Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Deáki Soma

ADAPTÍV OKTATÓJÁTÉK FEJLESZTÉSE ANDROID PLATFORMON

Biofeedback alapú keretrendszerre

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2014

Tartalomjegyzék

"Nem a gyereknek van tanulási zavara, hanem az iskolának tanítási zavara."

Dr. Gyarmathy Éva

Összefoglaló

A tanszéken futó AdaptEd projekt keretein belül folyamatosan fejlesztett keretrendszer már számos elkészült játék és oktatószoftver alapjául szolgál. Ez a keretrendszer képes arra, hogy különböző bioszenzorok (mint például EEG vagy EKG) által küldött adatokat feldolgozva következtetéseket vonjon le a felhasználóról, mint például azt, hogy mennyire figyel, vagy hogy mennyire erőltette meg egy adott feladat. A keretrendszerre épülő alkalmazások ezeket az információkat felhasználva személyre szabhatják az általuk nyújtott felhasználói élményt.

Munkám során elkészítettem a Meixner Alapítvány számára készülő alkalmazás egy új verzióját, melyet velük szoros együttműködésben fejlesztek. Az alapítvány többek között különböző részképességzavarral küzdő gyerekeket (mint a diszgráfia vagy a diszkalkulia) egy saját, kidolgozott módszertan segítségével tanít és fejleszt. Az alkalmazás lényege, hogy az ehhez használt típusfeladatokat digitális formában tudják a tanulók megoldani, mivel jelenleg ezek elkészítése renget papírt és emberi munkát igényel.

Az alkalmazás az AdaptEd keretrendszerre épül, az ebből érkező biofeedback adatok segítségével a feladatok nehézsége és a megoldásokért járó jutalom az egyes tanulók igényei szerint finomhangolhatóak. Ezt az alrendszert a legfrissebb neurológiai és pszichológiai kutatásokra építettem. Munkám során ezen kívül új bioszenzorokat is csatolok ehhez a keretrendszerhez, hogy ezek bevonásával teljesebb képet kapjak a felhasználóról. Mind a szenzor, mind az általam megvalósított javaslatgeneráló rendszer alkalmazható további játékok esetén is a keretrendszer adottságainak köszönhetően.

Abstract

The AdaptEd framework, developed by the Department of Automation and Applied Informatics, is already utilized by several games and educational applications. This framework is capable of collection data from different biosensors (such as an EKG or an EEG), and also processes the data, which allows itto infer the mental state of the user, such as their level of attention or the strain caused by the current task. Applications that utilize the framework can use this information to adjust certain parameters and tailor the experience to the particular user.

The application I've been developing was requested by the Meixner Foundation. This foundation's main functions include using special educational methods in teaching and developing children suffering from various partial learning disabilities (such as dysgraphia or dyscalculia). The application allows students to solve special puzzles digitally, which currently require a high amount of paper and human labor to produce.

The application utilizes the AdaptEd framework, using the acquired information to fine-tune the difficulty of and rewards given after each task to the particular student. My work also includes the introduction of new sensor types to this framework in order to be able to get a more complete and saturated picture about the user. Both the sensors introduced and the new reward and difficulty calculation mechanics can be used by any game through the framework.

1 Bevezetés

Mindannyian emlékezhetünk arra, amikor általános iskolában tanév elején megkaptuk új könyveinket. Tantárgyanként általában kiosztásra került egy-egy tankönyv és munkafüzet, melyek sok esetben vaskosak és nehezek voltak. Ezek összeállítása, nyomtatása és kiszállítása rengeteg erőforrást emészt fel, arról nem is beszélve, hogy a hatalmas mennyiségű elhasznált papír milyen megterhelő környezetünk számára. Könyv azonban kell a tanuláshoz, a papír és tinta pedig egészen a közelmúltig nem volt kiváltható mással.

Az utóbbi években egyre szélesebb körök számára nyílik lehetőség arra, hogy a hagyományos nyomtatott médiumok helyett elektronikus eszközöket vegyenek igénybe, mint például táblagépeket vagy ebook olvasókat. Az ezekre írt alkalmazások piacának rohamos növekedésével párhuzamosan egyre nagyobb az igény oktatószoftverekre is, amelyek ki tudják használni a készülékek által kínált lehetőségeket. Az AdaptEd keretrendszer e készülékek számítási kapacitását és a perifériák egyszerű kezelhetőségét kihasználva vizsgálja a felhasználó állapotát különböző bioszenzorok segítségével, ezen adatok alapján pedig visszacsatolást tud biztosítani a rá épülő alkalmazások felé. Ez a keretrendszer lehetőséget biztosít arra, hogy számítógépes játékok emberekre gyakorolt hatását vizsgálhassuk, mely egy nagyon érdekes és sokak által kutatott terület. Saját magunk és mások által készített tanulmányok eredményei hatékonyan felhasználhatóak a visszacsatolás számításához.

