaycastCommand.ScheduleBatch(_raycastCommands, _raycastHits,
commandsPerJob, 1, default(JobHandle));
```

8.2 kódrészlet RaycastCommand beállítása

8.1 Fény szimuláció lehetséges javítása

A raytracing megközelítés estén nem a játékmotoron keresztül érik el a 3D objektumokat, hanem egy sokkal gépközelibb állapotban férnek hozzá a 3D objektumokhoz. Az alacsony szintű elérés által képesek kihasználni a GPU párhuzamosítását, amely által nagyságrendű gyorsítás érhető el. A raytracing során egy másodperc alatt több százezer sugárvetést képesek elérni.

9 Látószög szimulálása

Az algoritmus tervezésnél az első részben a látószög és lyukkamerás modell szimulálását tárgyaltuk. Ebben a részben az algoritmus első részének kibővítéséről lesz szó, hogy hogyan alkalmaztam a keretrendszer által nyújtott szolgáltatásokat és hogyan kapcsoltam össze a megtervezett modellt a keretrendszerben használt funkcióval. A kép készítésénél a kamerának a vízszintes és függőleges látószöge alapján és a kamerának a részletessége alapján készíti el a képet. Mivel a program csak a CPU számítási kapacitását használja, a kép készítésének sebessége lassabb és a kép felbontása rosszabb minőségű. A sugárvetésnek a szög számításához

a vízszintes és függőleges szögek paramétereit használtam, amit a valóságban a fényképezőgép kamerájának a lencséinek sorrendje, fajtái és távolságuk határoz meg relatívan egymástól és a fényérzékelőktől. A számítások során a felbontás szerint lineáris beosztás alapján lettek felosztva a sugarak szögei, amelyet a néző irányhoz adtunk hozzá, ezzel globális irányultságot megkapva a lokálisból. Lokálisan kiszámolja a program a szögeket, majd a transzformációval képes a többi objektumhoz hasonlóan a világ színterében értelmezni. Ezáltal az egyenletes eloszlással modellezi le, hogy a lencse külsőjén milyen szögből érkeznének a sugarak, amelyeket a lyukkamera modellel készítünk. Mivel lyukkamerát használunk, így a sugarak számolása szempontjából a kiinduló pontjuk megegyezik az összes sugárnál és csak a világban számolt szögek különböznek az egyes sugaraknál. Az előző részben tárgyaltuk, hogy a sugár vetéshez `raycastCommand` függvényt használtam, amelynél beállítjuk előre az összes sugarat, amelyet asszinkronikusan, majd kiértékel a Unity. Miután a sugarakat végig számolta a program azután képes kinyerni a szükséges adatokat, ami a képkészítés részhez szükséges, azaz az objektumok színértéke és távolsága.

10. Mélységélesség szimulálása

Az algoritmus tervezésénél már beszéltem a mélységélesség szimulálásához szükséges lépéseimről. Ebben a fejezetben kifejtem azokat a lépéseket, amiket alkalmaztam. A mélységélesség eléréséhez szükségem volt a kamerába érkező sugaraknak a tárgy távolságára, azaz, hogy milyen messzire helyezkedik el az objektum a kamerától. Ennek a távolságnak a segítségével és a fényképezőgép tulajdonságaival képes a program kiszámolni, hogy a kamerának hol található a mélységélességnek a távoli és közeli határai. Ezek a pontok meghatározzák, hogy a modell által készített képen mely területek mosódnak el és mely területek maradnak élesek. Ezeket a számításokat az alábbi módon kaphatjuk meg:

focaldistance = a kamera és a tárgy távolsága

flenght= a lencse fókusz távolsága

$$DoFFarMax = (h * focusdistance) / (h - (focusdistance - focusdistance))$$

$$DoFNearMin = (h * focusdistance) / (h + (focusdistance - flenght))$$

1 képlet. mélységélesség

Ahol a *h* a Hyperfocal distance/ hiperfokális távolság jelöli, amelyet szóródási kör és a modell tulajdonságaiból kapunk a következő módon:

fnumber = rekeszérték

$$h = \text{flenght} + (\text{flenght} * \text{flenght}) / (\text{fnumber} * coc)$$

2 képlet. Hyperfocal distance

Miután kiszámoltam a szóródási kört és a modell által szimulált kamera és objektívből származó tulajdonságokat, mégpedig a mélységélesség számot. Ezekből az információkból megkapható, hogy a szimulált kamerának a szenzorján egy adott tárgy pontjainak a képe mennyire élesen jelenik meg. Ezt a számot megkapjuk a használt mértékegységben, ami a programban és a szakmában használt milliméter, hogy milyen kiterjedésű szóródási kör keletkezik mindegyik pontból. Ennek az információnak alapján képesek vagyunk és továbbiakban képesek lennénk a Bokeh hatást alkalmazni az általunk kiszámolt tulajdonságokra. Sajnálatos módon a feladat megoldásánál nem voltam képes kihasználni ezeket a tulajdonságokat és csak közelíteni tudtam az eredményt a szakmában ismert gauss eloszlassál és ennek egy statikus változatát voltam képes alkalmazni ezzel a kívánatos hatást közelíteni.

