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

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. Régebben párhuzamvonalzóval és papírra rajzolt tustervek készültek. Mára pedig eljutottunk BIM modellek készítéséhez, ahol mind a építészeti tervek, mind a szakági tervek, mind az anyagkimutatás, de a tervek ellenőrzése is a BIM modell segítségével történik. 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. Az összehasonlítást végző szoftver nem a szakdolgozat kereteiben készül. A összehasonlítás eredményeként meghatározható, hogy a drón a tervek szerint halade, vagy eltér a megírt repülési tervtől.

2. fejezet

Felhasználói dokumentáció

2.1. A feladat ismertetése

A program feladata szimuláció készítése egy szemantikusan szegmentál térhálóról. A felhasználó grafikus felületen keresztül 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 felületháló megjelenítő lett. A felületháló környezetét a Nap pozíciójának megfelelő skybox szimulálja, amelyeket a program a megadott paraméterek szerint számolja.

2.2. 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).

2.3. 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.

2.4. 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

3. fejezet

Fejlesztői dokumentáció

3.1. Funkcionális követelmények

3.1.1. 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 Descartes koordináta rendszerben. A felületháló pontjainak GPS koordinátáit, a pontok közötti felületek anyagtulajdonságait tartalmazza egy meghatározott formátumú, .stl kiterjesztésű fájl.

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), kamera tájolás (Euler szögek yaw-pith-roll), GPS időpont (sec,nsec). Fontos megjegyezni, hogy 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 vagy szemantikai hibákat tartalmaznak. A program a bemenetek megfelelő betöltése érdekében a bevitel megismétlését kéri, ha hibát észlel.

3.1.2. 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, anyagtulajdonság)

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

Kétféle módszer közül van lehetőség választani:

- 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 időpontjának módosítására, a fájlban meghatározott időpontok rugalmasan beállíthatóak. Lehetőség van a kamera trajektóriának eltolására és elforgatására a modell felett. A kamera mozgatásának sebessége a trajektória mintavétele határozza meg. Mértéke kb 50Hz. A kamera útvonalat a program egy térbeli vonalláncként jeleníti meg a felületháló felett.

Szabad barangolás

A felhasználó billentyűzet és egér segítségével szabadon bejárhatja. A felvétel utasítás kiadásával a szabad barangolás közben bejárt útvonalról készített képek is kimenthetőek. A Nap állása a felvétel előtt egy bemeneti mezőben beállítható időpont megadással, azonban a szimuláció közben a fény iránya statikus marad, az idő függvényében nem változik.

Felhasználói interakció biztosítása

A szimuláció közben a megfelelő billentyűkombinációval, megfelelő gombra való kattintva legyen jelezhet a programnak. A felhasználó szabad kameraállások sorozatával megtekintheti a modellt. A szabad bebarangolás a billentyűzet nyilaival, az egér mozgatásával lehetséges.

Árnyalás és fényelés

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

3.1.3. Kimeneti adatokra vonatkozó követelmények

Képek mentése

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

3.2. 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 100 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 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.

