

TDK-dolgozat

Szerzők:

Magyari Beáta

Kiss Botond

Virtuális valóság keretrendszer

Szerzők:**Magyari Beáta**

Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év

Kiss Botond

Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év

Témavezető:

dr. Simon Károly egyetemi adjunktus,

Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Magyar Matematika és Informatika Intézet

Kivonat

A dolgozat egy olyan keretrendszer megvalósítását mutatja be, amely kiterjesztett valóságon alapuló, több felhasználós online kalandjátékok fejlesztését támogatja okostelefonokon.

Röviden ismerteti a szoftver fejlesztése során felhasznált technológiákat, a rendszer architektúráját, a szerver, illetve az iOS és Android platformokon futó kliensalkalmazások implementációjának néhány részletét, valamint a felhasználási és továbbfejlesztési lehetőségeket.

Szó esik egy konkrét játékról is, amelynek megvalósítása a keretrendszer felhasználásával történt.

Tartalomjegyzék

Bevezető	5
1 Felhasznált technológiák, minták, eszközök	7
1.1 Szerver oldali technológiák	7
1.2 Kliens-szerver kommunikáció	7
1.3 Kliens oldali technológiák	8
1.3.1 Vuforia keretrendszer	8
1.3.2 OpenGL ES	8
1.3.3 Az Android kliens fejlesztéséhez felhasznált technológiák	9
1.3.4 Az iOS kliens fejlesztéséhez felhasznált technológiák	9
1.3.4.1 Objective-C	10
1.3.4.2 iOS SDK 5.1	13
1.3.4.3 Map Kit Framework	13
1.4 Tervezési minták	13
2 A Kiterjesztett valóság keretrendszer	15
2.1 Specifikáció	15
2.2 Elemzés	15
2.2.1 Környezeti elemzés, központi entitások	16
2.2.2 Használati esetek	16
2.3 A szoftver szerkezete	17
2.3.1 Architektúra	17
2.3.2 Az Android kliens	20
2.3.3 Az iOS kliens	20
2.4 Megvalósítás	20
2.4.1 Az Android kliens implementációja	20
2.4.2 Az iOS kliens implementációja	21
2.5 Felhasználói dokumentáció	22
2.5.1 Az Android kliens felülete és funkcionalitásai	23
2.5.2 Az iOS kliens felülete és funkcionalitásai	24
3 A keretrendszer felhasználási lehetőségei	26
Következtetések, továbbfejlesztési lehetőségek	27
Hivatkozások	28

Bevezető

A dolgozat egy olyan keretrendszer megvalósítását mutatja be, amely kiterjesztett valóságon alapuló, több felhasználós online kalandjátékok fejlesztését támogatja okostelefonokon.

A keretrendszer által támogatott kalandjátékok egy viszonylag új játékkategóriát képeznek, alapjuk ellenőrzőpontok elérése, a pontokban adott feladat végrehajtása, például valamilyen információ megszerzése, egy konkrét tárgy megtalálása stb. A Kiterjesztett valóság keretrendszer ilyen típusú játékok fejlesztését támogatja, kibővítve a virtualitás nyújtotta izgalmas lehetőségekkel: az ellenőrzőpontokban a kiterjesztett valóság részeként lephetünk rá a „kincsekre”. A keretrendszer fejlesztését az motiválta, hogy egyre elterjedtebbek az ilyen típusú kalandjátékok, viszont olyan még kevés van, amely kiterjesztett valósággal lenne színezve. Nem utolsó szempont, hogy egy ilyen játék létrehozásánál a konkrét tárgyak elhelyezése sokkal időigényesebb és költségesebb, mint virtuális megfelelőik esetében. Ugyanakkor a kiterjesztett valóság számos más érdekes megoldással is szolgálhat, mivel lehetőséget ad egy adott helyszín "átalakítására".

A kliens-szerver architektúrán alapuló program komplexitása viszonylag nagy, így a tervezés és fejlesztés során a megfelelő fejlesztési alapelvek betartása és az alkalmazott tervezési minták optimális kiválasztása nagyon fontos volt. Nagy hangsúly került a túlterheltség kivédésére, hiszen a központi szerver feladata a teljes játékvezérlés, így sok kliens kiszolgálása is szükséges lehet egy időben (például egy verseny esetében, ahol többtagú csapatok versenyeznek). A dolgozat röviden tárgyalja a keretrendszer architektúráját, emellett a hálózati kommunikáció megvalósítására, valamint a túlterheltség kivédésére alkalmazott módszert is.

A kliensalkalmazás iOS, illetve Android platformra lett fejlesztve. Játék során a szerver által szolgáltatott pontokat a kliens egy térképen jeleníti meg, az odavezető úttal együtt. Ennek segítségével találhatja meg a játékos a pontokat. Mindkét kliens esetében a Google térképpel történő megvalósításhoz használt eszközök nagyon rövid ismertetése is helyet kap a dolgozatban.

Az ellenőrzőpontokban a virtuális valóságban megtalálható tárgyak pontos helyének meghatározása képfelismeréssel történik. A pontos hely megtalálása után látható a kiterjesztett valóságban jelenlévő „kincs”. A felhasználó pozíciójának meghatározása mellett

a képfelismerés szerepe azért is fontos, mert így a későbbiekben a szoftver zárt térben, épületekben is használható lehet. A dolgozatban bemutatásra kerül az ehhez alkalmazott módszer.

Az alkalmazás specifikációja, környezeti elemzése, használati esetek tárgyalása valamint a felhasználói dokumentáció szintén részei a dolgozatnak. Emellett bemutatásra kerül a kliensek implementációjának néhány részlete is.

Végül a felhasználási lehetőségekről, valamint továbbfejlesztésről esik szó.

A program kifejlesztésénél kiindulási pontként szolgált, a Babeş-Bolyai Tudományegyetem, Matematika és Informatika Karán, az informatika szak keretén belül 2011-ban kifejlesztett közös projekt, amelynek hallgatóként részesei voltunk. Ezúton szeretnénk köszönetet nyilvánítani a projekt fejlesztésében résztvevő csoporttársainknak segítségükért, amellyel hozzájárultak a Kiterjesztett valóság keretrendszer alapját képező prototípus fejlesztéséhez. Mind a tervezés, mind a fejlesztés során értékes tanácsokat kaptunk Bálint Zsolttól, köszönet mindezekért. Továbbá szeretnénk megköszönni szakmai irányítónknak, Simon Károly tanár úrnak a segítségét.

1 Felhasznált technológiák, minták, eszközök

Ez a fejezet a szerver és a kliensek fejlesztése során felhasznált eszközöket, mintákat és technológiákat sorolja fel.

1.1 Szerver oldali technológiák

Az alábbi táblázat a szerver fejlesztése során felhasznált technológiákat mutatja be.

Megnevezés	Leírás
MySQL Server 5.5.8	Adatbázisszerver
JDK 1.6	Fejlesztői csomag a Java kód fordításához
Eclipse 3.7.2 (Helios)	Java fejlesztői környezet
MINA 2.0.2	A kliensek és a szerver kapcsolatát megvalósító könyvtár-csomag

1. táblázat: a szerver fejlesztése során alkalmazott eszközök és technológiák felsorolása

1.2 Kliens-szerver kommunikáció

A projekt szerver és kliens része között hálózati kommunikáció létezik. Szerver oldalon a kommunikációt az Apache Szoftver Foundation által kifejlesztett Mina keretrendszer [4] biztosítja. Az Android kliens szintén a Mina keretrendszert használja a kapcsolat kialakításához. A szerver és az Android kliens a TCP/IP protokollt használja, amelyet a MINA a Java NIO API segítségével valósít meg.