10.1 Elmosódás eloszlása

A szenzoron felvett képen a mélység élességből kifolyólag elmosódott lesz különböző mértékekben. Az elmosódás az eredeti tárgytól minél távolabb haladva egyre kevésbé látványos. Az elmosódást jól leíró görbe a Gauss görbe. Az elmosódás nagyságától függően használható különböző mértékű Gauss görbék. A görbék kiszámításához figyelembe kell venni a látószög méretarányát és a szenzor méretarányát. Ez által megkaphatjuk, hogy milyen széles és hosszú görbét kell alkotnunk.

ii = távolság a középponttól j tengely

jj = távolság a középponttól j tengely

$$gauss[i , j] = (float)(Math.Exp(-(ii * ii + jj * jj) / (2 * sigma * sigma)) / (2 * Math.PI * sigma * sigma));$$

3 képlet. kétdimenziós Gauss formula

Az ezáltal kapott eloszlás táblázat összege megközelíti az egyet, ami az elvárt, de kisebb nagyobb eltérések lehetnek. Ezért miután az eloszlást kiszámoltam még elvégzek rajta egy normalizálást, hogy garantáltan egy legyen az összegük.

sum=a tömb elemeinek az összege

$$gauss[i ,j] = gauss[i , j] / sum$$

4. képlet. Eloszlás normalizálása

Miután az eloszlás normalizálva lett használhatom, hogy a mélységélességnek az elmosódását szimulálni tudjam. Azonban mivel több különböző elmosódottság fordul elő egy képen a távolságoktól függően ezért több különböző kétdimenziós Gauss görbére lesz szükségem. Ahhoz, hogy ezeket ne kelljen mindig újra számolni egy szótár struktúrát alkalmaztam, ahol a szélesség és magasság alapján eltárolom a különböző eloszlásokat és szükség esetén kiegészítem a hiányzó elemekkel.

