Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozáselmélet és Szoftvertechnológiai Tanszék

Stilizált 3D szemantikus látvány adott időpontban, adott GPS lokáción

Szerző:
Poros Tamás Gábor
programtervező informatikus BSc

Belső témavezető:
Fábián Gábor
egyetemi adjunktus, PhD

Külső témavezető:
Hiba Antal
kutató, PhD

Budapest, 2023

Tartalomjegyzék

1. fejezet

Bevezetés

Építészmérnökként szerettem volna az eddigi szakmai pályafutásomhoz kapcsolódó témát választani. Az építészetben a térkompozíció alapvető fontosságú, hiszen az épületek és építmények kialakítása során azok formai és térbeli összefüggéseit kell megtervezni. A tervező az ábrázoló geometria szabályaira támaszkodva készíti el a terveket. Az 1960-as évek előtt párhuzamvonalzóval, papírra rajzolt tustervek készültek. 1963-ban Ivan Sutherland doktori disszertációjában bemutatta a Sketchpad számítógépes programot, amelyet a mai számítógéppel támogatott tervezési (CAD) programok őseként tartanak számon. Mára pedig eljutottunk az épületinformációs modellezésig (angolul *Building Information Modeling*, továbbiakban BIM), ahol egy létesítmény fizikai és funkcionális jellemzőit digitális formában ábrázoljuk. Jelenleg mind az építészeti tervek, mind a szakági tervek, mind az anyagkimutatás, de a tervek ellenőrzése is a BIM modell segítségével történik. A mérnöki tevékenységek, főként azok ábrázolásai szoros összefüggést mutatnak a számítógépes grafikával. Kézenfekvő volt, hogy a számítógépes grafika témakörében keressek magamnak témát.

A szakdolgozat keretében a SZTAKI önvezető drónok fejlesztésével foglalkozó részlegének munkájába kapcsolódtam be, az általam készített program a kutatás egyik elemét képezi. A szoftvert az automata drónok fejlesztésében részt vevő kutatók arra fogják használni, hogy segítsen a drónok helyzetének validálásában, hibák kiszűrésében. Mindezt úgy, hogy a kutatók a program által készítetett képeket összevetik a drónok által repülés közben készített felvételekkel.

Az önvezető drónok fejlesztése az utóbbi években egyre nagyobb figyelmet élvező kutatási terület. Az ilyen drónok alkalmazása széles körben elterjed, többek között

használják épületek felügyeletére, mezőgazdasági munkák elvégzésére, csomag kézbesítésre, katonai és természetvédelmi feladatokra is. Az önvezető drónok további fejlesztése érdekében számos kutatói csoport dolgozik azon, hogy az automata repülőgépek egyre pontosabbak, hatékonyabbak és biztonságosabbak legyenek.

A kijelölt célterület felett a drónok repülés közben fedélzeti kamerájukkal képeket készítenek, amelyeken kutatók szemantikus szegmentációt hajtanak végre. A szegmentálást végrehajtó program előre definiált logikai osztályokba sorolja a fénykép pixeleit. A célterületről Lidar, aktív távérzékelési technológiával georeferált ponthalmazt (pontfelhőt) készítenek. A pontfelhő a felszín és a felszínen lévő objektumok (épületek, távvezetékek, fák, stb.) magassági értékeit jelenti. A lézerszkennelt ponthfelhőből pedig háromszögelt térbeli hálót generálnak. A háromszögelt térháló elemeit egy neurális háló szemantikusan szegmentálja, a térbeli modell elemeit a megfelelő logikai osztályokba csoportosítja.

A szakdolgozatban szereplő szoftver feladata, hogy a területről készített 3D-s háromszögelt térhálón, a Nap pozíciójának ismeretében, meghatározott kamera útvonal mentén szimulációt hajtson végre. A szimuláció során a megfelelő fénybeállításokkal a szegmentált térhálóról felvételeket készít.

A szoftvert felhasználva a kutatóknak lehetősége van a drónok által készített szemantikusan szegmentált fényképek és a program szimulációs képeinek összehasonlítására, így valósítva meg a kamera alapú navigációs szenzor hiba detekcióját. Az összehasonlítást végző szoftver nem a szakdolgozat kereteiben készül. A összehasonlítás eredményeként meghatározható, hogy a drón a tervek szerint halad-e, vagy eltér a megírt repülési tervtől.

2. fejezet

Elmélet

A program feladata, hogy egy szemantikusan szegmentált térhálóról adott időpontban, adott GPS lokáción felvételt készítsen. A program először a saját, derékszögű világ-koordináta rendszerét tűzi ki. Ehhez meg kell határoznunk a koordináta rendszer origóját, amely értékül az adott GPS lokációt kapja. Majd a koordináta rendszer tengelyeit megfeleltetjük az északi és nyugati iránynak. A világ-koordináta rendszerben az időpont és a lokáció alapján szimulálnunk kell a modell környezetét, ezt úgy érhetjük el, hogy azonosítjuk a Nap helyzetét. A paraméterek alapján meghatározzuk a Napból származó párhuzamos megvilágítás irányát. A felületháló modelleket a saját lokális koordináta rendszerükben szokás ábrázolni. A felülethálót világ-transzformáció segítségével a programunk saját, világ-koordináta rendszerébe kell illeszteni. Az ehhez szükséges elméleti ismeretek ebben a fejezetben kerülnek ismertetésre.

2.1. Az idő ábrázolása

Az idő ábrázolására többféle rendszer működik. A különböző ábrázolási módok eltérő célterületeknek felelnek meg. Az ábrzolási módszerek közötti átjárás biztosított, azonban nem minden esetben lehetséges az egy-egy értelmű megfeleltetés.

2.1.1. Egyezményes koordinált világidő, UTC

Az egyezményes koordinált világidő (Coordinated Universal Time, UTC) az órák és az idő világszerte történő szabályozásának fő szabványa. 1961-ben váltotta fel a greenwichi középidőt (GMT). Az UTC az egyenletesen mért nemzetközi atomidőből

származik. Az UTC nem veszi figyelembe a nyári időszámítást, ezért az nem változik a napfordulók alkalmával. Az egyes világidőzónák pozitív vagy negatív eltolásokkal vannak meghatározva az UTC-hez viszonyítva. Bár az UTC-t nem befolyásolja a nyári időszámítás, azonban a helyi idő változhat a nyári időszámítás miatt. Emiatt az UTC és a helyi idő közötti eltérés évszakonként változhat. Magyarország télen UTC+1 időt, nyáron pedig UTC+2 időt használ. Az UTC az időt napokra, a napokat órákra, percekre és másodpercekre bontja fel. Minden nap 24 órából áll, és minden óra 60 percből. Azonban az 1 percen belüli másodpercek száma változó lehet. Az UTC napok többségénél minden perc 60 másodpercből áll, azonban bizonyos esetekben egy nap utolsó perce állhat 59 vagy 61 másodpercből is. A másodperc és a másodpercnél kisebb időegységek mind állandó időtartamúak. Az átlagos UTC nap pontosan 86 400 SI másodpercet tartalmaz, percenként pontosan 60 másodperccel. Azonban szoláris középnap valamivel hosszabb, mint 86 400 SI másodperc. Ezért van szükség arra, hogy időnként az UTC nap utolsó perce 61 másodpercre módosuljon. Az extra másodpercet nevezzük szökőmásodpercnek. Egy UTC nap utolsó perce 59 másodpercet tartalmazhat, hogy fedezze a Föld gyorsabb forgásának távoli lehetőségét, de erre még nem volt szükség. Ennek köszönhetően az UTC körülbelül egy másodpercen belül van a Föld forgásából számolt egyezményes világidőhöz (Universal Time, UT) képest a 0° hosszúságon.

Az UTC napok megfeleltethetőek a Gergely-naptár szerint. Azonban a csillagászatban a számításokhoz a Julián-dátumot használják, mert a szökőmásodpercek miatt az UTC nem folytonos, és ez a számításokat nehezen kivitelezhetővé tenné. Az UTC szabálytalan naphosszai miatt a töredékes Julian-napok és az UTC közötti átváltás bizonyos számításokhoz nem kellő pontosságú.