Az iOS kliens kommunikációja a szerverrel CocoaAsyncSocket [10] keretrendszer felhasználásával valósul meg, amely hatékony aszinkron feldolgozást biztosít.

A Mina elsődleges előnye az, hogy a szerveren egyetlen végrehajtási szálhoz létre a kapcsolatok lebonyolítására, és ezen a szálon keresztül egyszerre akár több ezer kliens is tud kapcsolódni. Így nem kell feleslegesen költséges szál-létrehozási műveleteket végrehajtani minden kliens kapcsolódásakor. A Mina segítségével a szerver több szinten is fel tudja dolgozni a kliensektől kapott üzeneteket bizonyos szűrők segítségével. A szerverben két szűrő dolgozza fel az adatokat; a LoggingFilter a naplózásért, a MessageCodecFactory pedig a bejövő és kimenő üzenetek kódolásáért és dekódolásáért felelős. A MessageCodecFactory és

az üzenetekkel kapcsolatos osztályok egy külön könyvtár részei, ez biztosítja azt, hogy az Android kliensbe is könnyen integrálhatóak legyenek.

1.3 Kliens oldali technológiák

Ez a fejezet bemutatja a Vuforia [2] keretrendszert, melyet mindkét kliens használ a képfelismeréshez, és amely segíti a kiterjesztett valóság ábrázolását. Továbbá leírja a technológiákat, melyek a kliensek fejlesztésére lettek felhasználva.

1.3.1 Vuforia keretrendszer

A Vuforia a Qualcomm által kifejlesztett keretrendszer, amely egy olyan ablakot biztosít, amelyen keresztül betekinthetünk a kiterjesztett valóságba, ahol a valós és a virtuális világ keresztezi egymást. A kamera élő nézetét a képernyőre vetítve látható a valós világ, majd az élő kamera nézetre helyezett virtuális 3D vagy 2D modell képezi a virtuális világot. A keretrendszer tartalmaz egy képfelismerő algoritmust is, és csak akkor rajzolja ki a virtuális modellt, ha felismeri a kamera által szolgáltatott képet. A képeket melyeket fel kell ismerjen a keretrendszer fel kell tölteni a <http://ar.qualcomm.at> honlapra. Az oldal automatikusan kigenerál egy könyvtárat (Androidnak osztott objektumot - libQCAR.so, iOSnek pedig statikus könyvtárat – libQCAR.a), melyet össze kell kapcsolni a projekttel.

A <http://ar.qualcomm.at> Target Management rendszere segítségével dat és xml kiterjesztésű fájlok állíthatók elő a képekből. Ezen források projekthez való hozzáadásával biztosíthatók az adatok a képfelismeréshez.

A keretrendszer a Singleton minta alapján megvalósított osztályokat tartalmaz, és több fontosabb részből áll. A Camera felelős azért, hogy a kamera által szolgáltatott kép hatékonyan eljusson a képfelismerőhöz. Az Image Converter átalakítja a képet, hogy az megfelelő legyen az OpenGL-nek és a képfelismerőnek. A Tracker tartalmazza a képfelismerő algoritmusokat.

1.3.2 OpenGL ES

Az OpenGL ES (OpenGL for Embedded Systems) a Khronos Group által kifejlesztett ingyenes, platform-független API [1], mely segítségével kezelhető a grafikus kártya, és három dimenziós grafika programozható mobiltelefonokon, PDA-kon és játék konzolokon. Az asztali OpenGL részét képezi. Hatékony és rugalmas interfész a szoftver és a grafikus gyorsítás között. Lebegőpontos és fixpontos rendszereken egyaránt működik.

1.3.3 Az Android kliens fejlesztéséhez felhasznált technológiák

Az Android ingyenes és nyílt forráskódú operációs rendszer, melyet mobiltelefonokra és tabletekre fejlesztettek ki. 2005-ben a Google Inc. megvásárolta az Android Inc.-et, és azóta az Open Handset Alliance többi tagjával együtt fejleszti. Androidra a fejlesztések főként Java programozási nyelvben történnek, de van lehetőség C-ben is fejleszteni. A Google kifejlesztett speciális könyvtárakat, melyek megkönnyítik a fejlesztést. Ezek a könyvtárak hozzáférést biztosítanak a készülékek szenzorjaihoz (accelerometer, gyrometer, GPS stb.) és bizonyos szolgáltatásokat nyújtanak (naplár, névjegyzékek, SQLite, stb.).

A szoftvereket fel lehet tenni az Android Marketre, amely 2012-ben része lett a Google Play-nek, Google Play Apps néven. A Google Play Apps-ról több mint 620000 alkalmazás tölthető le, melyek 57%-a ingyenes.

A projekt fejlesztése során fel volt használva a Google Maps API Android verziója is. Segítségével egy térképet megjelenítő grafikus komponenst fel lehet használni a szoftverekben, melyet a Google Maps JavaScript verziójához hasonlóan lehet kezelni.

1.3.4 Az iOS kliens fejlesztéséhez felhasznált technológiák

Ez a fejezet az iOS kliens fejlesztése során felhasznált eszközöket és technológiákat mutatja be.

Megnevezés	Leírás
Objective-C [5] [6]	iPhone kliens fejlesztéséhez használt nyelv
iOS SDK 5.1 [7]	Fejlesztői csomag iOS-re
Xcode 4.3 [7]	Objectiv-C fejlesztői környezet
Vuforia SDK v1.5	Kiterjesztett valóság fejlesztői csomag
OpenGL ES	Grafikához használt keretrendszer
MapKit [7]	Google Maps használatához szükséges keretrendszer
CocoaAsyncSocket	A kliens-szerver kommunikáció megvalósításához felhasznált keretrendszer.

2. táblázat: az iOS kliens fejlesztése során használt technológiák

Amennyiben Mac szoftverfejlesztésről van szó, így az iOS platformra való fejlesztés esetén is, az Objective-C az elsődleges programozási nyelv. Az iOS platformon futó alkalmazások fejlesztéséhez használt fejlesztői csomag az iOS SDK. A következőkben ezekről a technológiákról esik néhány szó.

1.3.4.1 Objective-C

Az Objective-C elsődleges fejlesztője, Brad Cox, az 1980- as évek elején kezdte el a projektet, majd együtt dolgozott rajta Tom Love- al, StepStone nevű vállalatuk számára. 1985- ben Steve Jobs megalapította a NeXt- et. A NeXt a Unix- ot választotta operációs rendszernek, majd megalkották a NextStep-et, amely egy erős felhasználói felület eszközkészlet lett. Az Objective-C licenszének megvásárlása lehetővé tette, hogy a NextStep és minden fejlesztői alkalmazás megírásához ezt használják.

Amikor az Apple megszerezte a NeXt-et 1996- ban, a NextStep a Cocoa nevet kapta, ezt vették a Mac OS X alapjául. Az operációs rendszernek az NeXT fejlesztői rendszere, a Project Builder és Interface Builder, valamint az Objective-C is része volt. A Project Builder- ből alakult ki később az Xcode, amely az elsődleges fejlesztői környezet az Objective-C számára.

Az Objective-C és a Cocoa az idők során elegáns és erőteljes eszközkészletté alakult. Az Objective- C jelenti a kulcsot az iPhone applikációk megírásához is.

Objektumorientált nyelv, mely ötvözi a népszerű és hordozható C -t az elegáns Smalltalk¹- al.

Bármely C program kompilálható az Objective- C kompilátorával, használhatunk C kódot Objective-C osztályokban.

Az Objective- C objektumorientált lehetőségeit illető szintaxisa a Smalltalk stílusú üzenetküldés implementációja. A nem objektumorientált műveletek (primitív változók, előfeldolgozás, kifejezések, függvénydeklarációk, függvényhívások) megfelelnek a C szintaxisának.