Az általam fejlesztett Android alkalmazás a Meixner Alapítvánnyal (1.2. fejezet) szoros együttműködésben készül, az ő megrendelésükre. Lényege, hogy a jelenleg is használt, papír alapú feladatokat átemeljük a digitális világba, hogy ezzel rengeteg papírt és emberi erőforrást spóroljunk. Az alkalmazás az említett AdaptEd keretrendszerre épül, amely a visszacsatolásos tanulás modelljén alapszik[2]. Segítségével a játék adaptívan változtatja paramétereit, ezáltal élvezetesebbé és hatékonyabbá téve mind az órai munkát, mind az önálló gyakorlást a tanulók számára.

1.1 Az AdaptEd keretrendszer

Az AUT tanszéken fejlesztett AdaptEd keretrendszernek két fő feladata van:

- A rá épülő alkalmazásokból és különféle bioszenzorokból érkező adatokat feldolgozni és ezek alapján javaslatot tenni a futó alkalmazások számára jutalmazást és nehézségi szintet illetően.
- Lehetővé tenni, hogy csatlakozó felügyelő kliensek manipulálhassák a keretrendszerre épülő, éppen futó alkalmazásokat.

Fejlesztése 2012-ben kezdődött, azóta pedig egy jól használható keretrendszerré nőtte ki magát. Az általam fejlesztett alkalmazás erősen épít a keretrendszer által tett nehézségi és jutalmazási javaslatokra, melyeket felügyelő alkalmazásból is lehetséges manipulálni. Munkám során illesztettem egy újfajta szenzort a keretrendszerhez, illetve a legfrissebb kutatásokra építve megterveztem, implementáltam és teszteltem több új javaslat számítási algoritmust, mely a felhasználó figyelme, nyugalmi állapota és válaszainak helyessége alapján ajánl nehézségi szintet, illetve kvalitatív és kvantitatív jutalmakat.

1.1.1 Fiziológiai jelek értelmezése

Rengeteg kutatás témáját képzi a különböző fiziológiai jelek és az ember mentális állapota közti kapcsolat keresése. Ilyen jelekből sokféle van, például vizsgálhatjuk az agy működését EEG készüléke használatával [11]. E jelek frekvenciatartománybeli elemzése során bizonyos hullámformákat kell keresnünk, melyek bizonyos eseményeket vagy állapotokat tudnak jelezni. Ilyen hullámforma például a P300, amely alapján az ember döntéshozásra való reakciójáról lehet következtetéseket levonni [13]. A P300 hullámformát például a poligráfiában is vizsgálni szokták a hazugságok detektálásához [12]. Egy másik jó példa ilyen fiziológiai jelre a pupillatágulat [14].

Az ilyen jelek gyűjtése és analízise lehetővé teszi, hogy az alany mentális állapotára következtessünk, mint például koncentráció, mentális erőfeszítés, vagy a nyugodtság, ezekből pedig akár összetett viselkedési tulajdonságok is megállapíthatóak [15]. Az ilyen tulajdonságok alapján a tanulási folyamat mélységére és hatékonyságára is következtethetünk.

A fiziológiai jelek vizsgálatához különböző perifériákra (például EEG készülékre, szívritmus vizsgálóra) van szükség. A keretrendszer jelenleg többféle ilyen készüléket is tud kezelni. Munkám során egy új, egycsatornás EEG berendezést csatlakoztattam, mely egy új eszköz a piacon. Ez a készülék kisebb és olcsóbb az eddig használt EEG-nél, így költséghatékony megoldást nyújt a felhasználók számára.

1.1.2 NeuroSky MindWave

Munkám során illesztettem a keretrendszerhez egy újfajta készüléket, a NeuroSky MindWave Mobile[3] (1.1. ábra) egycsatornás EEG-t. Ez a berendezés bluetooth segítségével kommunikál az Android készülékkel és sokféle adatot ki lehet olvasni belőle.

Míg az eddig használt EEG készülékekből nyers agyhullám adatokat lehetett csak kiolvasni, a MindWave ezek mellett elérhetővé tesz származtatott adatokat is, mint például a felhasználó figyelmének mértéke vagy nyugodtsága. Ezeknek a származtatott adatoknak összevetése az általunk számoltakkal még pontosabbá teszi a levont következtetéseket.