```
public class gauss2d {
    private int x;
    private int y;
    private float[,] gauss;

    public gauss2d(int x, int y){
        float sigma = 2.0f;
        float sum = 0f;
        this.x = x;
        this.y = y;
        this.gauss = new float[x , y];
        for (int i = 0; i < this.x; ++i) {
            int ii = i- Convert.ToInt32(Math.Floor((float)this.x / 2));
            for (int j = 0; j < this.y; ++j) {
                int jj = j - Convert.ToInt32(Math.Floor((float)this.y /
2));
                this.gauss[i , j] = (float)(Math.Exp(-(ii * ii + jj * jj)
/ (2 * sigma * sigma)) / (2 * Math.PI * sigma * sigma));
                sum += this.gauss[i , j];
            }
        }
        for (int i = 0; i < this.x; i++)
        {
            for (int j = 0; j < this.y; j++)
            {
                this.gauss[i ,j] = this.gauss[i , j] / sum;
            }
        }
    }
    public float getX(){ return this.x; }
    public float getY(){ return this.y; }
    public float getGauss(int x, int y) {
        if (x>0 && x<this.x && y>0 && y<this.y)
        {
            return this.gauss[x , y];
        }
        return 0;
    }
```

10.1.1 kódrészlet kétdimenziós Gauss görbe eloszlás osztály

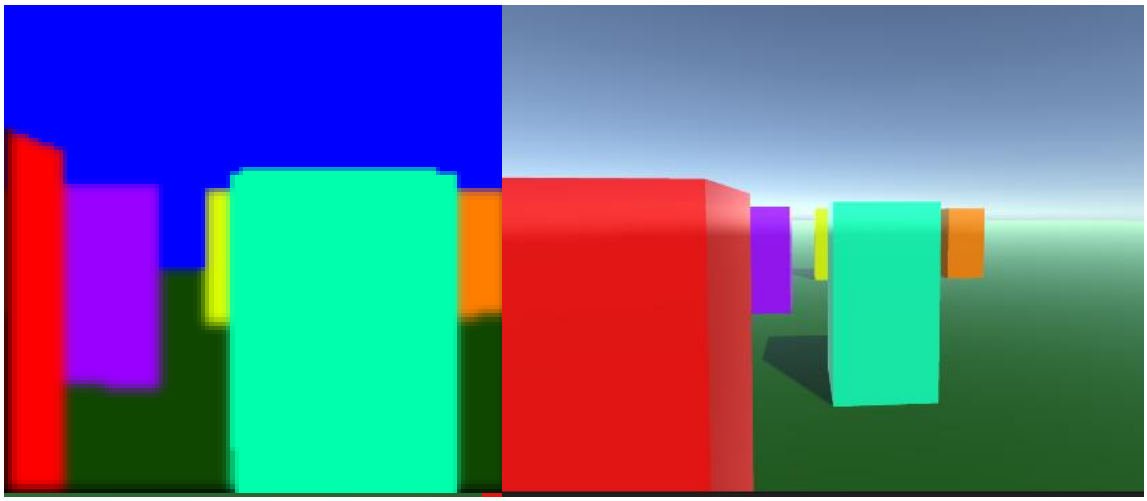
Hogy kihasználjam a programozási nyelv generikus programozás képeségét, hogyhasználni tudjam a már létező `Dictionary` sablon osztályt. A sablonban a `Vector2D` osztályt használtam, mint kulcs és a saját osztály, mint tárolando osztályt.

```
Dictionary<Vector2, gauss2d > gauss2DDictionary = new Dictionary<Vector2,