2.1.2. Unix-idő

A Unix-idő az UTC szerinti 1970. január 1. 00:00:00 óta eltelt másodpercek száma. A Unix-idő mindennap 86 400 másodperccel nő, a szökőmásodperceket is figyelembe véve. A Unix operációs rendszerek mellett számos alkalmazásban használatos. Az előző fejezetben ismertetett szökőmásodpercek miatt az Unix-idő nem valódi reprezentációja a UTC-nak. A két ábrázolás között nincs egy-egy értelmű megfeleltetés. Minden szökőmásodperc az őt közvetlenül megelőző vagy követő másodperc időbélyegét használja. Ha egy UTC nap nem pontosan 86 400 másodperc

hosszú a szökőmásodpercek miatt, akkor az Unix-időben egy szakadás következik be, mert az Unix-idő pontosan 86 400-zal nő minden nap, függetlenül a nap hosszától. Ha egy nap 86 399 másodperc hosszú lenne a szökőmásodpercek miatt, akkor az Unix-idő egy egységet ugrik azon a helyen, ahol a szökőmásodperc eltávolításra került. Ez azonban eddig még nem történt meg. Ha beszúrnak egy szökőmásodpercet, azaz egy UTC nap 86 401 másodperc hosszú lenne, akkor az Unix-idő folyamatosan növekszik a szökőmásodperc ideje alatt, majd a szökőmásodperc végén egy egységet visszaugrik.

2.1.3. Julián-dátum

A csillagászati számításoknál használt Julián-dátum a Kr. e. 4713. év első napjától eltelt napok számával és óra-perc-másodperc helyett a nap decimális törtrészeivel adja meg az időpontokat. Mint bármely naptártól független periódus, a Julián-dátum közvetítõként használható egyik naptári rendszerrõl egy másikra való gyors átszámításhoz. A Julián dátum kezdete Kr. e. 4713. január 1 dele. Ezt azonosítják a 0. Julián nappal. Minden további napot egy egész számmal azonosítanak, amely azt jelzi, hogy az első Julián naptól az adott napig hány nap telt el. Ezt a egész számot nevezzük Julián napnak. (Julian day number, JDN). A Julián dátum (Julian date, JD) egy Julián nap plusz a megelőző nap delétől eltelt idő decimális törtrészben kifejezve.

2.2. Koordináta rendszerek

A navigációs rendszerekben a repülőgépek mozgásának leírására többféle koordináta rendszer használatos. Ennek oka, hogy a különböző jelenségek leírása egyszerűbb különböző koordináta rendszereket alkalmazva.

2.2.1. Föld-központú, Földhöz rögzített koordinátarendszer

A Föld-központú, Földhöz rögzített koordinátarendszer (Earth-centered, Eart-fixed, ECEF), más néven geocentrikus koordináta-rendszer, egy derékszögű térbeli vonatkoztatási rendszer. A koordináta rendszer origóját a Föld középpontjához rögzítik, Z tengelye az északi pólus felé mutat és azon halad át, X és Y tengelye az egyenlítő síkjában fekszik és X tengelye a fő meridiánon (greenwichi délkör) halad

át. Segítségével Föld felszín alatti és feletti pontok is ábrázolhatók az X,Y,Z koordináták megadásával. A GPS alapvetően ezt használja, mert nem szükséges hozzá egy ellipszoid modell használata, csak a Föld középpont pozíciójának és a rendszer orientációjának ismerete.

2.2.2. Földrajzi koordináta-rendszer

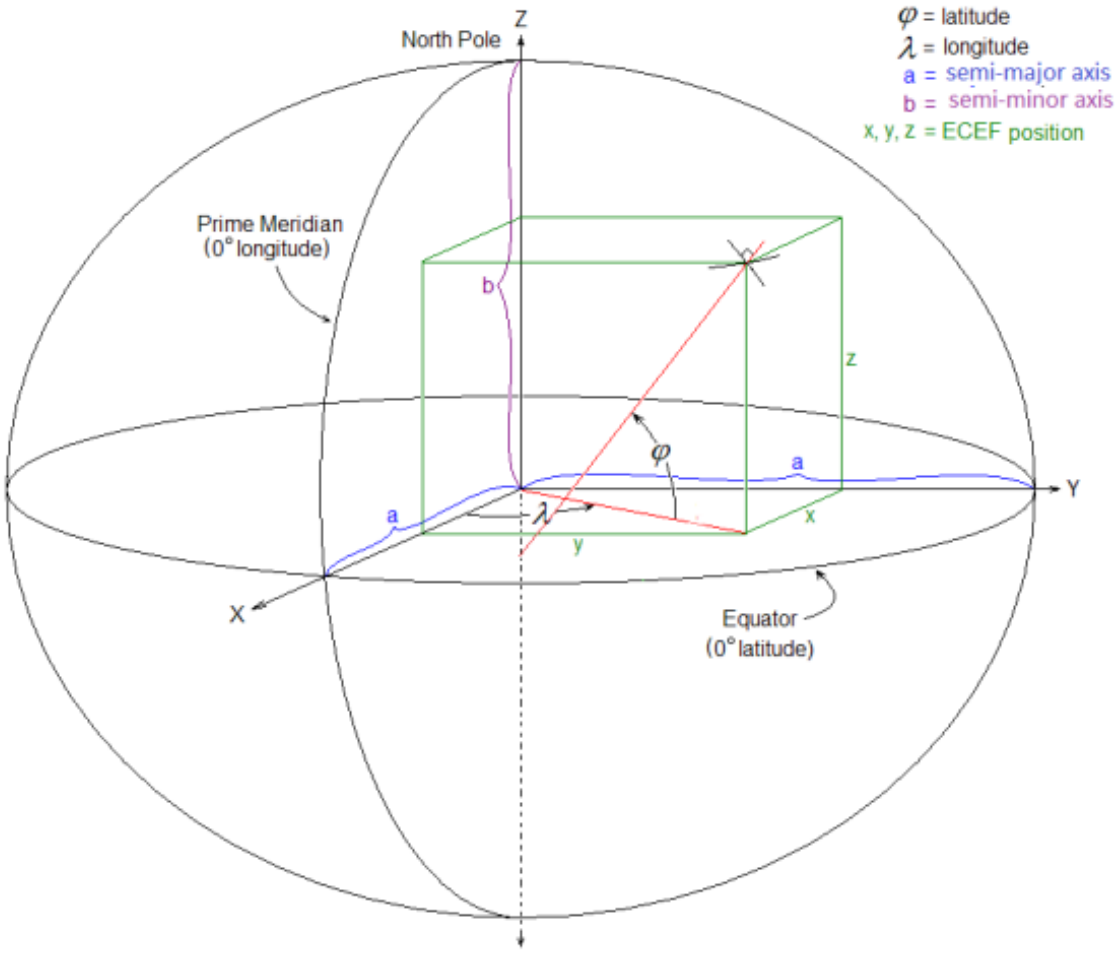
A földrajzi koordinátarendszer (geographic coordinate system, GCS) egy gömb- vagy ellipszoid koordináta-rendszer, amely a Földön lévő pozíciók mérésére szolgál, szélességi és hosszúsági fokban kifejezve. A referencia modelltől mért távolságot pedig a magassággal jelöljük, jele: h. A három koordinátát a gyakorlatban szokás LLA (Latitude, Longitude, Altitude azaz szélesség, hosszúság, magasság) koordinátáknak nevezni.

Földrajzi szélesség

Földrajzi szélesség angolul latitude, jele: ϕ.
Egy adott P pont szélessége egyenlő a ponton és a Föld középpontján átmenő egyenes és az Egyenlítő síkja által bezárt szöggel fokban kifejezve. Megállapodás alapján északi irányba pozitív, déli irányba negatív az érték előjele. Az azonos szélességű pontok egyazon szélességi körön vannak. A szélességi körök által meghatározott síkok páronként párhuzamosak. Néhány nevezetes szélességi kör: Egyenlítő: $\phi = 0°$, Északi-sark: $\phi = +90°$, Déli-sark: sark $\phi = -90°$.

Földrajzi hosszúság

Földrajzi hosszúság angolul longitude, jele: λ.
Egy adott P pont földrajzi hosszúsága egyenlő a pont és a két pólus által meghatározott sík, meridiánsík és a kitüntetett kezdő meridián sík által bezárt szög fokban kifejezve. Megállapodás szerint keleti irányban pozitív, nyugati irányban negatív Az azonos hosszúságú pontok alkotta azonos hosszúsági körön helyezkednek el. A hosszúsági körre másik szokásos terminus: a meridián. A kitüntetett kezdő hosszúsági kör, $\lambda = 0°$ a greenwichi obszervatóriumon (Royal Observatory, Greenwich) halad keresztül. A hosszúsági körök a definíció miatt nem azonos hosszúságúak és nem párhuzamosak.