1.1. ábra: NeuroSky MindWave Mobile

1.2 Meixner alapítvány

A Meixner Alapítvány [1] a Dr. Meixner Ildikó által lefektetett különleges módszertan alkalmazásával többek között egy általános iskolát üzemeltet, ahol olyan különböző tanulási képességzavarokkal küzdő gyerekekkel foglalkoznak, mint például a diszlexia, diszgráfia, vagy a diszkalkulia. A módszertan meglehetősen összetett, egy-egy konkrét problémára ad speciális megoldásokat. Én elsősorban a diszgráfiás tanulók számára összeállított tananyaggal foglalkozom, mely jól definiálható típusfeladatokból épül fel. Az általam fejlesztett alkalmazás a jelenleg papír alapú feladatokat hivatott a lehető leghitelesebben szimulálni Androidot futtató táblagépeken. Az alapítvány terve az, hogy az iskolában pályázati pénzből vásárolt táblagépeken az iskolában, továbbá otthon saját készülékeken használhassák a tanulók az alkalmazást. Az alkalmazás korábbi verziójában a típusfeladatok közül elkészült 3 fajta. Munkám során pontosan definiáltam a lehetséges feladattípusokat, az eddig meglévőeket fundamentálisan átalakítottam, implementáltam a többi típust, elkészítettem a feladatsorok megnyitásának és megoldásának folyamatát, valamint a keretrendszer javaslatait integráltam a mindezt megvalósító modellbe.

Néhány példa az alapítvány iskolájában használt tankönyvekből:

1.2. ábra: párosítás

Vágd szét! Rakd ki a mondatokat! Ki mivel foglalkozik? 2+2+2+3/névszó

2.

A reklámcég	a lakodalommal.	A mohamedán	a szigeteléssel.
Az uralkodó	a szabadalommal.	A polgármester	a katalógussal.
A feltaláló	a birodalommal.	A kőműves	a Koránnal.
A vőlegény	a településsel.	A munkáltató	a mosogatással.
Az ügyintéző	a jövedelemmel.	A konyhalány	a határozattal.

1.3. ábra: mondatkészítés

Vágd szét! Rakd ki a szavakat! Az első szótag sárga, a második zöld, a harmadik piros. Tedd a képek alá! 3+2+2

1.

				
				
csin	kon	szar	pör	par
pan	kal	kan	csep	bog
lan	del	dal	csil	gety
ket	fet	tal	pen	lár
la	ló	ka	ta	tyú
ló	la	ti	tő	tyű

1.4. ábra: párosítás

Ezeket a feladatokat jelenleg az alapítvány munkatársai nyomtatják papírra és vágják szét kézzel, illetve adott esetben a szétvágást a tanulókra bízzák. Saját elmondásuk szerint azzal, ha nem kellene a nyomtatással és szétvágással időt tölteni, heti szinten 3-4 órát spórolnának fejenként, ez pedig rengeteg idő.

A feladatok és feladatsorok összeállításának terhe természetesen továbbra is a tanárokra hárul, viszont mivel az Android kliens nem tud Word dokumentumokat bemenetként fogadni (eddig ilyen formátumban készültek a feladatok), készítettem egy webes feladatkészítő alkalmazást. Ennek segítségével a munkatársak kényelmesen és hatékonyan tudnak összeállítani feladatokat és feladatsorokat, melyeket ezután akár teljesen automatikusan lehet az Android alkalmazásba exportálni.

1.2.1 Feladatkészítő alkalmazás

Rövid gondolkodás és a munkatársakkal való egyeztetés után úgy döntöttem, hogy a legmegfelelőbb megoldás a feladatok készítésére egy webes vékonykliens. Így nem szükséges minden, a projektben résztvevő tanár gépére telepíteni új szoftvert, a feladatok bárhonnan elérhetőek, maga az alkalmazás pedig megfelelő felhasználói felületet tud biztosítani ezek kényelmes elkészítéséhez. Az alkalmazás elkészítése során különös figyelmet fordítottam arra, hogy ez a felület maximálisan ergonomikus, a feladatok elkészítésének folyamata pedig gördülékeny és logikusan felépített legyen. Megfelelő jogosultságok birtokában a tanárok által elkészített feladatsorokat könnyedén lehet exportálni az Android alkalmazás számára értelmezhető formában, minden felhasznált erőforrással (például képekkel) együtt.