gauss2d > ();
//képontok vizsgálata és tagváltozok kiszámitása minden pixelre
if (cPerSensorx <= 1 || cPerSensory <= 1) {
    continue;
}
if (!gauss2DDictionary.ContainsKey(new Vector2(cPerSensorx,
cPerSensory))){
    gauss2DDictionary.Add(new Vector2(cPerSensorx, cPerSensory), new
gauss2d(cPerSensorx, cPerSensory));
}
gauss2d currentGauss = null;
gauss2DDictionary.TryGetValue(new Vector2(cPerSensorx, cPerSensory), out
currentGauss);
//szükséges cellák átlagolása
```

10.1.2 kódrészlet Gauss eloszlás osztály előkészítése

11. Program eredményének bemutatása

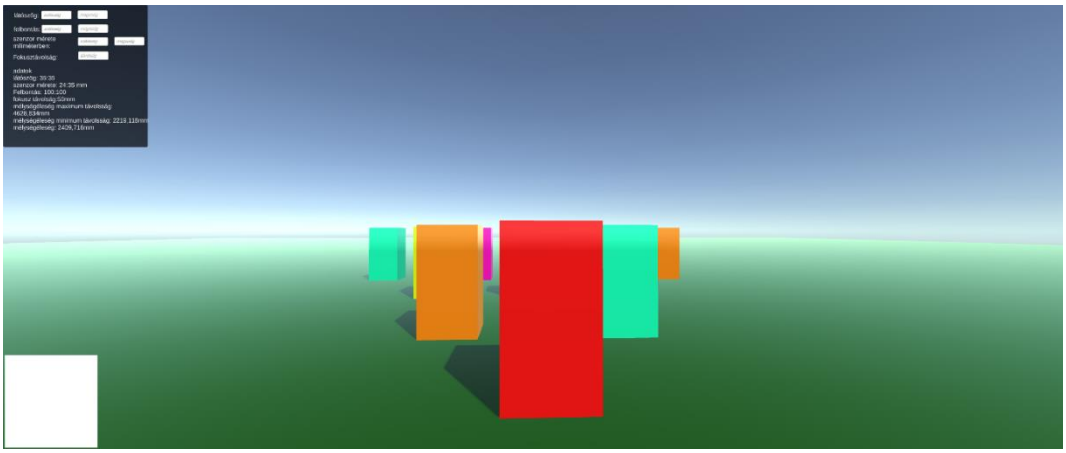
Az alkalmazásban a Unity editor segitségével vagyunk képesek beállitani a kamera modell paramétereit. A lent látható képnél a középső kép a játék motor által szimulált kép, ahol az objektumok kristály élesek mivel nincs mélységélesség a képen. A kép bal alsó sarkában egy 100×100 pixeles változatot látunk, amelyet az algoritmus készitett. A képen láthatóak azok, hogy a háttérben lévő színes objektumok életlenek. A kép készítése során a látószög vertikálisan 30°-os és horizontálisan 40°-os szögeket zárnak be a szimulált képhez képest egy keskenyebb képet kapunk, melyet a szimulálthoz képes nagyítottnak érzünk.



11.1. ábra. bal a készített kép és jobb az általunk látott színtér

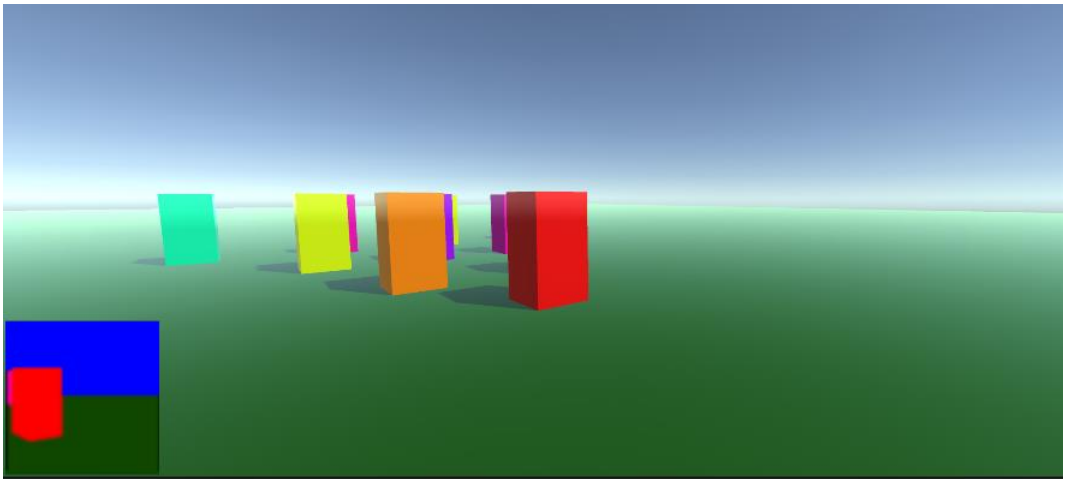
A képen láthatjuk, hogy a piros és a zöld alakzat még túl közel van a kamerához, a többi alakzat meg már túl távol van a kamerától. Így mindegyik alakzat elmosódott a képen és az éles tartomány a két alakzatcsoport között helyezkedik el.