2.1. ábra. Az ECEF koordináták a földrajzi szélességhez és hosszúsághoz viszonyítva

WGS84

A WGS84(World Geodetic System) egy geodéziai világrendszer, egyezményes földi vonatkoztatási rendszer, amely magába foglalja a Föld normálalakjának és méreteinek leírását, a Föld gravitációs (EGM) és a mágneses erőtérének modelljét (WMM), és a földi vonatkoztatási koordinátarendszert. A WGS84 szabvány a Földet egy ellipszoid-modell-lel közelíti, az ellipszoid modell paraméterei a 2.3.1-es számú fejezetben olvasható.

A GPS-műholdak által sugárzott fedélzeti pályaadatok vonatkoztatási rendszere WGS84 geoid modellt alkalmazza.

2.2.3. NED (North, East, Down, azaz észak, kelet, lefelé) rendszer

A földfelszín egy adott pontjához, (pl egy adott GPS koordinátához) rögzített helyi koordináta rendszer, melynek 1. számú, N tengelye az ellipszoid északi pontja, 2. számú, E tengelye az ellipszoid keleti iránya, 3.számú, D tengelye pedig az ellipszoid belseje felé mutat a helyi normálvektor mentén. Általában kis távolságú, rövid idejű repülések koordináta rendszereként alkalmazzák.

2.2.4. Grafikus API-k koordináta rendszere

A 3D grafikus alkalmazások jellemzően bal, illetve jobbsodrású derékszögű koordinátarendszert is használnak. A Direct3D alapértelmezett módon balkezes koordinátarendszert használ.

2.3. Koordináta rendszerek közötti transzformáció

2.3.1. Transzformáció WGS84-ből ECEF-be

A transzformáció végrehajtásához ismerni kell a WGS84 által használt ellipszoid modell paramétereit:

fél nagytengely: $a = 6378137.000m$

fél kistengely: $b = a \cdot (1 - f) = 6356752.314m$

ellipszoid lapultsága: $f = \frac{1}{298.257\,223\,563}$

első numerikus excentritás: $e = \sqrt{\frac{a^2-b^2}{a^2}} = 0.08181919$

A görbületi sugár számítása az adott szélességi körön

$$N = \frac{a}{\sqrt{1 - e^2 sin^2\phi}}$$

Koordináták számítása:

$$X = (N + h)\cos\phi\cos\lambda$$

$$Y = (N + h)\cos\phi\cos\lambda$$

$$Z = \left(\frac{b^2}{a^2}N + h\right)\sin\phi$$

2.3.2. Transzformáció ECEF-ből NED-be

A két derékszögű koordináta rendszer közötti átváltás egy forgatási és eltolási transzformációval lehetséges.

A forgatási transzformáció mátrixa

$$R = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\phi\cos\lambda & -\cos\phi\sin\lambda & \sin\phi \end{bmatrix}$$

Ha rendelkezésünkre áll egy pont ECEF koordinátája (X,Y,Z) és a NED koordinátarendszer origójának ECEF koordinátái (N_0, E_0, D_0), akkor a 2.1 egyenlet segítségével kiszámíthatjuk a pont N,E,D koordinátáit.

$$\begin{pmatrix} N \\ E \\ D \end{pmatrix} = R \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - \begin{pmatrix} N_0 \\ E_0 \\ D_0 \end{pmatrix} \tag{2.1}$$

2.3.3. Transzformáció

$$R_{3DZ} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{3DX} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{bmatrix}$$

$$R_{3DY} = \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

$$R_{3D} = R_{3DZ} \cdot R_{3DX} \cdot R_{3DY} \tag{2.2}$$

2.4. Nap pozíciója

A Nap helyzetének meghatárzosása egy adott helyen, egy adott időpontban az alábbi lépések sorozatával lehetséges.

1. A Nap helyzetének számítása az ekliptikai koordináta rendszerben. A csillagászatban az ekliptikai koordináta-rendszer egy égi koordináta-rendszer, ame-

lyet általában a Naprendszer objektumai látszólagos helyzetének, pályájának és pólusirányának ábrázolására használnak.

2. Átváltás ekvatoriális koodináta-rendszerbe.

3. Átváltás horizontális koodináta-rendszerbe. A horizontális koordináta rendszerben a Nap helyzete az adott helyi égbolton jelenik meg, azaz az irányt és a magasságot jelöli meg, ahonnan a Napot megfigyeljük. A számításhoz szükség van a megfigyelő helyének geográfiai koordinátáira, valamint a megfigyelés időpontjának pontos ismeretére.

2.4.1. Ekliptikai koordináták

A 2.4.1 fejezetben az ekliptikai koordináták meghatározására kerül sor. Az ekvatorális koordináták számításához a Nap ekliptikai hosszúságának (l) és az ekliptika hajlásának (ϵ) ismerete szükséges.

2.1.3-as számú fejezetben említésre került, hogy a csillagászatban használt időábrázolási forma az ún. Julián-dátum. Elsőként az univerzális időben (UT) megadott megfigyelési időpont alapján meghatározzuk a Julián-dátumot. 2.3-es számú egyenlettel a Julián-dátum egész része, a Julián-nap kerül meghatározásra. 2.3-es egyenletben kizárólag egész osztásokat végzünk, az egyenletben szereplő y a megfigyelés UT-ben megadott évét, a m a hónapját, a d a napját jelölik. 2.4-as számú egyenletben a Julián-dátum tizedes részének meghatározására kerül sor.

2.5-as számú egyenlet greenwichi dél óta (2000. január 1.) eltelt napok számát határozza meg a töredéknapokat is beleértve. Ezt a számot jelöljük n-nel.

A további egyenletekben használt jelölések:

$\boldsymbol{L}$ a perihélium ekliptikai hosszúsága, (mean longitude of the Sun)

$\boldsymbol{g}$ közép anomália, (mean anomaly of the Sun)

$\boldsymbol{l}$ Nap ekliptikai hosszúsága, (the ecliptic longitude of the Sun)

ϵ az ekliptika hajlása, (obliquity of the ecliptic)

$$\begin{aligned} jdn = &(1461 \cdot (y + 4800 + (m - 14)/12))/4 \\ &+ (367 \cdot (m - 2 - 12 \cdot ((m - 14)/12)))/12 \\ &- (3 \cdot ((y + 4900 + (m - 14)/12)/100))/4 \\ &+ d - 32075 \end{aligned} \tag{2.3}$$

$$jd = jdn + (hour - 12.0)/(24.0) + min/1440.0 + sec/86400.0 \tag{2.4}$$

$$n = jd - 2451545.0 \tag{2.5}$$

$$\Omega = 2.1429 \cdot 0.0010394594 \cdot n \tag{2.6}$$

$$L = 4.895063 + 0.017202791698 \cdot n \tag{2.7}$$

$$g = 6.2400600 + 0.017202791698 \cdot n \tag{2.8}$$

$$\begin{aligned} l = &L + 0.03341607 \sin(g) + 0.00034894 \sin(2g) \\ &- 0.0001134 - 0.0000203 \sin(\Omega) \end{aligned} \tag{2.9}$$

$$\epsilon = 0.4090928 - degToRad(0.0000004) \cdot n + 0.0000396 \cdot \cos(\Omega) \tag{2.10}$$

2.4.2. Ekvatoriális koordináták

Az ekliptikai koordináták átváltása az ekvatoriális koordinátarendszerbe 2.11 és 2.12 egyenletek segítségével történik. Az α a rektaszcenziót (angolul: right ascension), a δ pedig a Nap deklinációját (angolul: declination) jelöli. A deklináció és a rektaszcenzió a II. ekvatoriális koordináta rendszer két koordinátája.