1.2.2 Meixner játékok Androidon

A Meixner Alapítvány számára készülő Android alkalmazás fejlesztését már előttem elkezdték, végleges verzió azonban nem született belőle. Ennek a kezdetleges alkalmazásnak az alapjait felhasználva az alapítvány munkatársaival való konzultáció után implementáltam a megbeszélt feladattípusokat, a feladatkészítő alkalmazásból exportált formátum beolvasását, teljesen új felhasználói felületet készítettem, valamint integráltam az AdaptEd keretrendszerrel folytatott kommunikációt.

Az alkalmazás eleinte belső terjesztéssel kerül publikálásra, nagyközönség elé egyelőre nem tervezünk állni vele. Bár gyakorlatilag nincs akadálya annak, hogy az

Android kliens automatikusan a feladatkészítő alkalmazást futtató szervertől kérje le a rendelkezésre álló feladatokat, az alapítvány kérésének megfelelően a feladatsorok exportálásának és importálásának folyamata egyelőre manuális lesz. Ez azt jelenti, hogy az alkalmazásban megjelenő feladatsorok statikus erőforrásként lesznek jelen, melyeket szerver alkalmazásból kell exportálni és megfelelően elhelyezni. Terveink szerint egy-egy teljes értékű alkalmazást fogunk exportálni meghatározott kategóriánként, például évfolyamonként és tantárgyanként, ennek konkrétumai még nincsenek rögzítve. Nincs akadálya annak, hogy a jövőben kiépítsünk egy olyan rendszert, amin keresztül a tanárok tudnak közvetlenül a szerveren lévő feladatsorokat tanulókhoz rendelni.

Dolgozatomban először a 2. fejezetben röviden bemutatom a feladatkészítő alkalmazást, majd a 3. fejezetben az Android klienst. Ezek működésének ismeretében a 4. fejezetben részletezem kutatásaim legfontosabb lépéseit és eredményeit, itt bemutatom az új szenzor csatolását, az új, adaptív nehézség- és jutalomszámítási algoritmust, továbbá ezek finomhangolását és használatát a Meixner Játékok alkalmazásban.

2 Meixner Authoring Tool

2.1 Felhasznált technológiák

A feladatkészítő alkalmazás egy webes vékonykliens. A szerveren egy php MVC alkalmazás fut, amely egy MySQL adatbázisból nyeri az adatokat. MVC php keretrendszert nem használok, mindent saját magam írtam. A kliens oldal HTML5 és CSS3 alapú, jQuery-t [5], a jQuery UI [6] könyvtár néhány elemét és a Bootstrap library-t [4] használtam fel hozzá. Utóbbi kettő esetén különösen körültekintően jártam el, hogy semmilyen ütközés ne legyen a könyvtárak között.

2.2 Tervezés

Az alkalmazás két fontos, alapvető fogalmat használ:

- Feladat: egy alkalmazásbeli feladat megfelel egy hagyományos, papír alapú feladatnak. Minden feladatnak van típusa, ami meghatározza, hogy milyen elemekből épül fel és mi a felhasználó célja megoldáskor. Ezen kívül a feladatokhoz tartozik ajánlott évfolyam, tantárgy, témakör és nehézség. A nehézség az alapítvány munkatársainak kérésének megfelelően bináris: egy feladat vagy könnyű, vagy nehéz.
- Feladatsor: a feladatsorok feladatokból állnak. Minden feladatsorhoz tartozik ajánlott évfolyam, tantárgy, témakör és hossz. Az Android kliensben ezeket a feladatsorokat lehet elindítani, melyekből a beállított hossznak megfelelően és a framework javaslatait figyelembe véve véletlenszerűen választunk feladatot. Bár mind a feladatok, mind a feladatsorok tartalmaznak információt az évfolyamra, tantárgyra és témakörre vonatkozóan, ezek közül csak a feladatsor által tárolt adatok mérvadóak. A feladatokhoz tartozó adat pusztán a felhasználói felület átláthatóságát segíti elő, valamint lehetőséget ad jelentősen részletesebb szűrésre és rendezésre. A tanárok így gyorsan és könnyedén meg tudják találni az általuk éppen keresett feladatot.