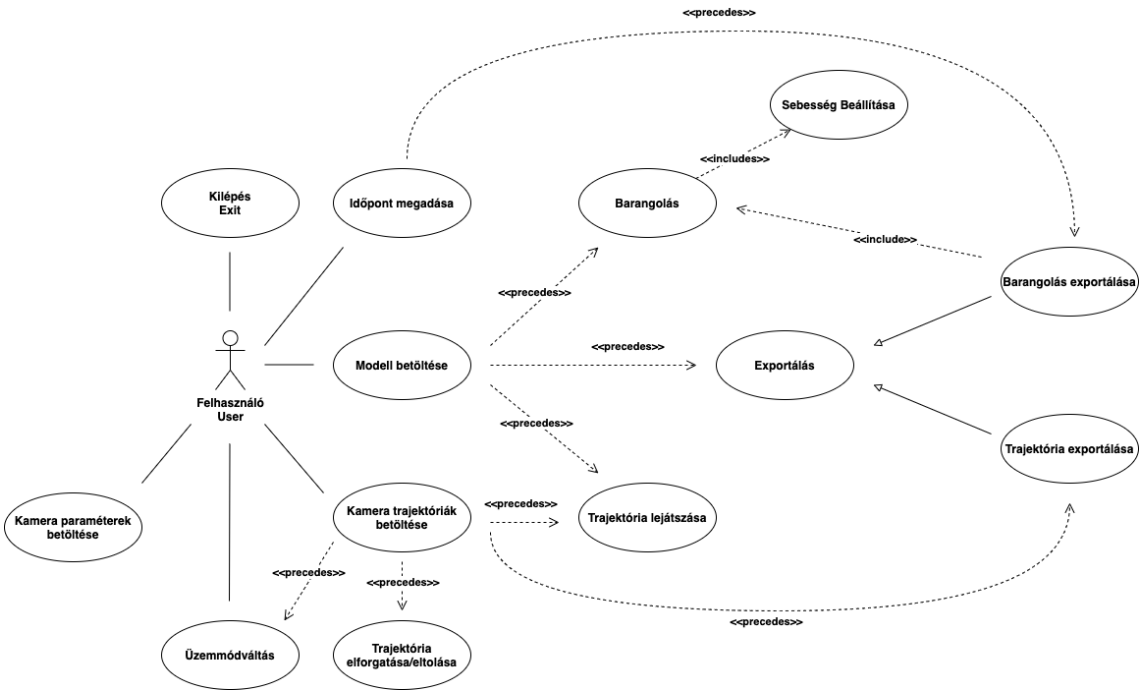
- A felhasználói leírásban külön segédletet kell készíteni a használatról. Ez lehetőséget ad a felhasználóknak, hogy részletesebben megismerjék a programot és hogyan használják azt.

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.

3.3. Felhasználói eset diagram

A 3.1. ábrán a felhasználói eset diagram mutatja be milyen utasításokat adhat ki a felhasználó.



3.1. ábra. Felhasználói eset diagram

3.4. Felhasználói történetek

3.4.1. 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

3.1. táblázat. Felhasználói eset, külső állományok betöltése

3.4.2. 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.

3.2. 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

3.3. táblázat. Felhasználói eset, *Barangolás*

3.4.3. 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

3.4. 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.

3.5. 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

3.6. táblázat. Felhasználói eset, kamera trajektória lejátszásának megállítása

3.5. Architektúra

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

3.5.1. 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

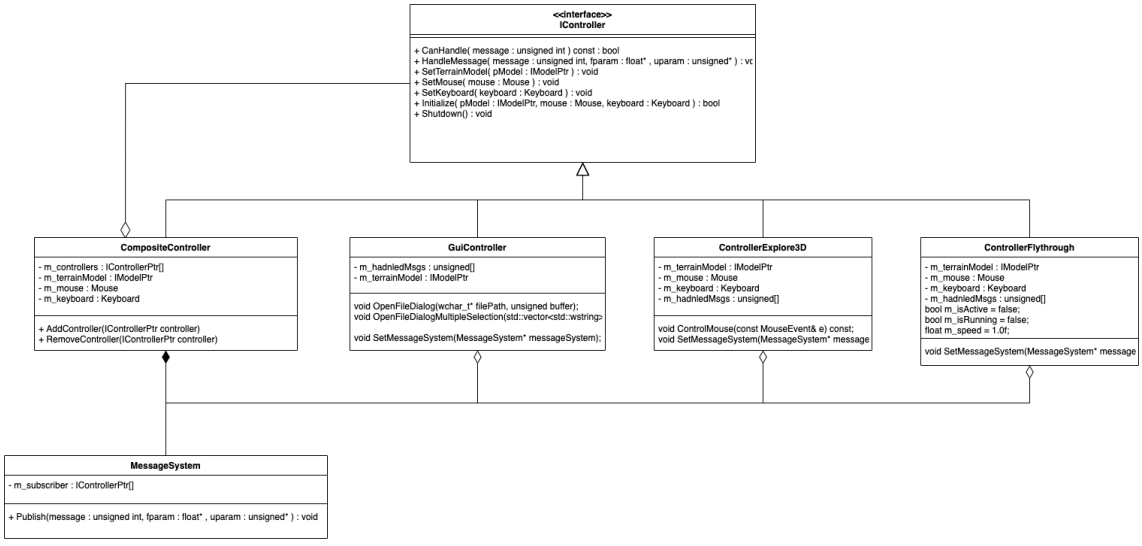
A Model és a View közötti kapcsolatot az Megfigyelő (Observer) tervezési minta biztosítja. Az Alany (Subject) objektum lehetővé teszi, hogy több különböző Megfigyelők (Observer) feliratkozhasson rá. A feliratkozók értesítést kapnak, amennyiben az Alany állapota megváltozik. A Model-View kapcsolatában 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.