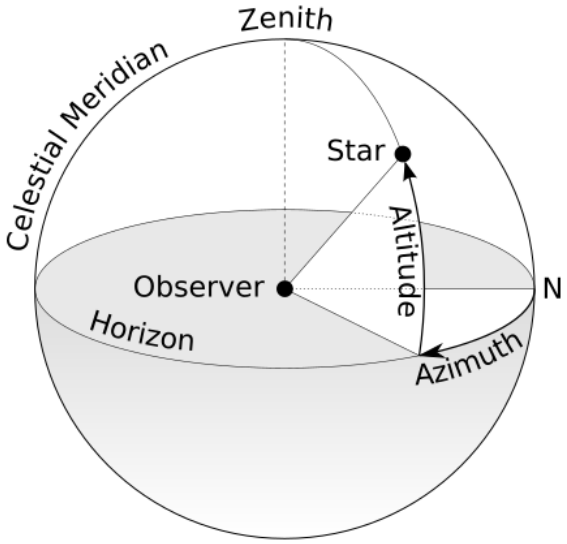
$$\alpha = \arctan\left(\frac{\cos(\epsilon) \cdot \sin(l)}{\cos(l)}\right) \tag{2.11}$$

$$\delta = \arcsin\left(\sin\epsilon \cdot \sin l\right) \tag{2.12}$$

2.4.3. Horizontális koodináták

A horizontális koordináta-rendszer egy olyan csillagászati vonatkoztatási rendszer, amely szerint az adott megfigyelési pontból az égbolton látható objektumokat egy gömb felületen látjuk. Az égitestek látszólagos pozícióját ezen a gömbfelületen helyezzük el. A pozició megadásához két szög koordináta megadására van szükségünk. Az azimutra(γ) és a napmagasságra(α_s). A napmagasság helyett szokás a zenitszöget(Θ_z) megadni. $\Theta_z + \alpha_s = \frac{\pi}{2}$

A napmagasság a megfigyelő és az objektumot összekötő egyenes és a horizont által bezárt szög. Az azimut az északi irány és megfigyelőt az objektummal összekötő egyenes horizontra vetett képe közötti szög, balsodrású koordináta rendszerben.



2.2. ábra. Azimut és napmagasság

A további egyenletekben használt jelölések:

gmst greenwichi közép sziderikus idő, (Greenwich mean sidereal time)

lmst helyi közép sziderikus idő, (local mean sidereal time)

ω az óraszög (Hour angle)

$$gmst = 6.6974243242 + 0.0657098283 \cdot n + hour \tag{2.13}$$

$$lmst = (gmst \cdot 15 + \lambda) \cdot (\pi/180) \tag{2.14}$$

$$\omega = lmst - \alpha \tag{2.15}$$

$$\Theta_z = \arccos\left(\cos(\phi)\cos(\omega)\cos(\delta) + \sin(\delta) * \sin(\phi)\right) \tag{2.16}$$

$$\gamma = \arctan\left(\frac{-\sin(\omega)}{\tan(\delta)\cos(\phi) - \sin(\phi) * \cos(\omega)}\right) \tag{2.17}$$

2.5. Grafikus csővezeték

A grafikus API-k a grafikus adatok képernyőn való megjelenítésének feladatát részfeladatokra bontják. Az adatok egymástól függetlenül haladnak át a grafikus csővezetéken (graphics pipeline), ahol minden egyes állomásnak meg van a maga feladata, a maga felelősségi köre. Minden állomás bemenete a megelőző feldolgozó egység kimenete. Vagyis az s_i kimenete az $s_i + 1$ bemenete. A grafikus csővezeték végzi el a színtér megjelenítését a megadott bemeneti adatok alapján. A csővezeték bemeneti adatai magukba foglalják a megjelenítendő tárgyak geometriai és optikai modelljeit, virtuális kamera adatokat, a képkeretet, a színtér fényforrásait és a megvilágítási adatokat. A grafikus API-k lehetővé teszi a fejlesztők számára az

egyes állomások finomhangolását és testre szabását. A DirectX több feldolgozó egységgel rendelkezik, amelyek közül az alábbi öt kerül részletes bemutatásra: Input Assembler, Vertex Shader, Rasterizer, Pixel Shader és Output Merger.

2.5.1. Input Assembler(IA)

Az Input Assembler (IA) felelős az adatok összegyűjtéséért és feldolgozásáért, majd továbbítja a következő állomásnak. Kiolvassa a csúcspontokat a memóriából, és létrehozza belőlük a geometriát. Valamennyi geometriai elemet egyedi azonosítóval lát el, majd továbbítja a csővezetéken. A feladatok elvégzése nagyrészt automatikusan történik a fejlesztőnek az alábbi lehetőségei vannak az Input Assembler működésének beállítására.

Az adatok kiolvasásához a fejlesztőnek meg kell határoznia a vertex és index puffer memóriacímét.(IASetVertexBuffers, IASetIndexBuffers). Ahhoz, hogy a memóriában található adatokat a GPU helyesen értelmezze meg kell adnunk úgynevezett Input Layout-ot (IASetInputLayout). Itt definiálhatjuk az adatok sorrendjét és típusát. Pl.: Az egy vertexhez tartozó adatok : pozíció vektor, normál vektor, szín vektor. A geometriai primitívek topológiájának meghatározásával megadható, hogy a pontok sorozatából milyen geometriát építsen fel az IA, például háromszögek sorozat't, vonalláncot vagy vonalak sorozatát (IASetPrimitiveTopology).

2.5.2. Vertex Shader (VS)

A Vertex Shader olyan program, amely minden egyes csúcspontot feldolgoz és kiszámítja annak helyzetét a 3D térben.

A Vertex Shader a bemeneti regiszterekbe várja a vertex adatokat. Ezekből a regiszterekből olvassa ki például a vertexhez tartozó pozíció vektort, normál vektort és/vagy a szín vektort.. Egyéb paramétereket, amelyek nem a vertex tulajdonságait írják le, a konstans és ideiglenes regiszterekből olvas ki, például a megvilágítás paramétereit.

A vertex shader eredményeit a kimeneti regiszterekbe helyezi. A kötelezően feltöltendő regiszter az oPos regiszter, amelybe a csúcspont homogén képernyőkoordinátái kerülnek. Ez a pozíció vektoron végrehajtott világ, nézet és vetítési transzformációk egymás utáni sorozatával kapjuk meg.

2.5.3. Rasterizer

A Rasterizer a vágóvonalakat számolja ki, amelyek meghatározzák, hogy melyik elem melyik része jelenik meg a képernyőn, véglegesíti a takarási viszonyokat. Többek között ebben a szakaszban kerülnek kitöltésre a vertexek közötti felületek képpontokkal, interpolációs technikával.

2.5.4. Pixel Shader (PS)

A Pixel Shader a Rasterizer után kerül végrehajtásra. Feladata, hogy minden egyes pixel számára kiszámolja annak színét, és itt végezhetőek el a textúrázási, pixelenkénti megvilágítási és egyéb utófeldolgozási műveletek, például az elmosódás beállítása. A Pixel Shader a szín- és textúra-regiszterekből kapja meg az interpolált szín- és textúraadatokat, amelyeket a Rasterizer generál. Az ideiglenes és konstans regisztereket további paraméterek átadására használhatjuk. A Pixel Shader eredménye a pixel színértéke, amely a színpufferbe kerül. A színpuffer a képernyőn ténylegesen megjelenített kép előállításának utolsó fázisa.

2.5.5. Output Merger(OM)

Az Output Merger a grafikus csővezeték utolsó része. Az OM összegyűjti az összes Pixel Shader által előállított szín értékeket, majd azokat összeilleszti a végleges képpé. Az OM-nak van lehetősége a végleges kép felülvizsgálatára és manipulálására is, például a színmaszkok és átlátszóságok beállításával.

2.6. Árnyalás

2.7. Fogalmak

2.7.1. szemantikus szegmentálás

2.7.2. kamera trajektória

2.7.3. triangulált térháló

3. fejezet

Felhasználói dokumentáció

3.1. A feladat ismertetése

A program feladata szimuláció készítése egy szemantikusan szegmentál térhálóról. A felhasználó grafikus felületen választhatja ki a szimulációhoz szükséges állományokat. A szimuláció közben mind a billentyűzet, mind a grafikus user interface elemeinek segítségével módosíthatja a szimuláció paramétereit. A fejlesztés eredménye egy C++ programozási nyelven írt, Windows specifikus, DirectX 11 grafikus API alapú felülethálló megjelenítő lett.

3.2. Minimum rendszerkövetelmények

A program rendszerkövetelményei a Windows SDK rendszerkövetelményei alapján kerültek meghatározásra.

Processzor legalább 1.6 Ghz -es, x86 architektúrájú processzor

Memória 1 GB RAM

Videókártya DirectX 11-et támogató videókártya

Tárhely legalább 100MB szabad tárhely

Operációs rendszer Windows 10 (x86).

3.3. Szoftveres követelmények

A program DirectX11 grafikus API-t használ, emiatt a szükség van a Windows SDK telepítésére. A Windows 8 operációs rendszertől kezdve a DirectX SDK a Windows SDK része. Korábban a DirectX SDK a Windowson történő játékfejlesztés platformjaként működött. Azonban mára, hogy a számítógépek széles körben rendelkeznek Direct3D támogatással, így az egyszerűbb asztali alkalmazások is kihasználhatják a grafikus hardveres gyorsítást. A Microsoft integrálta a DirectX technológiákat az operációs rendszerbe. A program futtatásához nem szükséges telepíteni a korábbi(legacy) DirectX SDK-t. A program kizárólag a Windows SDK által biztosított függvényeket használja.