Az első fontos tervezési lépés az implementálandó feladattípusok meghatározása volt. Ezt több alkalommal is egyeztettük az alkalmazást használó tanárokkal és végül a következő típusokat határoztam meg:

- **Párosítás**: két vagy több elemből álló tuple-ök összerendelése. Például egy szó párosítása szinonimáival.
- **Csoportosítás**: elemek csoportosítása valamilyen szempont alapján. Például zöldségek és gyümölcsök szétválogatása.
- **Sorrendezés**: elemek helyes sorrendjének felállítása. Például történelmi események időbeli sorrendjének megállapítása.
- **Mondatkészítés**: mondatok összerakása részekből. Hasonlít a párosítás típushoz, azzal a különbséggel, hogy itt mondatokká kell az elemeket összerendezni.
- **Mondat kiegészítés**: hiányos mondat kipótlása rövid szavakkal vagy szövegrészletekkel.
- **Igaz/hamis**.
- **Memóriajáték**.
- **Párosítás és sorrendezés**: néhány elemből álló ennesek összerendelése, majd ezek helyes sorrendezése.
- **Csoportosítás és sorrendezés**: elemeket csoportokba kell rendezni, a csoporton belül pedig helyes sorrendet kell felállítani.
- **Mondatkészítés és sorrendezés**: Mondatok összeállítása darabjaiból, majd az összerakott mondatok sorrendezése.
- **Mondatkészítés és csoportosítás**: Mondatok összeállítása darabjaiból, majd az összerakott mondatok csoportosítása.
- **Mondat kiegészítés és sorrendezés**: Hiányos mondatok kiegészítése, majd a kiegészített mondatok sorrendezése.
- **Mondat kiegészítés és csoportosítás**: Hiányos mondatok kiegészítése, majd a kiegészített mondatok csoportosítása

Ezzel az összes jelenleg megbeszélt típusfeladatot lefedtem. Későbbiekben előfordulhat, hogy szükség lesz új típusokra is, ezek integrációját a jól megtervezett struktúra hatékonyan lehetővé teszi. Ehhez fel kell vinni egy új bejegyzést az adatbázis megfelelő táblájába, majd az alkalmazás megjelenítésért felelős részében meg kell adni

az új típushoz tartozó HTML kódot és megkötéseket. A php alkalmazás MVC jellegének köszönhetően nem szükséges a felhasználói felület elkészítéséért felelős kódon kívül egyebet módosítani.

Az eddigieknek megfelelően meghatároztam a feladatokat alkotó elemi egységeket:

- Szöveg
- Kép
- Csoport
- Hiányos mondat

Ezekből az elemekből bármelyik feladattípus felépíthető. A szöveges és képi elemek nem szorulnak különösebb magyarázatra, a másik kettő viszont valamivel összetettebb. Egy csoport sorrendezve tartalmaz akárhány más elemet (alapvetően akár több csoportot is). A hiányos mondatok tartalmaznak felhasználható mondatrészeket, melyek nem jelennek meg külön atomi elemként, mivel kizárólag ilyen módon kerülnek használatra. Természetesen feladattípustól függ, hogy ezen elemek milyen elrendezése számít legálisnak, az erre vonatkozó megkötéseket típusonként meg kell adnunk. Egy párosítás feladatban például egyáltalán nem szerepelhet hiányos mondat, egy sorrendezés feladat esetén pedig nincs értelme több szintű csoport tartalmazást kialakítani.

Az elkészített feladatot mentéskor még kliens oldalon XML formátumba alakítjuk, melyet parse-olás nélkül tárol a szerver. Úgy döntöttem, hogy felesleges a feladatok szerkezetét adatbázis szinten is implementálni, mivel szerver csak az ezekhez tartozó metaadatokkal végez műveleteket. Ilyen módon az exportálás is könnyebb, mivel a feladatot leíró XML-hez csak ezeket a metaadatokat kell csatolni. Az egyetlen eset, amikor az elmentett XML formátumú adatot kénytelenek vagyunk parse-olni, az a feladat betöltése, ilyenkor ez alapján generáljuk a feladatot reprezentáló HTML kódot. Az egyes elemekhez tartozó XML kódok:

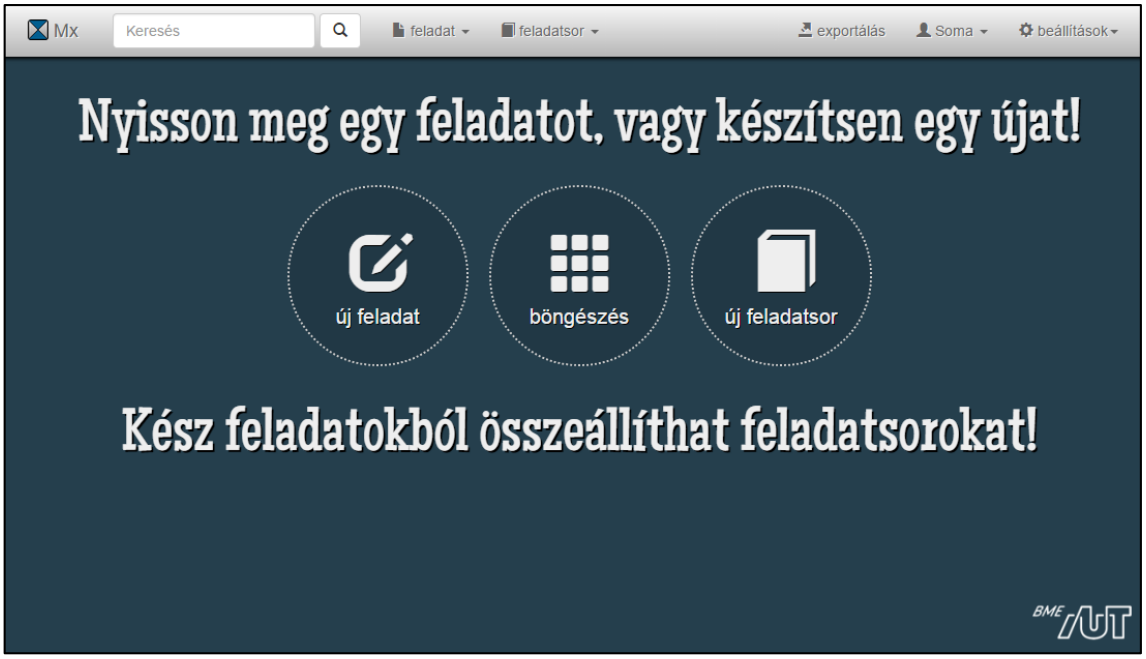
- Szó: `<word>tartalom</word>`
- Kép: `<image>fájlnév</image>`
- Csoport: `<group title=”cím”>tartalom</group>`

- Hiányos mondat:

```
<sentence text=”mondat szövege”>
     <part number=”sorszám”>szöveg</part>
</sentence>
```

2.3 Felhasználói felület

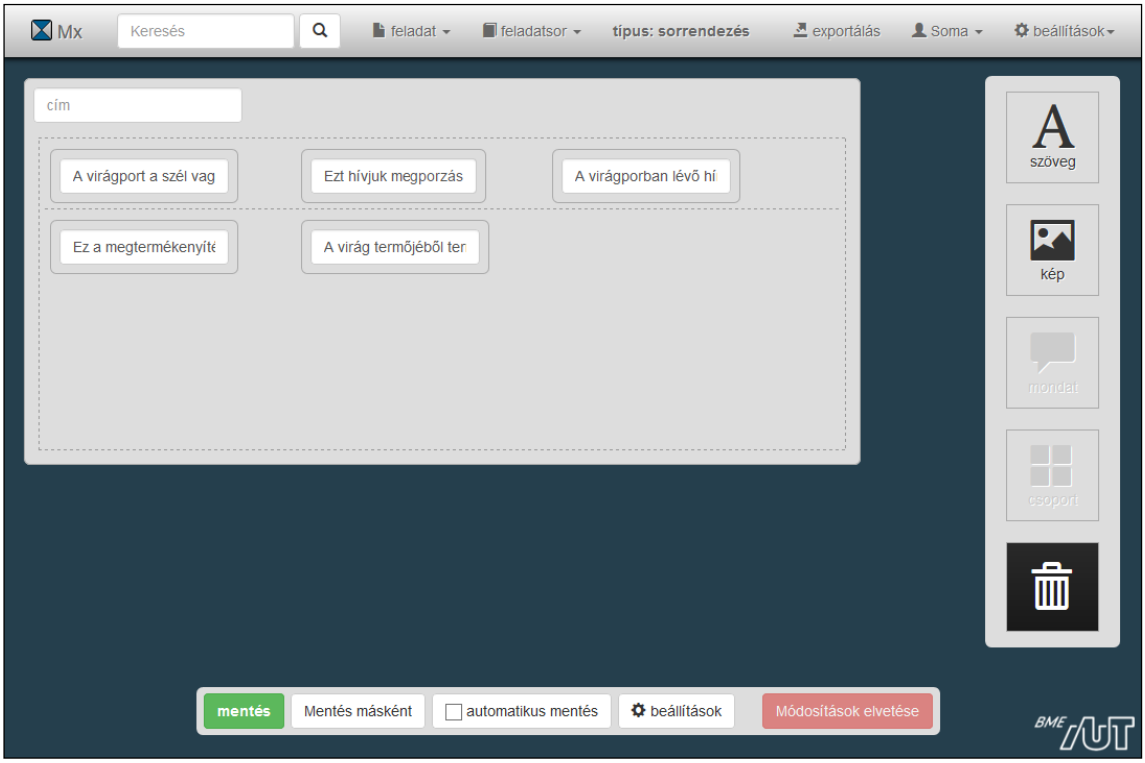
A felhasználói felületet igyekeztem a lehető legergonomikusabbra tervezni. A nyitóoldalt mutatja a 2.1. ábra.