A nyelv objektumorientált programozási modellje az objektumoknak való üzenetküldésen alapszik. Nem egyszerű függvénymeghívás történik, hanem üzenetküldés. Objective-C-ben az üzenet által közvetített feladat futási időben dolgozódik fel a fogadó objektum által. A metódus neve nincs meghatározva a fordító által a kódszegmensben, mint a Simula² stílusú nyelvekben. Objective-C-ben az üzenet egy szelektorral van azonosítva, a tárbeli cím futási időben van csak meghatározva. Ennek következménye, hogy nem történik típusellenőrzés. A

¹ A Smalltalk objektumorientált nyelv, az 1970-es években fejlesztették a Xerox Parc vállalatnál, többek között Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Scott Wallace.

² A Simula az 1960-as évekbe visszanyúló, egyik legkorábbi objektumorientált programozási nyelv. Szimulációs célokra fejlesztették Osloban.

fogadó objektum lehet, hogy nem tud reagálni az üzenetre, ebben az esetben kivétel keletkezik, mivel NULL pointert ad vissza.

C++-ban az üzenetküldés a következőképpen valósítható meg:

```
obj -> method(argument);
```

Ugyanez Objective-C-ben a következő:

```
[obj method:argument];
```

Objective-C-ben az interfész header fájlba kerül (általában .h a kiterjesztés), a megvalósítás pedig implementációs állományba (általában .m kiterjesztés). A header fájlok beillesztése az implementációs fájlokba az #import direktíva használatával valósul meg. Ez garantálja, hogy egy bizonyos header fájl csak egyszer kerüljön beillesztésre az illető implementációs fájlba.

Példa interfészre:

```
@interface osztály : őszosztály {
    típus változó;
}
+(típus)osztálymetódus:(típus)paraméter;

-(típus)példánymetódus:(típus)paraméter;
```

A fenti interfész implementációja a következő formában történik:

```
@implementation osztály
+(típus)osztálymetódus:(típus)paraméter {
    //implementáció
}

-(típus)példánymetódus:(típus)paraméter {
    //implementáció
}
@end
```

Objective-C-ben egy osztály példányosítása két lépésben történik. Első lépés az alloc metódussal való memóriefoglalás az objektum számára, majd szükséges az inicializálás egy init metódussal. Az init metódus lehet alapértelmezett, azaz paraméter nélküli, mely esetben a példányosítás a következőképpen néz ki:

```
Osztály *objektum = [[Osztály alloc] init];
```

Az init metódusnak használható, és megírható bármilyen paraméteres változata:

```
Osztály *objektum = [[Osztály alloc] initWithValue:value];
```

Objective-C-ben egy már létező osztályhoz kategória segítségével adhatunk hozzá új metódust. Példa:

```
@interface Osztály (Kategória)
-(típus)metódus;
@end

@implementation Osztály (Kategória)
-(típus)metódus {
//implementáció
}
@end
```

Kategóriák segítségével oldható meg a privát metódusok használata. Objective-C-ben nincsenek kimondottan privát metódusok, a kívánt eredmény viszont elérhető egy kategória behelyezésével az osztály implementációs fájljába. Így a kategóriában szereplő metódusok nem lesznek láthatóak más osztályok számára.

Objective-C-ben beszélhetünk informális és formális protokollokról. Az informális protokoll egy olyan metódusokat tartalmazó lista, melyet egy osztály implementálhat.

Az informális protokoll a gyakorlatban egy kategória elhelyezése egy osztályon.

A formális protokoll ugyancsak egy metóduslista, melyet egy osztály adoptálhat. Ha egy osztály megfelel egy protokollnak, akkor lehetnek olyan metódusok a protokollban, melyeket mindenképpen kell, hogy implementáljon az illető osztály.

Az Objective-C-ben gyakran használt Delegate tervezési minta protokoll használatával valósítható meg.

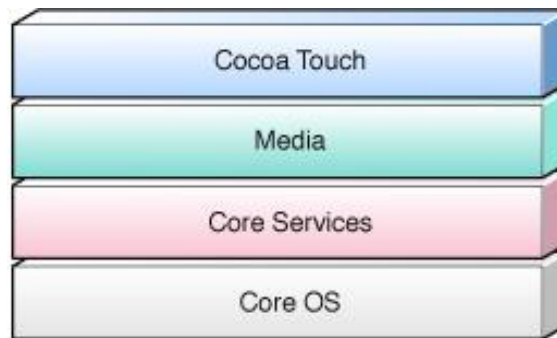
Egy adattag property-ként való deklarálásával, a @property direktíva segítségével, valamint a @synthesize használatával az implementációs fájlban, automatikus elérő metódusok generálódnak a kompilátor által. A property-knek megadhatók különböző opciók (readonly, readwrite, retain, assign, copy).

A Cocoa által használt memóriakezelési technika a referencia számlálás (reference counting technique vagy retain counting). Minden objektumhoz egy egész szám tartozik, ez az objektum referencia száma. Az objektum használatakor a kód növeli ezt a számot, amikor

pedig végzett vele, csökkenti. Amikor ez a szám 0 lesz, az Objective-C automatikusan dealloc üzenetet küld az objektumnak, így az felszabadul.

1.3.4.2 iOS SDK 5.1

Az iPhone, iPod touch és iPad készülékek operációs rendszere az iOS, amely biztosítja a natív- és rendszer applikációkhoz szükséges technológiákat a készülékeken, és felügyeli a hardvert. Az iOS négy réteget foglal magába, amelyet az alábbi ábra illusztrál:



1. ábra: az iOS rétegei

Forrás: <https://www.developer.apple.com>

Az iOS SDK az a keretrendszer, amely tartalmazza mindazokat az interfészeket és eszközöket, amelyek a natív applikációk fejlesztéséhez, telepítéséhez és futtatásához szükségesek. Az első verzió 2008-ban volt kiadva, ez volt az iOS 1.2b1 Beta 1. A legutóbbi verzió, az iOS 5.1 final 2012 márciusában látott napvilágot.

Az SDK-ban található csomagok az iOS rétegei szerint vannak csoportosítva, ezeknek a csomagoknak a használatával valósul meg a fejlesztés.

1.3.4.3 Map Kit Framework

Az iOS platformot célzó kliens fejlesztése során a térképhasználat a Map Kit csomag használatával valósult meg.

A térképek beágyazását az alkalmazások nézeteibe, ablakaiba, megjegyzések, rétegek hozzáadását a térképhez, a helymeghatározásokat koordináták alapján a Map Kit csomagban található interfészek biztosítják. A használathoz ezeknek az interfészeknek az implementálása szükséges. A térképadatokat Google szolgáltatásoktól nyeri a felület.

1.4 Tervezési minták

A szoftver fejlesztése során a megfelelő fejlesztési alapelvek betartása és az alkalmazott tervezési minták optimális kiválasztása nagyon fontos volt.

A szerver architektúrája többretegű, a multitier architecture tervezési mintát követi.

A szerver több alapvető tervezési minta alapján készült: Singleton [3], Abstract Factory [3], Command [3].

A Singleton tervezési minta lényege, hogy az illető osztályból egyetlen példány jöjjön létre. Singleton minta alapján készült például az adatbázishoz történő kapcsolódást megvalósító komponens.

A szerver és az adatbázis közötti kapcsolat interfészek segítségével valósul meg. Az adathozzáférési réteg az Abstract DAO Factory mintára épül, amelynek köszönhetően sokkal könnyebb a szervert módosítani, ha az adatbázis szerkezete módosul, vagy át kell térni más adatbázis-kezelő rendszerre, technológiára.

A szerverhez érkező utasításokat (parancsokat) végrehajtó osztályok a Command, illetve a Singleton mintára épülnek. A Command tervezési minta használata előnyös, mert egy új funkcionalitás hozzáadása nem vonja maga után a már implementált részek módosítását. Ez biztosítja, hogy könnyű legyen a szerver bővítése új parancsokkal.