3.4. macOS

A program implementálása macOS operációs rendszeren történt. A fejlesztés során a Parallels Desktop for Mac szoftver biztosította a szükséges hardvervirtualizációs környezetet a Windows specifikus funkciókhoz. A program futtatásához szükséges minimális rendszerkövetelmények macOS operációs rendszer esetén.

Operációs rendszer legalább macOS 10.14.4 vagy legalább macOS 10.15

Virtuális környezet Parallels Desktop 15

Virtuális operációs rendszer Windows 10

4. fejezet

Fejlesztői dokumentáció

4.1. Funkcionális követelmények

4.1.1. Bemeneti adatokra vonatkozó követelmények

Felhasználói interakció

A program indítását követően a felhasználónak lehetősége van a bemeneti adatokat meghatározni. A bemeneti adatokat a meghatározott formátumú és kiterjesztésű fájlok elérési útvonalának megadásával érheti el.

Bemeneti adatok

- 3D szemantikus térkép (.stl)
- trajektória fájl (.csv)
- szimuláció időpontja (beviteli mező, alapérték a .csv-ben található GPS idő [sec, nsec])
- kamera paraméter file (belső esetleg külső paraméterek és radiális torzítás paraméterek)

Felületháló adatai

A felületháló pontjai méterben adottak egy adott GPS koordinátán (origo) számolt WGS84 flat-Earth approximációból származó jobbsodrású Descartes koordináta rendszerben. A fájl tartalmazza a felületháló pontjainak az origóhoz viszonyított koordinátáit, a pontok közötti felületek normálvektorát és a szegmentációs osztályt.

A program képes a ASCII és a bináris kódolású .stl kiterjesztésű fájl beolvasására is.

Kamera trajektória

A program előre meghatározott formátumú kamera trajektóriákat képes betölteni. Az útvonal meghatározása kamera állások sorozatával történik. Egy adott sorozatelem adatai : Kamera pozíció (North-East-Down, továbbiakban NED), kamera tájolás (Euler szögek yaw-pith-roll), GPS időpont (sec,nsec). A drónok mozgásának leírására használt NED egy jobbsodrású koordináta rendszer.

Bemeneti fájlok ellenőrzése

A program a bemeneteket ellenőrzi és figyelmezteti a felhasználót, ha a bemeneti fájlok szintaktikai hibákat tartalmaznak. A program a bemenetek megfelelő betöltése érdekében a bevitel megismétlését kéri, ha hibát észlel.

4.1.2. Szimulációra, megjelenítésre vonatkozó követelmények

Térháló beolvasás

A betöltött adatokat a megfelelő formátumra átalakítva felépíti a felülethálót, elhelyezi a saját koordináta rendszerében. Térhálónak elemeihez tartozó adatok: pozíció, normál vektor, szegmentációs osztály. A felhasználó a grafikus felületen módosíthatja a modell elemek színét, pozícióját, tájolását, láthatóságát. A felhasználó törölheti a felülehálót a beolvasott modellek listájából.

Fényforrás beállítása

Nap helyzetének (direktcionális fény irányának) számítása GPS idő és kamera paraméterek között megadott GPS koordináta alapján történik. A szimuláció során azzal a feltételezéssel élünk, hogy a térháló max 1-2 km átmérőjű. Emaitt a modell térbeli kiterjedése nem haladja meg azt a léptéket, hogy az aktuális modellben a térbeli helyzet változása módosítaná a Nap állását. Statikusnak vesszük a fény irányát a tér függvényében.

Kamera és nézetkezelés biztosítása

A projekciós mátrixhoz szükséges kamera paraméterek beviteli mezőkön módosíthatóak:

- látómező, *Field of view*, FOV
- képarány, *aspect ratio*
- közeli vágósík, *near screen*
- távoli vágósík, *far screen*

Kétféle nézetkezelési módszer biztosított:

- kamera trajektória lejátszása, *Körséta mód* vagy angolul *Flythrough mode*
- szabad barangolás, *Barangolás mód* vagy angolul *Explore 3D mode*

Kamera trajektória lejátszása

A felhasználó a program nézeti ablakában lejátszhatja a kamera trajektória által meghatározott fix útvonalat, miközben a program futási időben rendereli a felületháló aktuális képét.
Lehetőség a szimuláció kezdeti időpontjának módosítására. A felhasználó egy beviteli mezőben megadhatja trajektória kezdeti időpontját Unix-idő formátumban. Lehetőség a modell eltolására és elforgatására, így a felhasználó megadhatja a trajektróia és a modell objektumok elhelyezkedésének egymáshoz való viszonyát.
A kamera mozgatásának sebességét alapértelmezetten a trajektória mintavétele határozza meg. Mértéke kb 50Hz. A felhasználó módosíthatja a trajektória lejátszásának sebességét. A trajektória lejátszása közben a felhasználó egy idővonal segítségével beállíthatja az aktuális képkockát. A kamera útvonalat a program egy térbeli vonalláncként jeleníti meg a felületháló felett.

Szabad barangolás

A felhasználó billentyűzet és egér segítségével szabadon bejárhatja a térháló környezetét. A kamera mozgatásának sebességét a felhasználó grafikus felületen módosíthatja. A Nap állása módosítható egy bemeneti mezőben beállítható időpont megadásával. A szabad barangolás szimuláció közben a fény iránya statikus marad, az idő függvényében nem változik.

Árnyalás és fényelés

DirectX 11 SDK által biztosított programozható modell biztosítja. Vertex és Pixel shaderek megírásra kerültek.

4.1.3. Kimeneti adatokra vonatkozó követelmények

Képek mentése

A szimuláció során az exportálás parancs kiadásával a nézeti ablakban lejátszott képek kimenthetőek .png formátumban.

4.2. Nem funkcionális követelmények

Hatékonyság

- A programnak a betöltött felületháló poligonszámával arányos processzor, GPU és memória terhelést kell generálnia. A memória és merevlemez terhelés a felületháló poligon számával arányos, de nem haladhatja meg a 500 MB-ot.
- A programnak a legtöbb funkció esetén minden bevitelre gyors (1 másodperc alatti) válaszidőt kell biztosítania.
- A felületháló modell betöltése a háttértárról a memóriába több időt vehet igénybe, a maximum elfogadható várakozási idő 5 perc.

Megbízhatóság

- A szabványos használat mellett a programnak maximum 1.5 millió poligont kell kezelnie hibamentesen, és nem szabad hibaüzenetet vagy hibajelenséget előidéznie.
- Ha a felhasználó hibás bevitelt ad meg, a programnak hibaüzenetet kell kiadnia, majd lehetőséget kell biztosítania a bevitel megismétlésére.

Biztonság

- A programnak nincsenek biztonsági követelményei.

Hordozhatóság

- A program futtatása legalább Windows 10-es operációsrendszert igényel

- A programhoz szükséges összes komponens megfelelő használatához telepített Windows SDK-ra van szükség
- A program nem igényel külön telepítést