2.1. ábra: az alkalmazás nyitóoldala

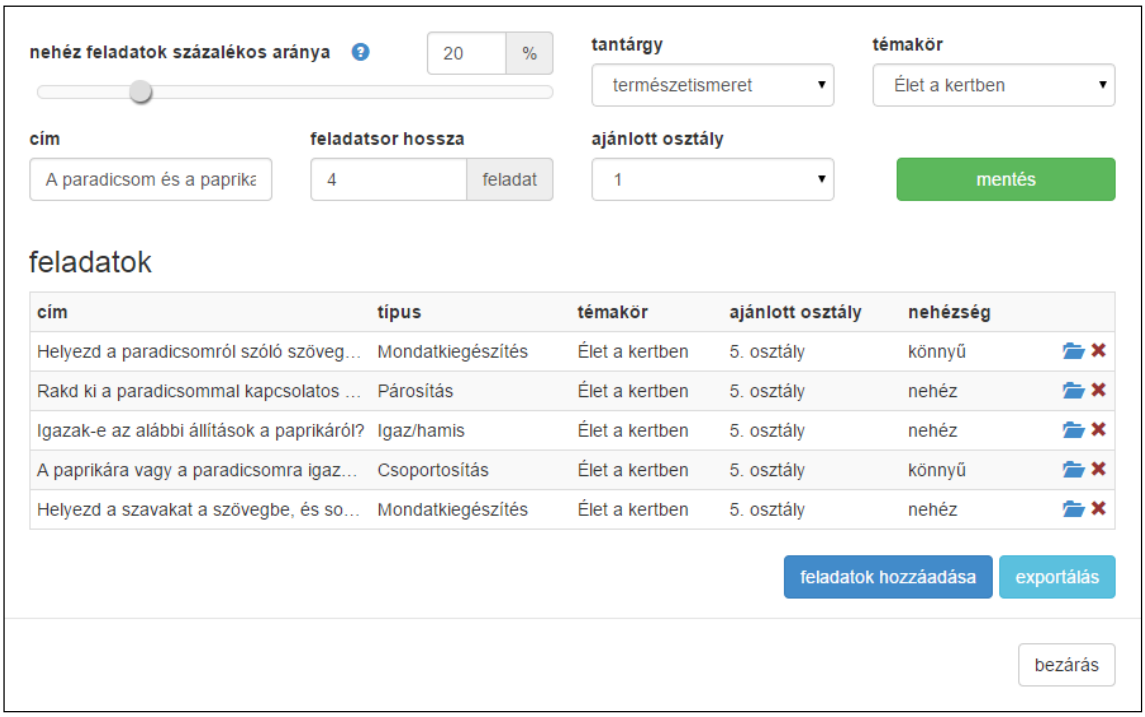
Új feladat készítése vagy meglévő megnyitása után láthatóvá válik a képernyő jobb oldalán egy paletta, amelyről drag&drop módon elemeket lehet elhelyezni a felületen. Ezekből kell az adott feladat struktúráját kialakítani és feltölteni azt tartalommal. Az alkalmazás mindvégig figyel rá, hogy érvénytelen struktúra kialakítása ne legyen lehetséges. Erre mutat példát a 2.2. ábra. A képen egy sorrendezéses feladat látható. Bal oldalt található az említett paletta, melyről ilyen feladattípus esetén csak szöveges és képes elemek húzhatóak a felületre. Ezek egy előre elnyelezett csoportban helyezhetőek el. A feladat elkészítésekor az elemeket helyes sorrendben kell megadni, ezek összekeverése az Android kliensben fog megtörténni. Ha egy felvitt elemet törölni szeretnénk, akkor a paletta alján látható kuka felé kell húzni.

A képernyő alján látható menü segítségével lehet mentéssel kapcsolatos műveleteket végezni és a megnyitott feladat metaadatait szerkeszteni.



2.2. ábra: megnyitott feladat szerkesztése

A böngésző ablak segítségével az elkészült feladatokat hozzáadhatjuk feladatsorokhoz. Egy megnyitott feladatsort mutat a 2.3. ábra.