A kliensalkalmazások az MVC (modell-nézet-vezérlő) tervezési mintát követik. A Command és Singleton tervezési minta több helyen volt alkalmazva az iOS kliens fejlesztése során is. Ezek mellett a Delegate [11] tervezési minta használata is fontosnak bizonyult az iOS klienset tekintve.

2 A Kiterjesztett valóság keretrendszer

A Kiterjesztett valóság keretrendszer célja, hogy lehetőséget biztosítson, olyan hálózaton keresztüli, okostelefonokra írt kalandjátékok fejlesztésére, amelyekben a kutatott „kincsek” a kiterjesztett valóság elemei. A keretrendszer megfelelő elemeinek testreszabásával fejleszthetők a játékok.

2.1 Specifikáció

A szerver funkcionalitásai:

- Adatbázis kezelése
- Információk szolgáltatása a kliensekhez
- Játékvezérlés

A kliens funkcionalitásai:

- Regisztráció lehetőségének biztosítása
- Bejelentkezés kezelése
- Kijelentkezés kezelése
- Elérhető játékok hozzáférhetővé tétele
- Játékok közül való választás lehetőségének biztosítása
- Térképhasználat biztosítása
- Lehetőség ellenőrzőpontok lekérésére
- Pontok megjelenítése a térképen
- Pontokhoz vezető legrövidebb út³ábrázolásának lehetősége
- Kamera használata
- Képfelismerés
- Kiterjesztett valóság ábrázolása
- Szerver értesítése az ellenőrzőpont megtalálása után

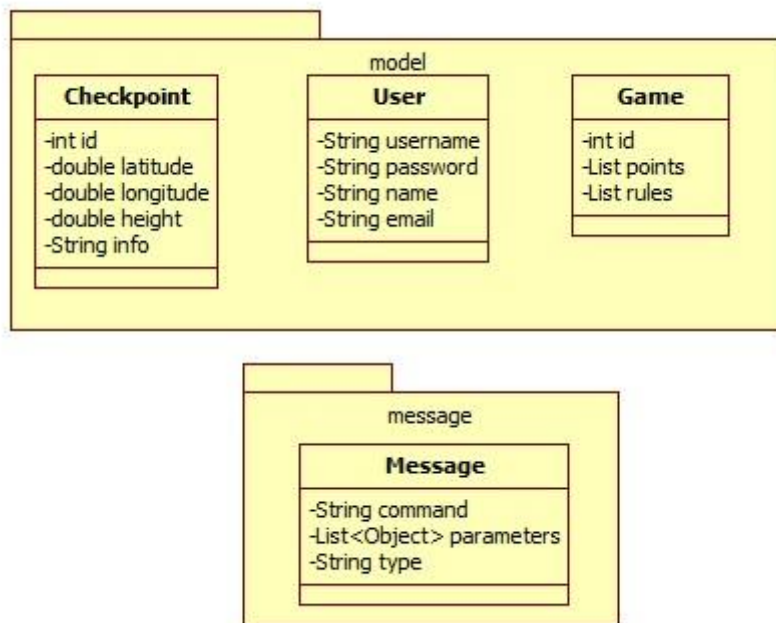
2.2 Elemzés

A szoftver elemzése magába foglalja a környezeti elemzést, a központi entitások beazonosítását, továbbá vázolja a használati eseteket.

³ A legrövidebb út itt a Google adatai alapján meghatározottat jelenti.

2.2.1 Környezeti elemzés, központi entitások

A szoftver alapvető entitásai a 2. ábrán szemlélhetők.



2. ábra: alapvető entitások

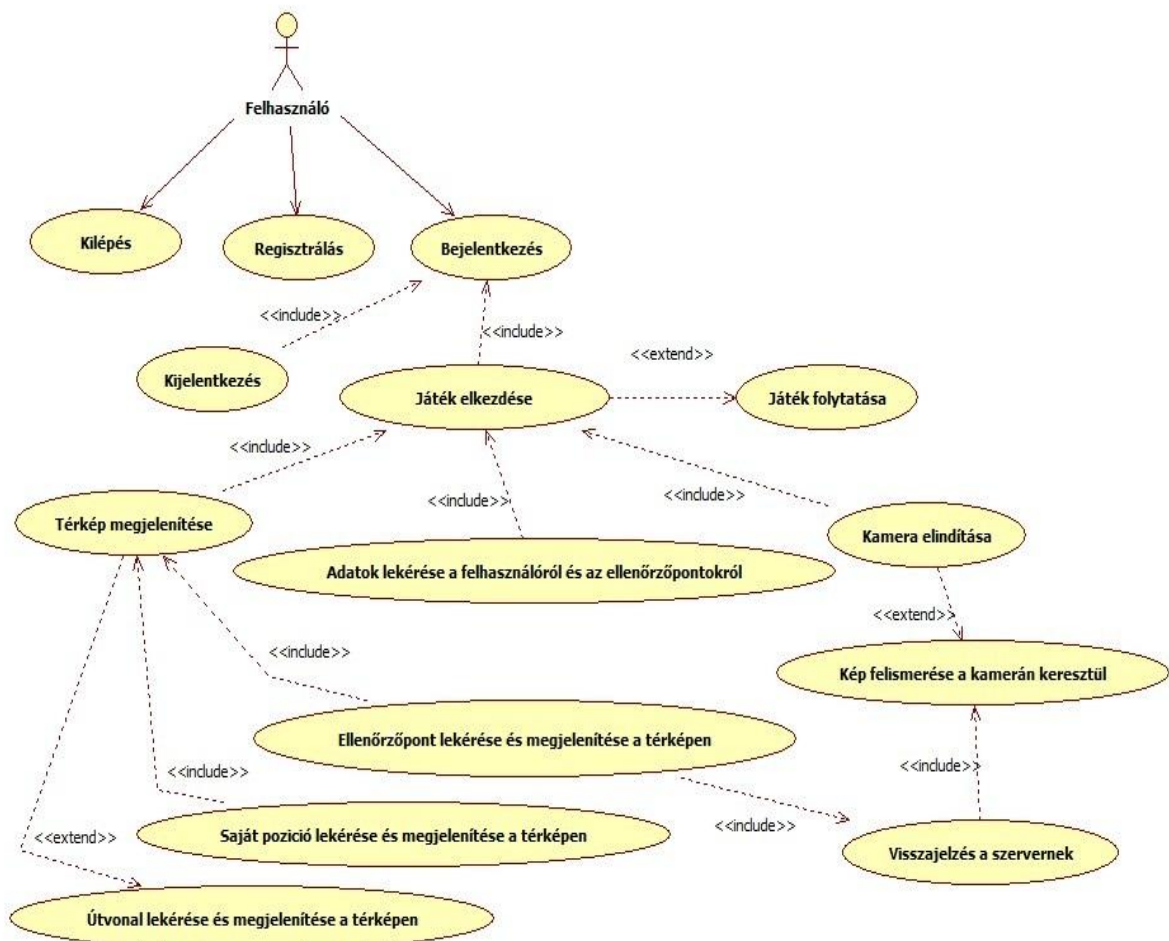
A program központi entitásai a következők:

- Checkpoint: az ellenőrzőpontok modell osztálya
- User: egy felhasználónak megfelelő modell osztály
- Game: egy játékot leíró osztály

A fenti osztályok a model csomag részei. Egy másik fontos entitás a message csomagban található Message osztály, amely kliens és szerver közötti kommunikáció protokolljának szerves része. A kliens és szerver közötti üzenetek Message-ekké konvertálódnak.