3.5.2. Controller

A 3.2. ábrán látható a Controllerek hierarchikus szerkezete.



3.2. 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

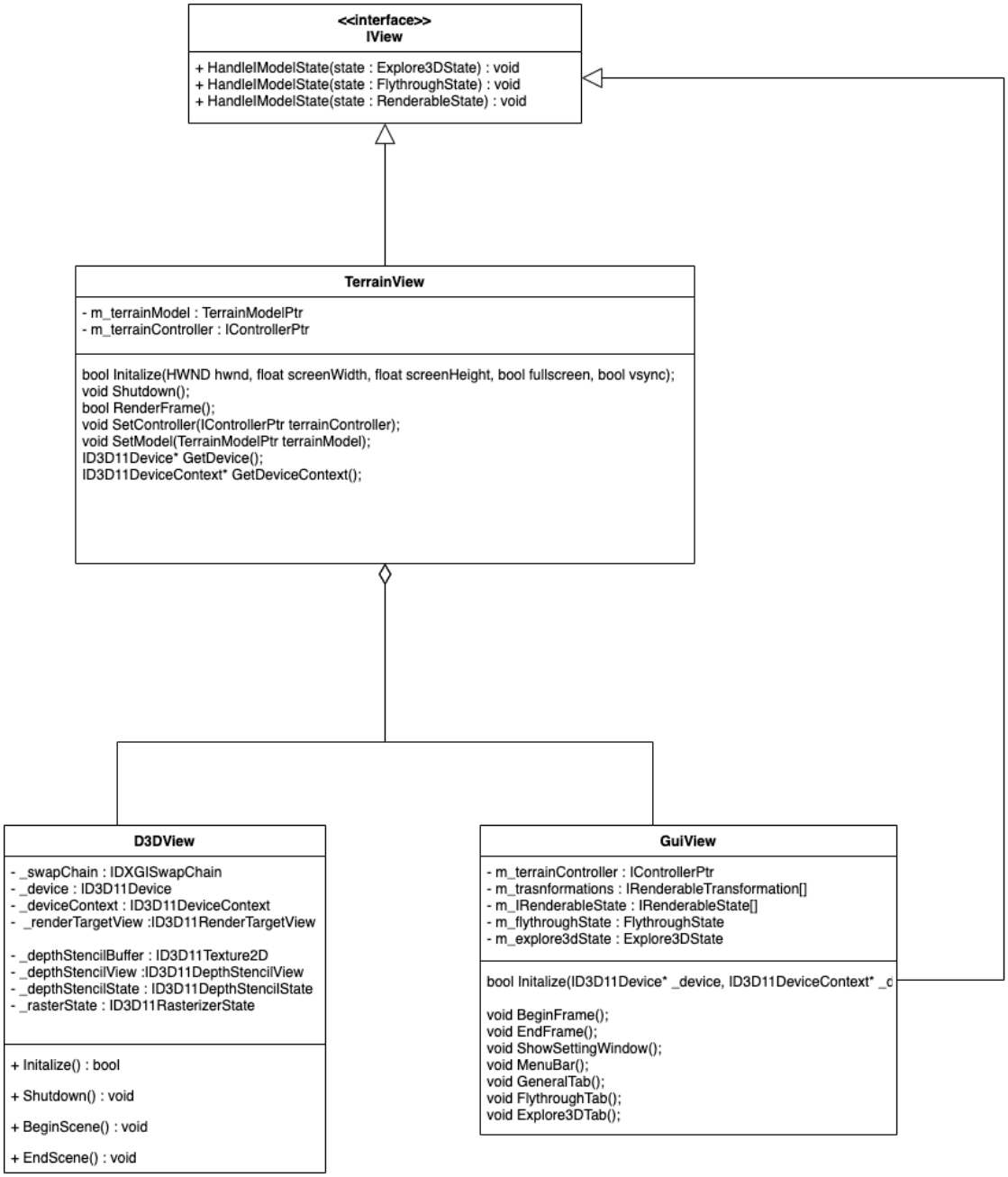
A Controllerek egymás között az Observer minta segítségével kommunikálnak, melyet a MessageSystem objektum tesz lehetővé. Minden Controller referenciával rendelkezik a MessageSystem objektumra, és a Publish metódus segítségével tudnak üzenetet küldeni egy másik Controllernek.