Felhasználhatóság

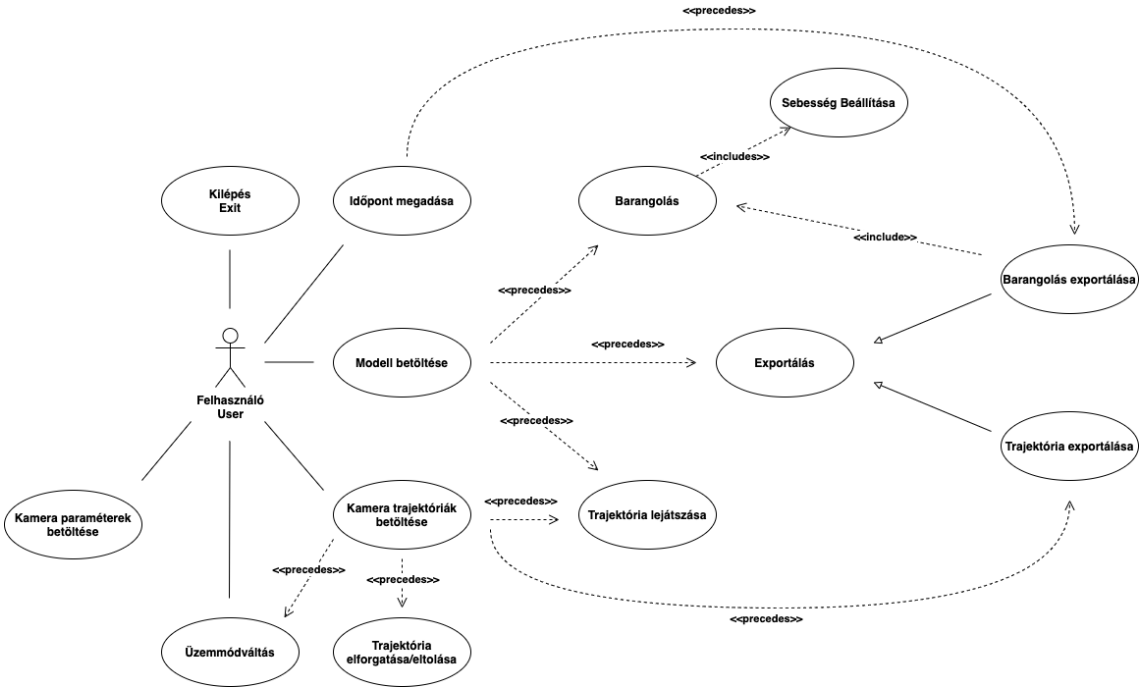
- Intuitív felhasználói felületet kell biztosítani, amely egyszerűen használható és könnyen érthető a felhasználók számára. Az instrukcióknak világosnak és pontosnak kell lenniük, hogy segítsék a felhasználókat a program megértésében és használatában.
- A felhasználói leírásban külön segédletet kell készíteni a használatról. Ez lehetőséget ad a felhasználóknak, hogy részletesebben megismerjék a programot és hogyan használják azt.

Fejlesztési

- A program nyelve a C++.
- A program fejlesztéséhez használt fejlesztői környezet: Visual Studio 2022 IDE.
- A program tervezésére és fejlesztésére használt paradigma objektumorientált. Az OEP könnyen karbantartható, újrafelhasználható és skálázható kódot eredményez.

4.3. Felhasználói eset diagram

A 4.1. ábrán a felhasználói eset diagram mutatja be milyen utasításokat adhat ki a felhasználó.



4.1. ábra. Felhasználói eset diagram

4.4. Felhasználói történetek

4.4.1. Külső állományok betöltése

AS A		Felhasználó
I WANT TO		betölteni a külső állományokat
SO THAT		inicializáljam a szimuláció paramétereit
1	GIVEN	program sikeresen inicializálja a DirectX eszközöket és elindul a *Barangolás* mód
	WHEN	megadjuk a triangulált terepmodell elérési útvonalát és sikeresen betöltődik
	THEN	a program jelzi a sikeres beolvasást
2	GIVEN	program sikeresen inicializálja a DirectX eszközöket és elindul a *Barangolás* mód
	WHEN	megadjuk a kamera trajektória elérési útvonalát és sikeresen betöltődik
	THEN	a program jelzi a sikeres beolvasást és elérhetővé válik a 2 mód közötti váltás
3	GIVEN	program sikeresen inicializálja a DirectX eszközöket és elindul a *Barangolás* mód
	WHEN	megadjuk a kamera paramétereket tartalmazó fájl elérési útvonalát és sikeresen betöltődik
	THEN	a program jelzi a sikeres beolvasást
4	GIVEN	program sikeresen inicializálja a DirectX eszközöket és elindul a *Barangolás* mód
	WHEN	bármelyik külső fájl betöltése sikertelen
	THEN	program figyelmeztet

4.1. táblázat. Felhasználói eset, külső állományok betöltése

4.4.2. Barangolás

AS A		Felhasználó
I WANT TO		*Barangolás* módba lépni
SO THAT		
1	GIVEN	a program alapértelmezett indítása *Barangolás* módban történik
	WHEN	
	THEN	
2	GIVEN	*Körséta* mód van beállítva
	WHEN	*3D Explore* gombra kattintva
	THEN	program elindítja a *Barangolás* módot.

4.2. táblázat. Felhasználói eset, *Barangolás* mód indítása

AS A		Felhasználó
I WANT TO		szabadon barangolni a terepmodell felett
SO THAT		megtekintsem a modellt
1	GIVEN	*Barangolás* mód van beállítva
	WHEN	a WSAD billentyűk lenyomásával
	THEN	a program módosítja a kamera pozícióját a megfelelő irány alapján, és megjeleníti az újra renderelt képet
2	GIVEN	*Barangolás* mód van beállítva
	WHEN	a bal egérgomb lenyomásával és az egér mozgatásával
	THEN	a program módosítja a kamera tájolását, és megjeleníti az újra renderelt képet
3	GIVEN	*Körséta* mód van beállítva
	WHEN	*3D Explore* gombra kattintva
	THEN	a program elindítja a *Barangolás* módot

4.3. táblázat. Felhasználói eset, *Barangolás*

4.4.3. Körséta, trajektória lejátszása

AS A		Felhasználó
I WANT TO		*Körséta* módba lépni
SO THAT		megtekintsem a modellt az kamera útvonal mentén
1	GIVEN	nincs betöltött kamera trajektória
	WHEN	*Flythrough* gombra kattintva
	THEN	a porgram figyelmeztet, hogy nincs még betöltött trajektória
2	GIVEN	van betöltött trajektória és a *Barangolás* mód van beállítva
	WHEN	*Flythrough* gombra kattintva
	THEN	a program elidítja *Körséta* módot és a kezdő pozícióba helyezi a kamerát

4.4. táblázat. Felhasználói eset, *Körséta* mód indítása

AS A		Felhasználó
I WANT TO		lejátszani a kamera trajektóriát
SO THAT		megtekintsem a modellt az kamera útvonal mentén
1	GIVEN	van betöltött trajektória és a *Körséta* mód van beállítva
	WHEN	*Play* gombra kattintva
	THEN	a program elindítja a szimulációt a modell felett a megadott kameraútvonal mentén.
2	GIVEN	van betöltött trajektória és a *Körséta* mód van beállítva
	WHEN	*SPACE* billentyűt leütve
	THEN	a program elindítja a szimulációt a modell felett a megadott kameraútvonal mentén.

4.5. táblázat. Felhasználói eset, kamera trajektória lejátszása

AS A		Felhasználó
I WANT TO		megállítani a kamera trajektória lejátszását
SO THAT		
1	GIVEN	a program a trajektróia útvonala mentén halad
	WHEN	*Play* gombra kattintva
	THEN	a program megállítja a szimulációt az adott kamerapozícióban
2	GIVEN	a program a trajektróia útvonala mentén halad
	WHEN	*PAUSE* gombra kattintva
	THEN	a program megállítja a szimulációt az adott kamerapozícióban

4.6. táblázat. Felhasználói eset, kamera trajektória lejátszásának megállítása

4.5. Architektúra

A program fejlesztéséhez használt architektúra a Model-View-Controller (MVC).

4.5.1. Model-View-Controller

Az MVC architektúra három fő komponenst tartalmaz.

- A Model felelős az adatkezelésért és feldolgozási logika definiálásáért. A Model független a View-tól és a Controller-től.
- A View az adatok megjelenítéséért felelős. Megjeleníti a Model aktuális állapotát a felhasználó számára és létrehozza a felhasználói felület. A View csak a megjelenítési szabályokat írja le, egyéb számításokat nem végez.
- A Controller fogadja a felhasználói bemeneteket, és továbbítja azokat a Modelnek és a View-nak.