2.2.2 Használati esetek

A felhasználó első dolga a regisztráció. Sikeres regisztráció után be tud jelentkezni, majd elkezdeni egy új játékot. A játékot bármikor abba lehet hagyni, majd folytatni. Sikeres bejelentkezés és játékválasztás után a következő ellenőrzőpont lekérése történik a szervertől. A felhasználó kérheti az útvonal kirajzolását a saját pozíciója és az ellenőrzőpont között. Ha a felhasználó a GPS és a kamera segítségével megtalálta az ellenőrzőpontot, a kliens visszajelez a szervernek, majd folytatja a "kutatást" a többi ponttal.



3. ábra: használati esetek

2.3 A szoftver szerkezete

A szoftver három alrendszerből tevődik össze: Android kliens, iOS kliens és szerver. A következőkben bemutatásra kerül a keretrendszer szerkezetét illető néhány fontosabb részlet.

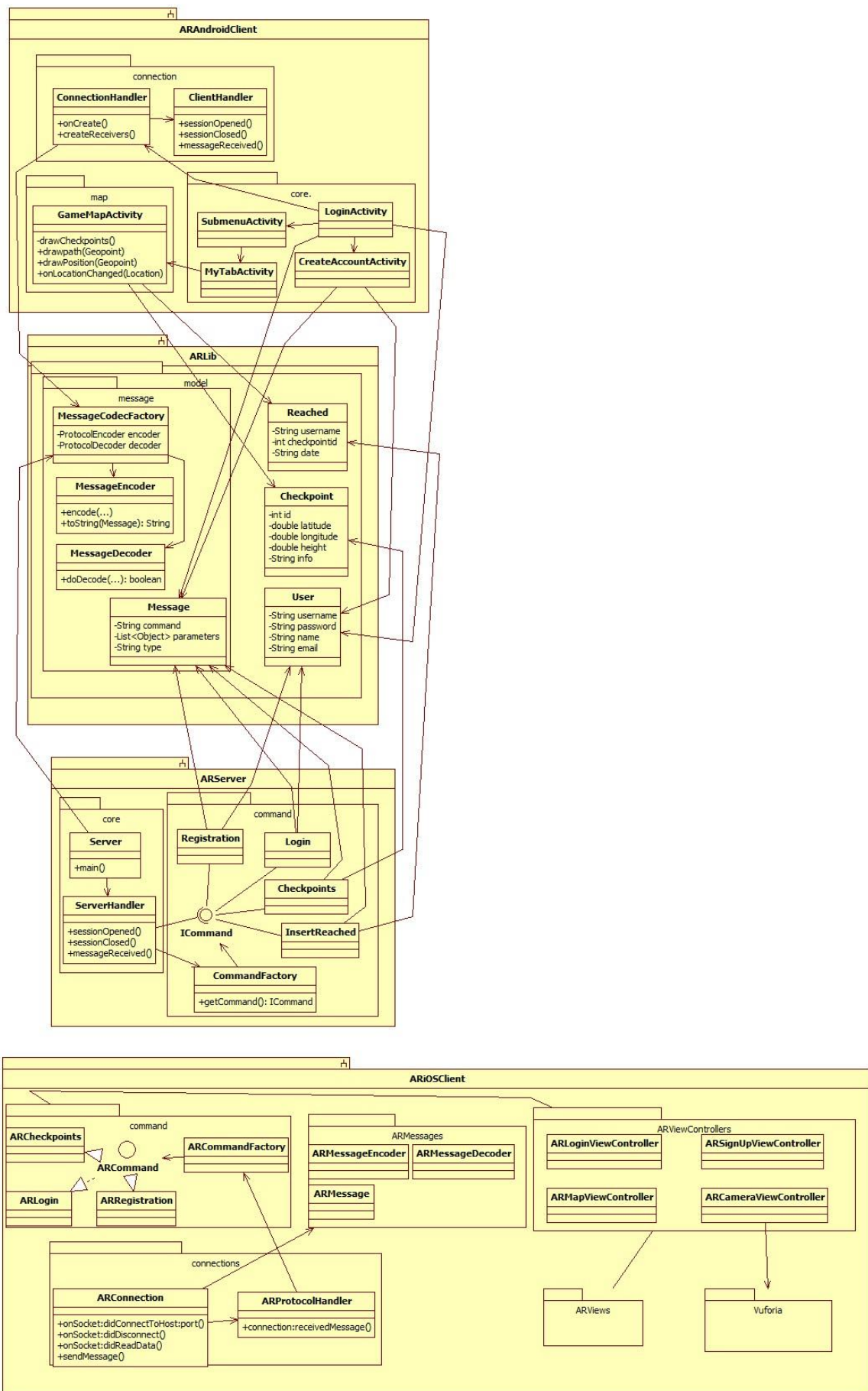
2.3.1 Architektúra

A szerver és kliens részen található csomagokat, ezeken belül a fontosabb osztályokat és a közöttük lévő kapcsolatokat a 4. ábra szemlélteti.

A szerverhez érkező üzenetek a `ServerHandler` osztály `messageReceived` metódusába érkeznek. Minden üzenet külön osztályban kerül feldolgozásra, ez a megvalósítás a Factory tervezési minta alkalmazásával történik. A parancs osztályok a Singleton tervezési minta szerint implementáltak, a `command` csomagban találhatóak. A `game` csomag tartalmazza a játékban alkalmazandó szabályokat leíró osztályokat. Az adatbázis kapcsolatért felelős osztályok a `jdbc.connection` csomagban találhatóak. A `jdbc` csomag az adatbázisban tárolt adatok eléréséért és feldolgozásáért felelős, az `ARLib` részben található modell csomaggal

együtt. Az alkalmazás a többrétegű szoftver architektúra alapelveit követi. A Data Access Object (DAO) tervezési minta alapján valósítja meg az adatbázisban tárolt adatok elérését. Az adatelérési réteg (Data Access Layer – DAL) az adatok lekérdezéséért felelős, a szolgáltatás réteg (Service Layer – SL) a lekérdezett adatok segítségével felépíti a megfelelő modell objektumokat, és ezeket továbbítja a következő réteghez. A rétegeknek megfelelő interfészek egy-egy konkrét implementációjának elérése Factory tervezési minta alkalmazásával történik.

A kliensek architektúrájának fontosabb részleteit a következő két fejezet írja le.



4. ábra: a szoftver alrendszerei, fontosabb csomagok, osztályok, kapcsolatok

2.3.2 Az Android kliens

Az Android kliens hat csomagból áll, melyek közül négy fontosabb. A Core csomag a bejelentkezési, regisztrálási és játékválasztási felületeket tartalmazza. A Connection csomag a felelős a kliens és a szerver közötti kapcsolat kialakításáért és az üzenetek elküldéséért. A Map csomag felel a térkép és az ellenőrzőpontok megjelenítéséért, illetve az útvonal kirajzolásáért. Az ImageTargets csomagban van implementálva a kamera használat, a képfelismerés és a kiterjesztett valóság megjelenítése.

2.3.3 Az iOS kliens

Az iOS kliens architektúráját tekintve a legfontosabb csoportok a következők:

- ARConnections: a kliens-szerver kapcsolat kezelését végző osztályok.
- ARMessages: a kliens és szerver között kialakított kommunikáció protokollja szerinti üzenet felépítéséért felelős osztályok.
- ARMainViewControllers: az alapvető nézetek vezérlői.
- ARGameViewControllers: a játék nézeteinek vezérlői.
- ARViews: grafikus felületek.
- ARVuforia: a Vuforia keretrendszer integráláshoz szükséges, valamint a háromdimenziós modell megjelenítéséért felelős osztályok.
- ARCommands: parancsosztályok
- ARMapAnnotations: a pontok térképen való kezelését végző osztályok

2.4 Megvalósítás

Az alábbiakban röviden bemutatásra kerül néhány részlet a kliensek implementációját illetően.