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 );
```

3.1. forráskód. HandleMessage tagfüggvény

3.5.3. View

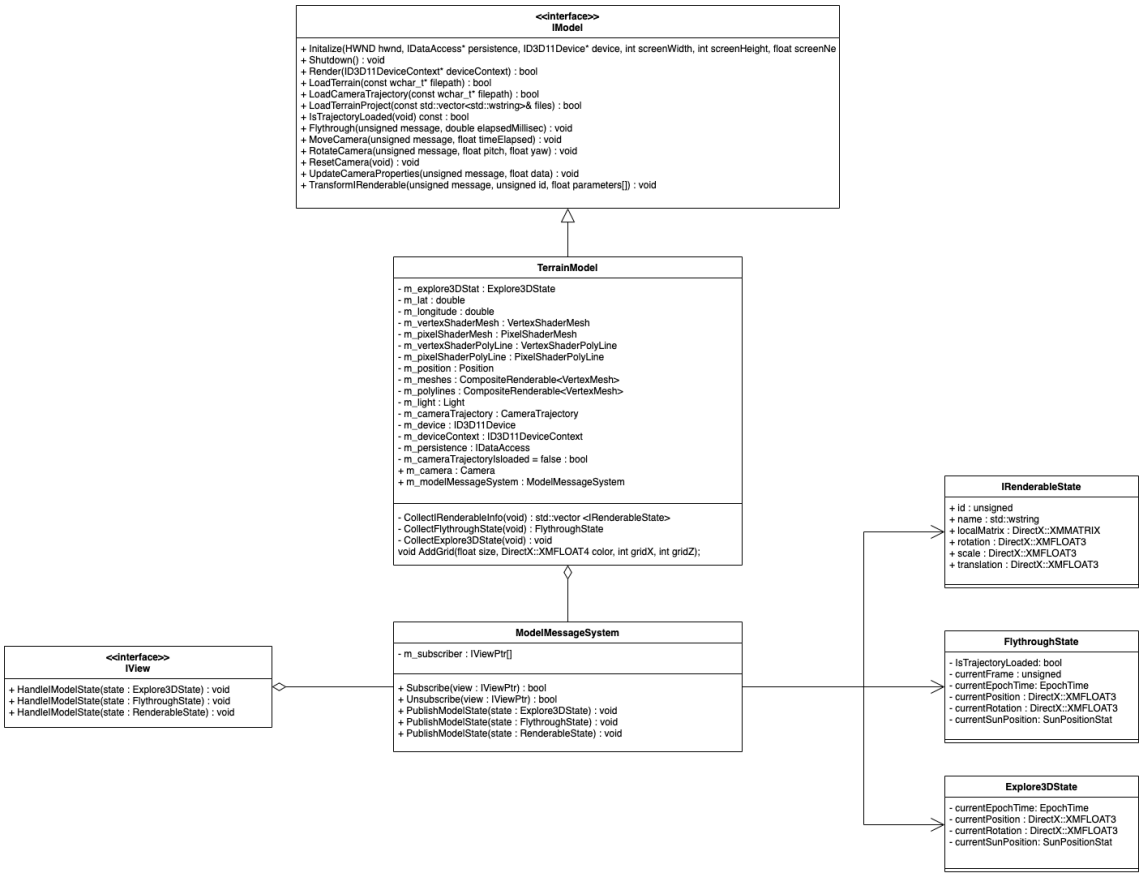


3.3. ábra. View UML osztálydiagram

3.5.4. Model - View

A Model és a View közötti kommunikáció az Observer patternnek megfelelően valósul meg, amint az 3.4. 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.

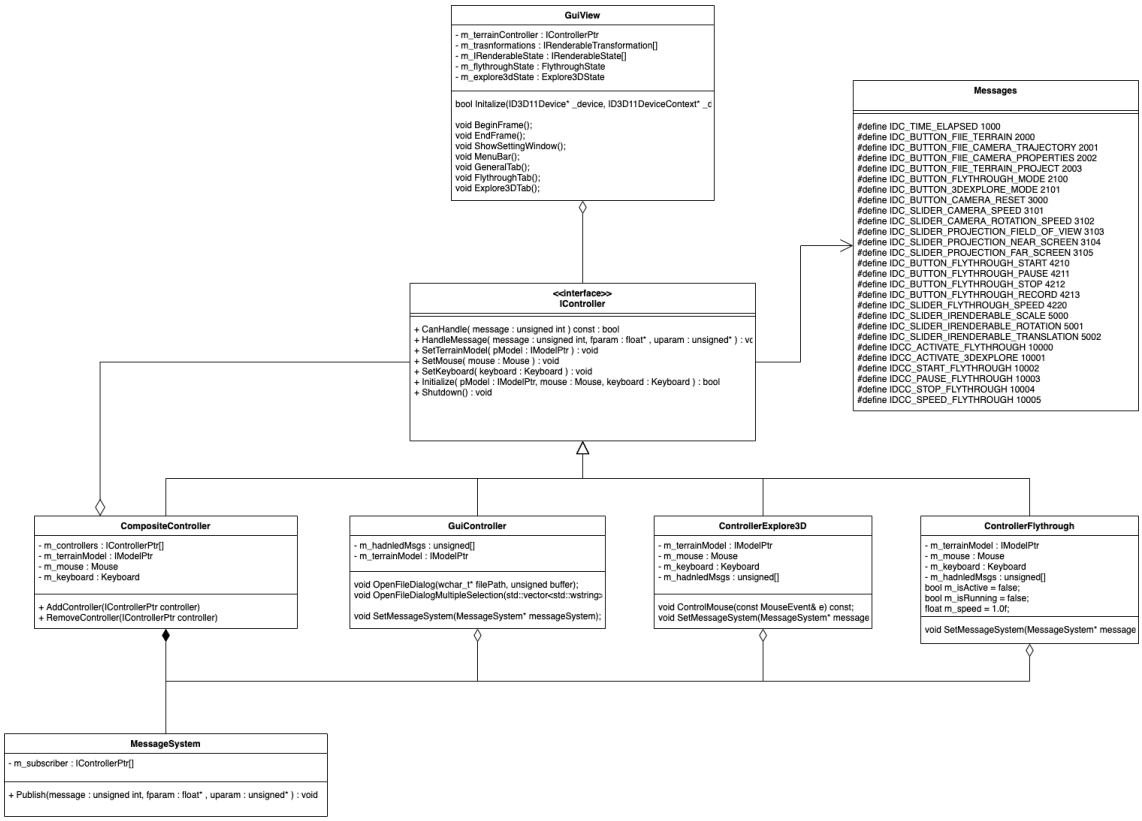
Az Observer pattern-t a ModelMessageSystem objektum biztosítja. Az objektumra IView típusú megfigyelők iratkozhatnak fel. Ha az IModel objektum állapota változik, a ModelMessageSystem objektum a feliratkozókat értesíti a PublishModelState metóduson keresztül.



3.4. ábra. IModel - IView kapcsolat, UML osztálydiagram

3.5.5. Controller - View

A 3.5. 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.



3.5. 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.

4. 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