Model - View

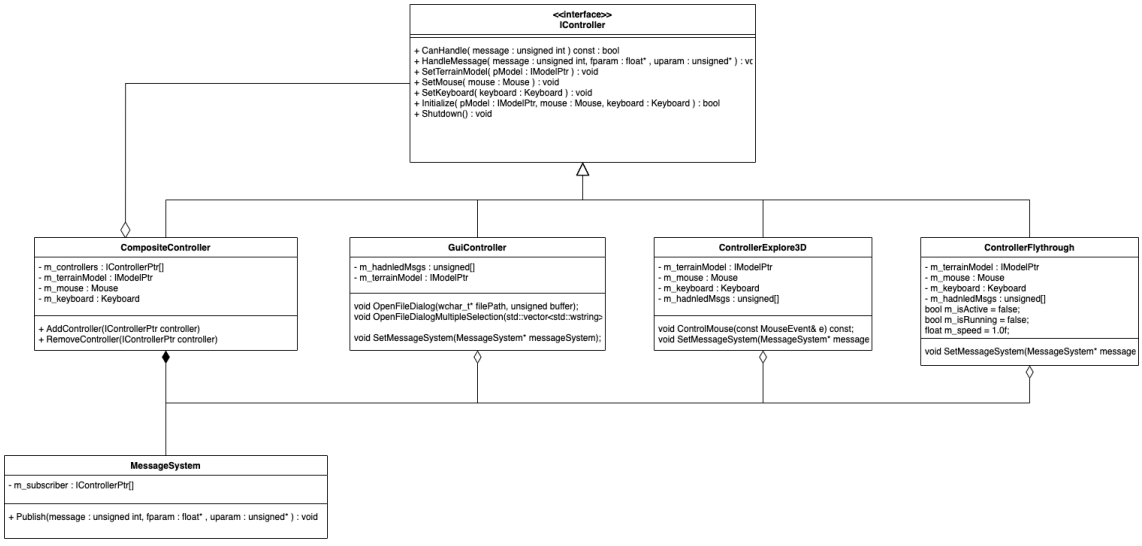
A Model és a View közötti kapcsolatot az Megfigyelő (Observer) tervezési minta biztosítja. Az Alany (Subject) objektum lehetővé teszi, hogy több különböző Megfigyelők (Observer) feliratkozhasson rá. A feliratkozók értesítést kapnak, amennyiben az Alany állapota megváltozik. A Model-View kapcsolatában a Model a Subject, a View Observerként jelenik meg. Az Observer minta lehetővé teszi, hogy a Model állapotának bármilyen változása azonnal értesítse az összes feliratkozott View-t, amelyek így azonnal frissíthetik a megjelenített adatokat

Controller - View

A Controller-ek között felépített hierarchia segítségével elérhetjük, hogy a program a felhasználói bemenetekre különböző módon válaszoljon. Az MVC a View és Controller kapcsolatának kialakítására a Strategy (Stratégia) tervezési mintát javasolja. A Környezet (Context) egy Stratégia(Strategy) objektumra való hivatkozáson keresztül éri el a Stratégia objektumhoz rendelt algoritmust. A Stratégia mintában a Környezet nem ismeri a Stratégia algoritmusát. Tehát, ha módosítjuk a Context által tartalmazott referenciát, módosítjuk az algoritmust. A Contreller és View kapcsolatában a Controller a Strategy, a View Context szerepét tölti be. A View által tartalmazott referencián keresztül lehetőség van, hogy futás közben dinamikus módosítjuk a végrehajtandó algoritmust.

4.5.2. Controller

A 4.2. ábrán látható a Controllerek hierarchikus szerkezete.



4.2. ábra. Controller UML osztálydiagram

Composite pattern

A Controllerek rendszere egy Kompozit (Composite) tervezési mintát követ. Az alkalmazott minta a Component interface, a Leaf és a Composite osztályokat használja. A Komponens (Component) interface előírja a tagfüggvényeket, amelyeket minden leszármazottnak implementálnia szükséges. A Levél (Leaf) objektumok a Komponens interface leszármazottjai. A Levelek többnyire nem tartalmaznak továb-

bi alelemet. A megkapott feladatot nem delegálják tovább. A Kompozit (Composite) objektum szintén a Komponens interfészt követik, és további Komponenseket tartalmaz, amelyek lehetnek Levelek, de akár további Kompozit objektumok is. A Kompozit objektumok alegységeikre delegálják a feladatokat.

A CompositeController referenciával hivatkozik a további Controllerekre. A további Controller-ek a Leaf szerepet töltik be.

Kommunikáció a Controllerek között

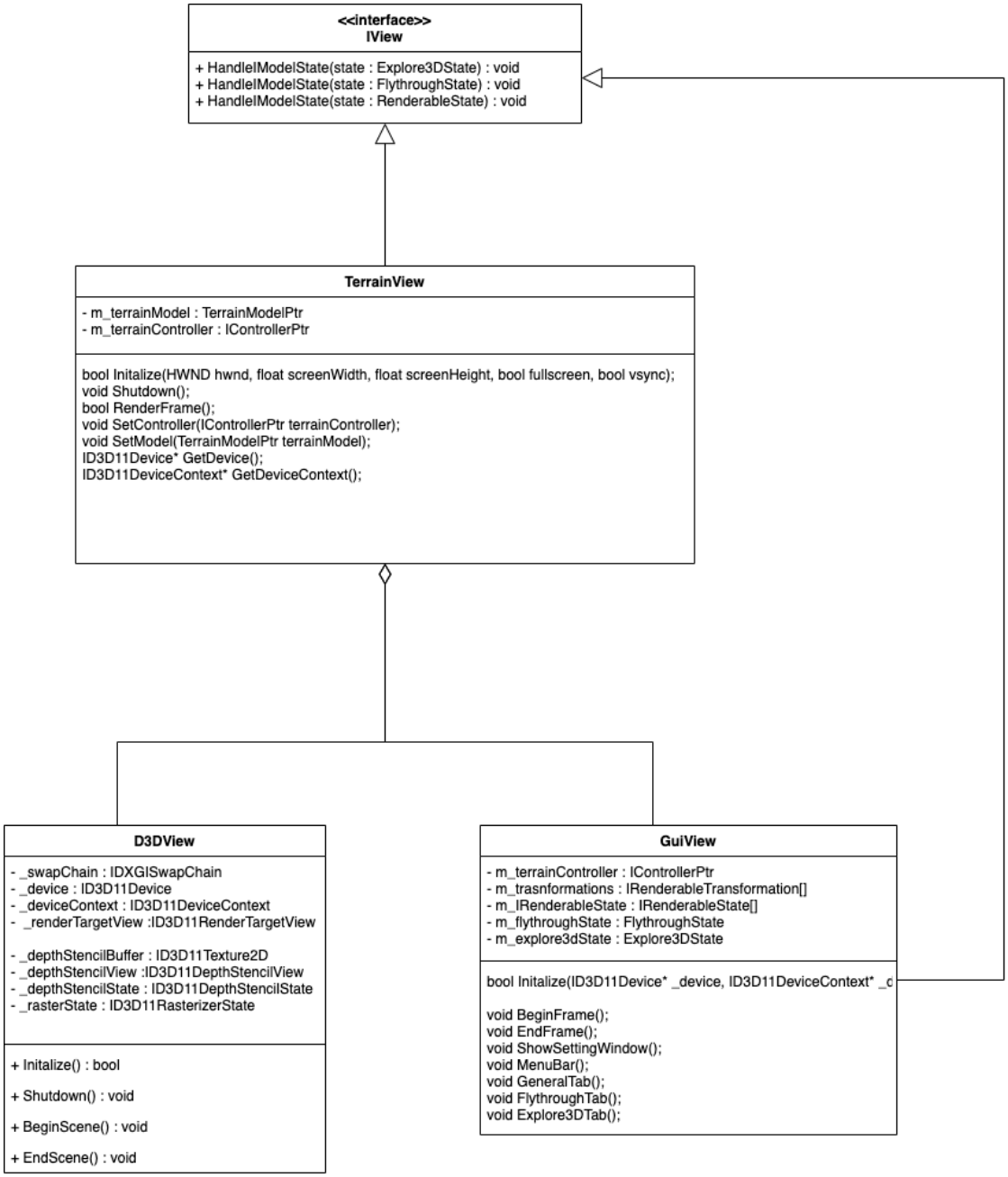
A Controllerek egymás között az Observer minta segítségével kommunikálnak, melyet a MessageSystem objektum tesz lehetővé. Minden Controller referenciával rendelkezik a MessageSystem objektumra, és a Publish metódus segítségével tudnak üzenetet küldeni egy másik Controllernek.