2.4.1 Az Android kliens implementációja

A kliens indításakor a ConnectionHandler objektum (a Connection csomag egy osztályának példánya) megnézi, hogy van-e internet kapcsolat. Ha nincs, akkor megállítja a klienst, különben létrehozza a kapcsolatot a szerverrel. A ConnectionHandler felelős az üzenetek küldéséért is a szervernek. A ClientHandler (szintén a Connection csomag egy osztálya) objektum fogja megkapni a válasz üzeneteket a szervertől.

A Core csomag négy felületet tartalmaz. A CreateAccountActivity nyújtja a regisztrálási lehetőséget, a LoginActivity a bejelentkezést. A SubmenuActivity-ben lehet kiválasztani, hogy új játékot kezd a felhasználó, vagy folytat egy már elkezdett (természetesen ha nincs

ilyen, akkor nem lehet kiválasztani ezt az opciót). A MyTabActivity tartalmazza a paneleket, melyek között navigálhatunk (térkép, kamera, adatbázis).

A Map csomag egy felülettel rendelkezik. A GameMapActivity a térképet jeleníti meg. A MapPointOverlay hozzáadja a térkép felületéhez a keresett ellenőrzőpontot, míg a MapPositionOverlay a felhasználó saját pozícióját adja hozzá a térképhez a GPS segítségével. A MapRouteOverlay segítségével lehet útvonalat kérni a térképtől. A MyMapOverlay felelős azért, hogy az ellenőrzőpontok, a saját pozíció, illetve az útvonal megjelenjen a térképen.

A Database csomag DatabaseActivity-je a harmadik felület a MyTabActivity-n belül. Ez kéri le a szervertől és jeleníti meg a felhasználóval és az ellenőrzőpontokkal kapcsolatos információkat. A Types csomag két típus felsorolást tartalmaz. Ezek határozzák meg, hogy a szervernek milyen parancsokat lehet küldeni. Az Vuforia csomag a Vuforia keretrendszer része.

2.4.2 Az iOS kliens implementációja

Az iOS kliensben használt hálózati kapcsolatért felelős keretrendszer az AsyncSocket. Az ARConnection osztály hoz létre egy kapcsolatobjektumot, a keretrendszer megfelelő osztályának példányosításával. Az objektum connectToHost:onPort metódusa végzi a kapcsolódást a szerverhez. A Delegate tervezési minta szerint, AsyncSocketDelegate protokollban meghatározott metódusok implementálása révén értesül a kliens a kapcsolat létrejöttéről, megszakításáról, az adatok kiolvasásáról. Az üzenet dekódolás után az ARConnectionDelegate protokollt implementáló ARProtokollHandler osztályhoz továbbítódik. Itt a Factory tervezési minta alapján megtörténik az üzenet parancsá való átalakítása, és a parancs végrehajtása. A parancsosztályok implementálása a Singleton tervezési minta alapján történt. A parancs végrehajtása során notification-t kap a megfelelő controller osztály, majd az végzi a további irányítást a felhasználó interakcióira válaszolva.

Az ARMapViewController osztály a térképnézet vezérlője, amely az Map Kit MKMapViewDelegate protokollt implementálja. Ez az osztály kapja meg az ellenőrzőpontokat az ARMapAnnotationsHandler egy példányától, és megjeleníti azokat a térképen a protokollnak megfelelően. Az ARMsgAnnotationsDataSource osztály felelős az ellenőrzőpontok szolgáltatásáért a térképnek. Egy ilyen objektum küld kérés a szervernek a pontokért, és adja tovább az ARAnnotationsHandler példányának az eredményt, amely értesíti a térkép vezérlőjét.

Az ARCameraViewController-ben történik a Vuforia keretrendszer inicializálása, a felismerendő képek elhelyezése a megfelelő holder-ekbe, a keretrendszer ARParentViewControllerének példányosítása, amely tartalmazza azt a nézetet, amelynek része lesz a kiterjesztett valóságért felelős felület.

Az AREAGLView osztály, a Vuforia AR_EAGLView származtatottja, felelős a kiterjesztett valóság nézetért, ez végzi a kamera képére való rárajzolást. Az ellenőrzőpontok kamerával való keresgélése során, a Vuforia keretrendszer képfelismerő algoritmusai dolgoznak a háttérben. Az AREAGLView példányában kell felépíteni a modelleket tartalmazó tömböket a háromdimenziós modellből exportált C- és Objective-C adatokból. Továbbá meg kell határozni az egyszerre megjelenítendő modellek számát, hogy a keretrendszer a kereteket ennek megfelelően elhelyezhesse a nézetben.

A Vuforia a renderFrameQCar metódust hívja meg folyamatosan egy háttérszálon, hogy renderelje az aktuális nézetet. Itt a modell kirajzolása OpenGL függvényekkel történik. A Vuforia Tool namespace-ében implementált convertPose2GLMatrix metódusának a meghívásával mátrixszá alakítódnak az aktuális modellhez tartozó adatok. Ezután a mátrixon elvégződnek a megfelelő translációk, skálázások, szorzások, hozzáadódik a modell anyaga, amely png kiterjesztésű képként kell legyen meghatározva. A glDrawElements meghívása eredményezi a kirajzolást a kamera képére.

Az implementáció folyamán nagy hangsúly került arra, hogy a tervezés során jól kidolgozott rendszer követve legyen, mivel az iOS platformra tervezett szoftverek esetében rendkívül fontos a megfelelő osztályhierarchia, az osztályközi kapcsolatok optimális kialakítása.

2.5 Felhasználói dokumentáció

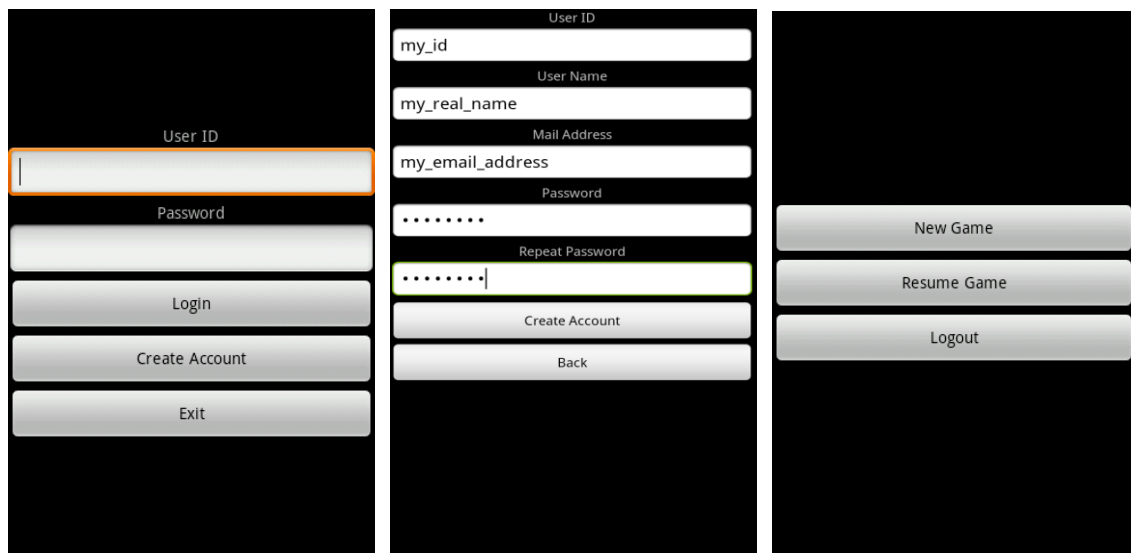
Az 1990-es évek elején még szenzációnak számító telefonok napjainkban már közismertté és könnyen hozzáférhetővé váltak. Egyre nagyobb teljesítményük arra ösztönzi az embereket, hogy telefonjaikat lecseréljék egy szinte számítógépet helyettesítő készülékre.