Az algoritmus során a program kiír több értéket. Az alsó képen látható, hogy a RaycastCommand során az egyes egységtömbben, amelyet a CPU-n futatni fog, azokban egyszerre hány darab raycast kerül kiértékelésre. Itt látható, hogy ebben az esetben egy időegységben 1428 kiértékelés került beütemezésre, amelyet azután a végeredmény táblába visszaír. Ebben a szimulációban a kamerának a fókuszatávolsága a kamerától 3 méterre helyezkedik el, ezáltal a program kiszámolta, hogy a mélységélesség távoli határa 4,6 méter körül helyezkedik el és a mélységélesség közeli határa 2,2 méterre található. Ezáltal látható a három métertől vett egy a kettőhőz arány a mélységélesség két határa között és igy a mélységélességi szám 2,4 méter, mely a kettőnek a különbségéből adódik.



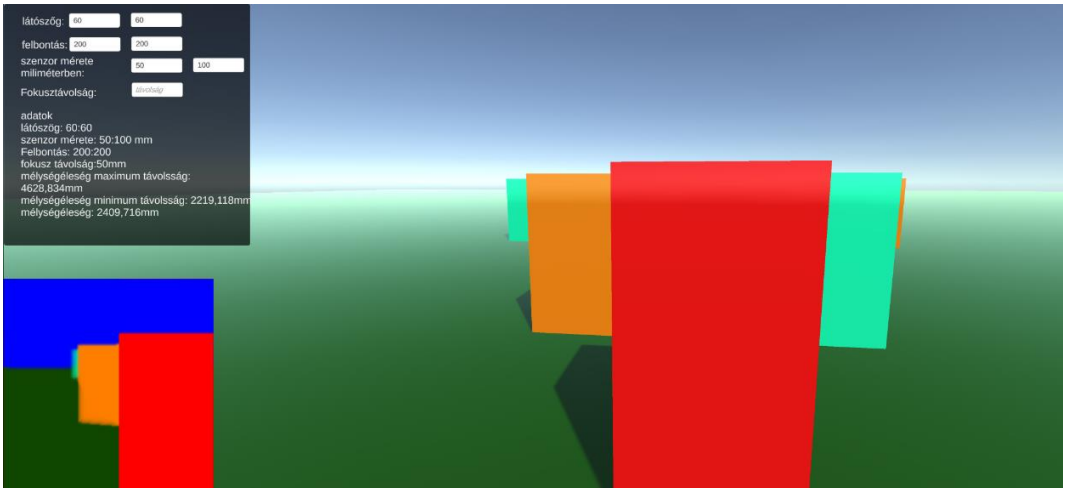
11.2. ábra. Adatbeviteli mező balfelső sarok és bal alsó sarok leendő kép helye a UI-on

A következő képen láthatjuk, hogy a szimulált kép középén található alakzat fókusztávolságban van. Ez a távolság kb 3 méter mely területen a kép éles maradt, míg a háttérben a zöld talaj és a kék ég összemosódott. Az alacsony látótávolságnak köszönhetöen csak a piros alakzat szerepel a készített képen.



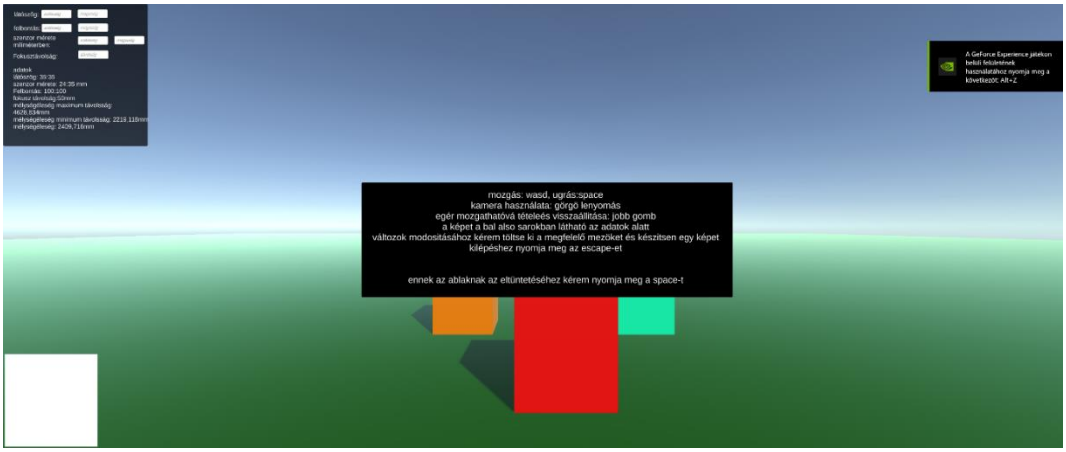
11.3. ábra. Bal alsó sarokban látható a kis látószögű kép, ami készült a színtérről a UI elrendezésben az adatok nélkül

A következő képen a kamera hátrafelé mozdult és az újonnan készített képen a piros alakzat szélei már egybemosódnak a környezettel. Az előző képpel összehasonlítva jól láthatjuk az elmosódott és éles képek különbségét.