Az IController interfész előírja a HandleMessage függvényt, amely segítségével a leszármazott Controller objektumok meg tudják kapni az üzeneteket. Az üzenetek típusát előre meghatározott makrók definiálják, és az üzenetek különböző paraméterekkel, például fparam és uparam értékekkel egészíthetőek ki. Az üzenetkezelés megoldása a Microsoft WndProc függvényére hasonlít. A HandleMessage függvény:

```
1 void HandleMessage( unsigned message,
2                     float* fparam,
3                     unsigned* uparam );
```

4.1. forráskód. HandleMessage tagfüggvény

4.5.3. View

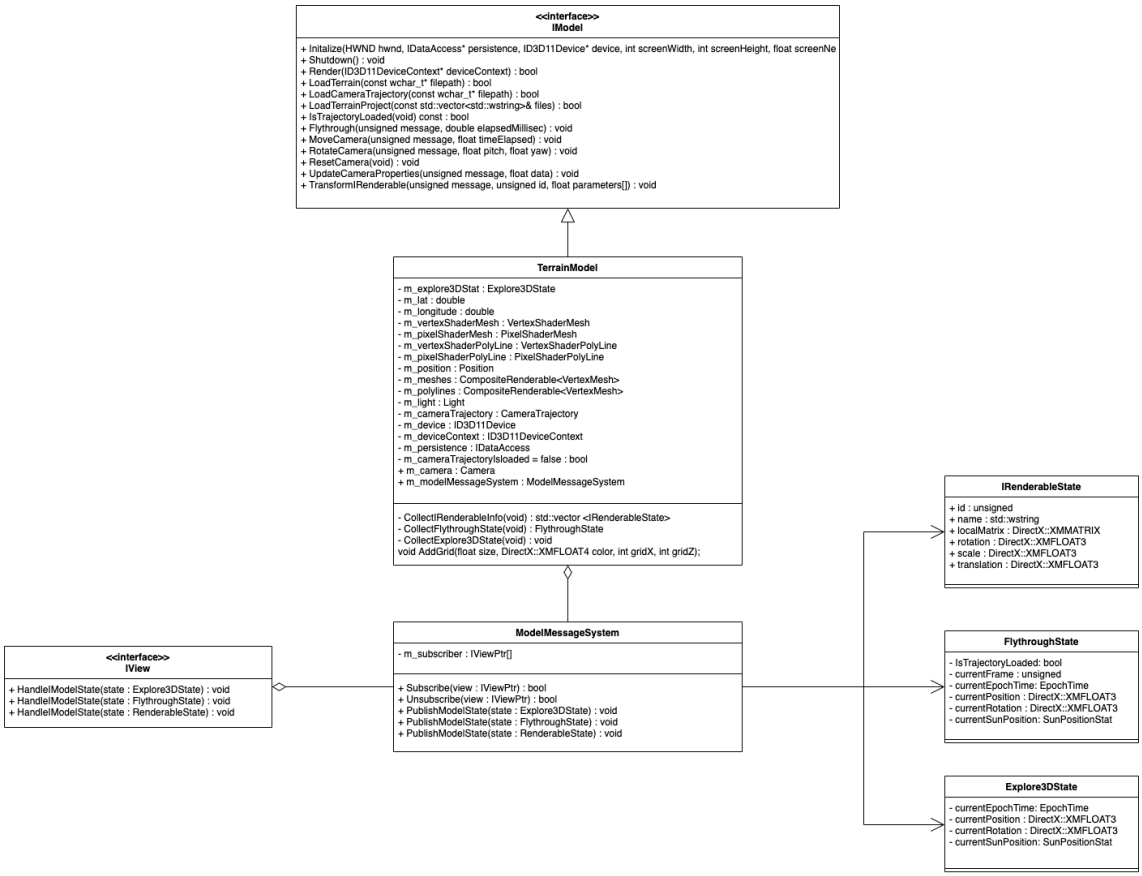


4.3. ábra. View UML osztálydiagram

4.5.4. Model - View

A Model és a View közötti kommunikáció az Observer patternnek megfelelően valósul meg, amint az 4.4. ábrán látható. Az IView interfész előírja a HandleModelState függvényt, amely képes különböző összetett adatszerkezetek fogadására. Ezekben a struktúrákban tárolja a View a Model aktuális állapotát, majd megjeleníti azt a képernyőn.

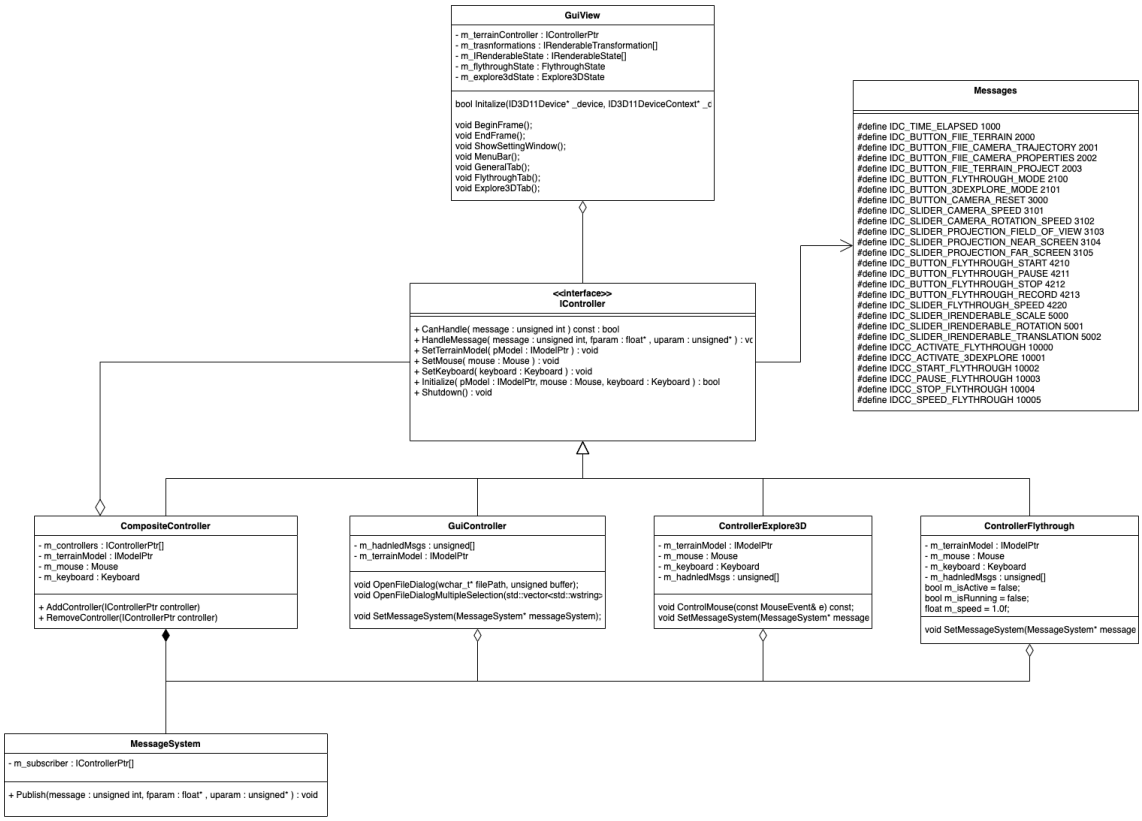
Az Observer pattern-t a ModelMessageSystem objektum biztosítja. Az objektumra IView típusú megfigyelők iratkozhatnak fel. Ha az IModel objektum állapota változik, a ModelMessageSystem objektum a feliratkozókat értesíti a PublishModelState metóduson keresztül.



4.4. ábra. IModel - IView kapcsolat, UML osztálydiagram

4.5.5. Controller - View

A 4.5. ábrán látható a Stratégia tervezési minta megvalósítása a grafikus interfész(GUI) és a Controller között. A komponensek egy üzenet alapú kommunikációt valósítanak meg a felhasználói bemenet kezelésére, a View egy előre meghatározott üzenet típust és paramétereket továbbít az IController HandleMessage metódusának.



4.5. ábra. IController - IView kapcsolat, UML osztálydiagram

Ez a megközelítés lehetővé teszi, hogy a GUI elemek könnyen kommunikáljanak az IController objektummal, anélkül, hogy ismeretük lenne egymás belső működéséről. Az azonosítók használata lehetővé teszi a GUI elemek azonosítását, így továbbítva a megfelelő üzenetet az IController felé. A további paraméterek lehetővé teszik a felhasználói bemenet részletes leírását, amely alapján az IController objektum kiválaszthatja a megfelelő kezelő függvényt és végrehajthatja a megfelelő műveletet.

5. fejezet

Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

Köszönyetnyilvánítás

Amennyiben a szakdolgozati / diplomamunka projekted pénzügyi támogatást kapott egy projektből vagy az egyetemtől, jellemzően kötelező feltüntetni a dolgozatban is. A dolgozat elkészítéséhez segítséget nyújtó oktatók, hallgatótársak, kollégák felé is nyilvánítható külön köszönet.

A. függelék

Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus,

sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

Ábrák jegyzéke

Táblázatok jegyzéke

Algoritmusjegyzék

Forráskódjegyzék