Az okostelefonok mellett szól az is, hogy az olyan funkcionalitások, mint a Wireless, GPS, illetve iránytű számos applikációban felhasználhatóak, akár informatív, akár szórakoztató jelleggel. Ilyen applikációk fejlesztését támogatja a Kiterjesztett valóság keretrendszer. A keretrendszer felhasználásával megvalósított kalandjátékok lehetőséget nyújtanak a felhasználónak arra, hogy a jól ismert kincskeresést már telefonjáról is játszassa. Az ellenőrzőpontokat meg kell találni a fizikai világban, majd a kincset a kiterjesztett valóságban a telefon kamerájának segítségével, a kijelölt helyen.

A továbbiakban a keretrendszer felhasználásával megvalósított játékok felülete és funkcionalitásai kerülnek bemutatásra. Mindez a két kliens esetében nagyon hasonló, hiszen a cél az volt, hogy a felületet és funkcionalításokat tekintve az Android és iOS platformokra fejlesztett alkalmazás ugyanaz legyen.

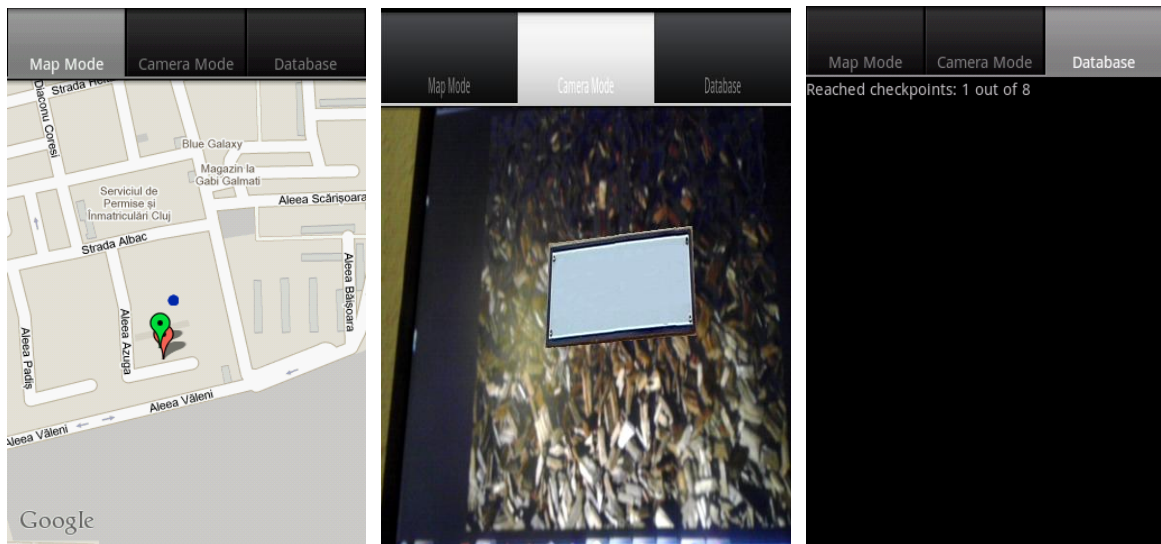
2.5.1 Az Android kliens felülete és funkcionalitásai

A kliens indításakor az első felület a bejelentkező oldal, ahol két szövegmező (felhasználónév és jelszó) és három gomb található (bejelentkezés, regisztrálás és kilépés). Sikeres belépés/regisztrálás esetén a következő felületen három opció közül választhat a felhasználó: új játék kezdése, egy már elkezdett játék folytatása, illetve kijelentkezés. A kijelentkezés visszavisz az első (bejelentkező) felületre. A következő ábrák szemléltetik a bejelentkező-, regisztrációs- és kezdőoldalt .



5. ábra: bejelentkezés-, regisztráció- és kezdőoldal

Ha elkezdünk egy új játékot (vagy folytatunk egy előzőt), akkor a következő felület egy úgynevezett TabActivity lesz, melyen három panel között lehet navigálni. Az első panel behozza a térképet, melyen meg lehet tekinteni a következő ellenőrzőpontot, és útvonalat is lehet kérni a saját pozíciótól az ellenőrzőpontig. A második panel inicializálja a Vuforia keretrendszert és elindítja a kamerát. Az ellenőrzőpontot kamerán keresztül is azonosítani kell. A harmadik panelen a kliens lekérhet a szervertől a felhasználóval és az ellenőrzőpontokkal kapcsolatos információkat.