11.4. ábra. közeli mélység éleség

A projekt elinditásakor egy információs oldal tünik fel ami röviden összefoglalja a projektben található vezérlési elemeket. A képernyő bal felső sarkában található az adatok beviteléhez tartozó mezők amely adat felvétele esetén automatikusan frissitik a rendszer elemeit és kép készitése esetén a részleteket kijelző oldal pedig ki irja a kép készitése során használt paramétereket és információkat. A projekt irányitása számitogépet és egér billentyü használatához készült.



11.5. ábra. kezdőképernyő UI felülete

12. Összefoglaló

A feladat elvégzése során képes voltam elérni a programom segítségével egy olyan állapotot, amin látható, hogy a modell miként alkalmazza a szakirodalomban ismert jelenségeket. A látószög változtatásával képesek vagyunk különböző beállításokat megtapasztalni, ilyen például a halszem effektus vagy ismertebben a fish eye effect. Melynek estében az általunk megszokottnál sokkal szélesebb a látószög, ezáltal az alkotott képet torzítottan látjuk. Illetve látható a mélységélesség megközelítése, amelyet a programban alkalmaztam. Ezzel a programmal és a dolgozattal, azt az érdekes témát jártam körbe és mutattam be, hogy a kamera által készített fényképnek a szimulálása egy digitális környezetben hogyan történne. Alapkövetelmény volt, hogy a kamera fizikai jellemzőit és a természet fizikai törvényeit az általam készített modell tartalmazza. A programmal elkészített fényképnek tükröznie kellett azok fizikai tulajdonságait. A dolgozatom során számomra láthatóvá vált a keretprogramnak választott játékmotor használatának nehézségei, akadályai és az, hogy ez a módszer egy járható út a digitális világban készített kép realisztikus elkészítésére. A dolgozatom során ismertettem, hogy a valóságban megismert tulajdonságok modellezését miképpen lehet a digitális világban még pontosítani.

Irodalomjegyzék

1. https://eta.bibl.u-szeged.hu/1257/2/modern_ikt_a_sportban/Modern_ikt_a_sportban/www.jgypk.hu/tamop15e/tananyag_html/Modern_ikt_a_sportban/camera_obscura.html

2. https://www.sumidamagazin.com/2019/06/06/rest_insert-217/

3. https://en.wikipedia.org/wiki/Bayer_filter

4. https://hu.wikipedia.org/wiki/Objekt%C3%ADv_%28f%C3%A9nyk%C3%A9p%C3%A9szet%29

5. Fotóelmélet: Optika 1 - Pixinfo.com

6. https://jancsofotosuli.com/rekesz.html

7. https://jancsofotosuli.com/rekesz.html

8. https://www.videomaker.com/how-to/shooting/composition/everything-you-need-to-know-about-depth-of-field/

9. https://photonify.com/3-steps-for-adjusting-the-depth-of-field-on-your-camera/

10. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/PostProcessEffects/DepthOfField/

11. https://docs.unity.com/

12. https://en.wikipedia.org/wiki/Depth_of_field

Nyilatkozat

Alulírott Galgóczki Norbert programtervező informatikus MSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Képfeldolgozás és Számítógépes Grafika Tanszékén készítettem, programtervező informatikus MSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Diplomamunka Repozitóriumában tárolja.

Dátum 2024.0.5.14.

Aláírás