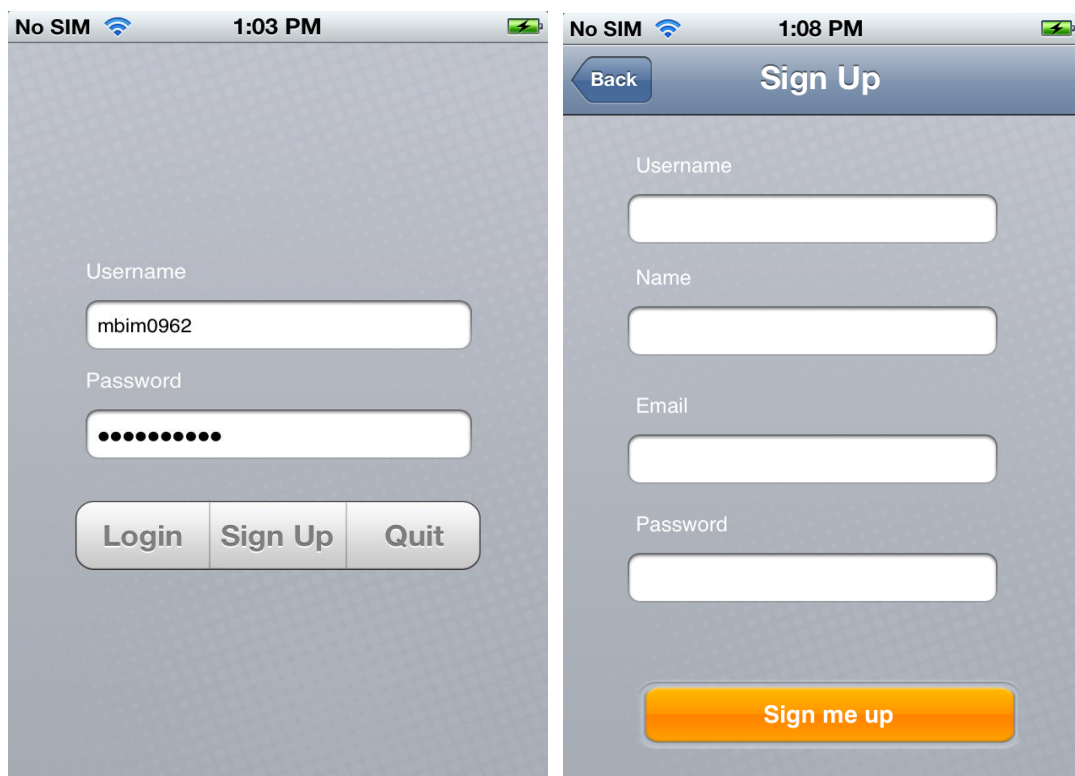


6. ábra: térkép-, kamera- és adatlekérési nézet

2.5.2 Az iOS kliens felülete és funkcionálisai

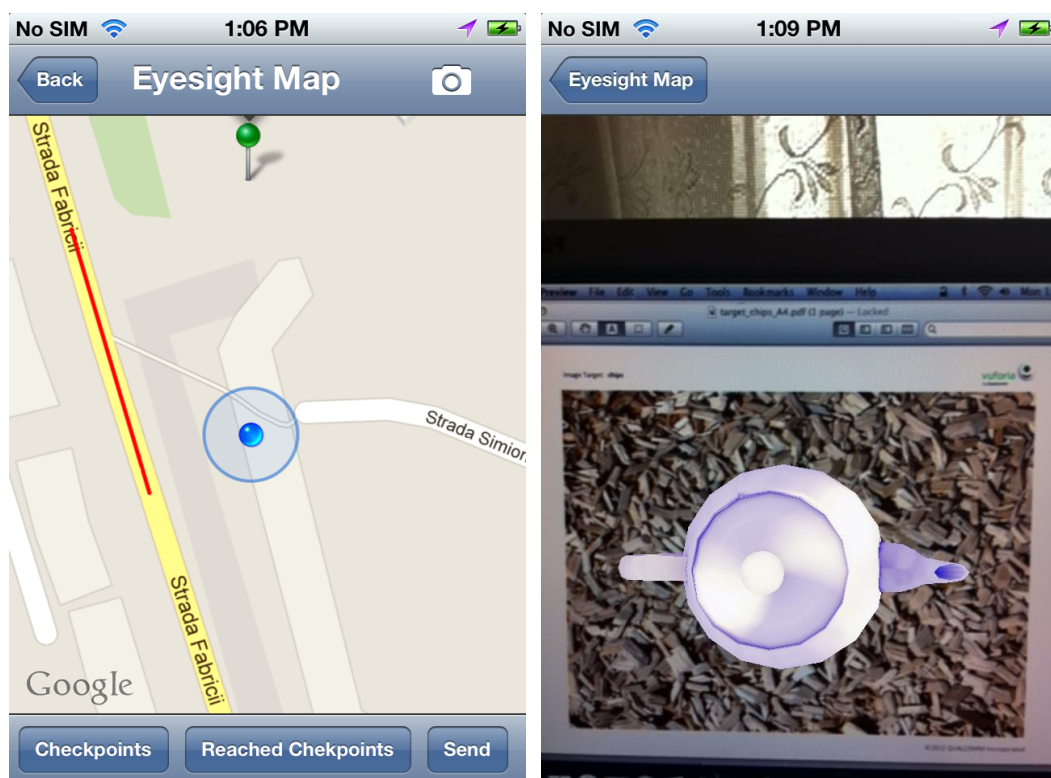
Az alkalmazás elindítása a bejelentkező oldal megjelenését eredményezi, ahol található két szövegmező (egy a felhasználónév és egy a jelszó számára), egy gomb a bejelentkezéshez, egy a programból való kilépéshez, és egy a regisztrációs oldalra való navigáláshoz.

Sikeres bejelentkezés/regisztráció után a felhasználó egy olyan nézethez jut, melyben választhat a gombok segítségével, hogy új játékot kezd, folytat egy már folyamatban lévő vagy esetleg kijelentkezik.



7. ábra: bejelentkezés és regisztráció

Játék során megkapja a megfelelő ellenőrzőpontokat, indulhat a „kincskeresés”. A térképen láthatóak a célpontok, egy pont kiválasztásával lekérhető az odavezető legrövidebb út. Egy gomb érintésével áttérhet a kamera nézetre. Amikor a játékos elérkezik egy ellenőrzőponthoz, a kamera segítségével találhatja meg a „kincset”, ezután a Send gombbal jelezheti a szervernek a sikeres felfedezést.



8. ábra: térkép és kamera nézet

3 A keretrendszer felhasználási lehetőségei

A Kiterjesztett valóság keretrendszer lehetőséget biztosít olyan hálózaton keresztül, okostelefonokra célzott kalandjátékok fejlesztésére, melyekben a kiterjesztett valóságban folyik a „kincskeresés”.

Egy ilyen kalandjáték fejlesztése a keretrendszer használatával a megfelelő elemek testreszabását igényli. A jól meghatározott felületeknek köszönhetően a testreszabás könnyű, a módosítandó réteg ezek révén illeszkedik a keretrendszer többi rétegéhez. Miután megvan az ötlet a játékhoz, ki lett találva a szabályrendszer (például egy történetre alapozva), a szerver megfelelő interfészének implementálása szükséges, ezeknek megfelelően. Az ellenőrzőpontok meghatározása a következő lépés, így rögzíthetők a játék helyszínei. Továbbá szükséges annak a megadása, hogy egy helyszínen, pontosan hol legyen megtalálható a keresett dolog, ezért egy kép kell erről, mely hozzáadódik az alkalmazáshoz (ezt a képet fogja azonosítani a keretrendszer képfelismerője). A pontokban megjelenítendő tárgyak, képek meghatározása is szükséges, ezek lehetnek két- vagy akár háromdimenziósak is. A háromdimenziós tárgyak megszerkeszthetők bármilyen 3D modellező eszközzel, majd ezek C vagy Objective-C adatokká alakíthatók, így integrálhatók a projektbe.

A játékoknak többféle mód adható meg: csapatoknak szóló vagy egyéni versenyeket célzó, akár versengés nélküli .

Ezek után már kész a is az új játék, indulhat a „kaland”.

Következtetések, továbbfejlesztési lehetőségek

A dolgozat egy olyan keretrendszert mutatott be, amelynek felhasználásával könnyen fejleszthetők kiterjesztett valóságon alapuló on-line kalandjátékok Android és iOS platformokra. Sikerült megtervezni egy olyan szoftverarchitektúrát, amely lehetővé teszi az Internetes játékot, akár nagyszámú csatlakozott kliens esetén is, és könnyen karbantartható, továbbfejleszthető.

Sikerült olyan kliensek kifejlesztése, amelyek a hatékony térképhasználattal és kiterjesztett valóságban való kalandozással izgalmas játékokra nyújtanak lehetőséget.

A továbbfejlesztési lehetőségek között szerepel más platformokon futó kliensek kidolgozása (pl. Windows Mobile), valamint a grafikus felület finomítása. Folyamatban van egy webes alkalmazás létrehozása, amely felhasználóbarát módon adna lehetőséget új játékok létrehozására.

A következtetéseket tekintve nem elhanyagolandó tény, hogy az ilyen típusú játékok mondhatni az „egészséges” kategóriába sorolhatók, hiszen mozgással, kinti tevékenységgel járnak együtt.

A keretrendszer segítségével elkészített első nyilvánosan is elérhető játék tesztelése folyamatban van, remélhetőleg az első felhasználók hamarosan kolozsvári kalandra indulhatnak, játék közben ismerkedve a Kincses Város magyar vonatkozású emlékműveivel, elmúlt idők híres személyiségeivel.

Hivatkozások

1. ***, OpenGL ES - The Standard for Embedded 3D Graphics, <http://www.khronos.org/opengles>
2. ***, (2011-2012)Developer Guide, <https://ar.qualcomm.at/qdevnet>
3. ***, Design Patterns, http://sourcemaking.com/design_patterns
4. Trustin Lee, Ashish Paliwal, (2009), Apache MINA Developer Guide, <http://mina.apache.org/developer-guide.html>
5. Mark Dalrymple, Scott Knaster, (2009), Learn Objective-C on the Mac
6. Bert Altenberg, Alex Clark, Philippe Moughin, (2006), Become An Xcoder
7. ***, iOS Developer Library, <https://developer.apple.com/>
8. Mark L. Murphy, (2008), The Busy Coder's Guide to Android Development
9. ***, Android Dev Guide, <http://developer.android.com>
10. Robie Hanson, General Documentation, <https://github.com/robbiehanson/CocoaAsyncSocket>
11. Damia Ferrer, The Delegation Pattern on iPhone, <http://www.hivestudio.cat>