2.3. ábra: megnyitott feladatsor

Minden feladatsorhoz megadható egy nehézségi szint, mely azt jelenti, hogy a feladatok közül milyen arányban kerüljenek kiválasztásra nehéz feladatok. Ennek a metaadatnak nincs jelentősége akkor, amikor a framework javaslataira figyelünk. Ha egy készüléken nem elérhető a keretrendszer, akkor ezt az értéket használjuk a soron következő feladat kiválasztásához. A feladatsor hossza azt jelenti, hogy annak megnyitása után hány feladatot kapjon a felhasználó. Így egy témakörhöz tartozó feladatsor tartalmazhat rengeteg feladatot, a felhasználónak azonban nem kell az összeset megoldania minden alkalommal.

2.4 Exportálás

Admin felhasználóknak lehetőségük van feladatsorokat exportálni egy-egy zip fájlként. Ez a fájl tartalmazza a feladatsort leíró XML fájl és minden kép felhasznált kép erőforrást. Az XML formátuma:

```
<set title="A paradicsom és a paprika"
     subject="environmental_studies"
     topic="garden_life"
     grade="5"
     length="4"
     difficulty="20">

     <puzzle title=”Igazak-e az alábbi állítások a paprikáról?”
             type=”true_or_false”
             difficulty=”2”>

          <![CDATA[ <!-- puzzle tartalma --> ]]>

     </puzzle>

     <!-- többi feladat -->

</set>
```

A feladatok tartalmát CDATA, azaz *character data* formájában írom csak bele az XML-be. Erre azért van szükség, mert ezek a fájok két lépésben kerülnek feldolgozásra. Az Android kliens indításakor nincs szükség a feladatok beolvasására, csak a feladatsorokat szeretnénk listázni, az egyes feladatok tartalmát leíró XML-t pedig nyers formában tároljuk. Ezeket csak akkor parse-oljuk, amikor az adott feladatot megnyitjuk.

3 Android alkalmazás

A Meixner Alapítvány megbízásából készülő Android alkalmazást Fekete András kezdte el fejleszteni az elmúlt években. Egy kezdetleges verzió el is készült, ez azonban alkalmatlan volt arra, hogy az alapítvány iskolájában használatba is kerüljön. 2014 elején kezdtem el a projekttel foglalkozni, azóta ezt az alkalmazást alapjaiban átírtam és rengeteg új funkcióval egészítettem ki. Az alkalmazás felhasználói felületének tervezésében Kocsi Olga, a MoME kutatója segít.

3.1 Technológiai háttér

Az alkalmazás az AdaptEd framework-ön kívül nem használ semmilyen külső könyvtárat. Futtatásához Android 4.0.3-ra (API level 15) van szükség. Az alapítvánnyal folytatott megbeszélések értelmében az alkalmazás célzottan 10”-os táblagépekre készül, egyéb képernyőméretekkel nem foglalkozom, ugyanis a legtöbb feladattípus egyszerűen nem jeleníthető meg kisebb kijelzőn úgy, hogy az kényelmes legyen az esetleg finommotorikai nehézségekkel küszködő gyermekek számára is. A cél az, hogy a papír alapú feladatokat a legélethűbben utánozzuk.

3.2 Tervezés

Első lépésként meg kellett határozni az alkalmazás használatának folyamatát:

1. Az alkalmazás megnyitása és az üdvözlő képernyő után a felhasználó a feladatsorok listájára kerül.
2. A felhasználó kiválasztja azt a feladatsort, amelyet meg szeretne oldani.
3. Elindul a feladatsor, betöltődik az első feladat.
4. A felhasználó elkezdi megoldani a feladatot. Ennek menete az aktuális feladat típusától függ.
5. Amikor a felhasználó úgy érzi, hogy készen van, megnyomja az ellenőrző gombot. Ilyenkor két lehetőség van:

a. A megoldás helyes, ki kell rajzolni egy dialógust az aktuális eredménnyel (megoldás ideje, próbálkozások száma, stb.). A dialógus két opciót kínál fel:

   i. Következő feladat betöltése.

   ii. Az aktuális feladat megtekintése. A felhasználó így jobban megjegyezheti, mi volt a helyes megoldás. Ilyenkor az ellenőrzés gomb ismételt megnyomására töltődik be a következő feladat.

b. A megoldás helytelen. Addig nem mehet tovább a következő feladatra a felhasználó, amíg nem ad helyes megoldást. Ilyenkor vizuális visszajelzést kell kirajzolni, amely alapján az aktuális megoldás javítható.

6. Ezen a ponton két lehetőség van:

   a. Amennyiben a feladatsornak nem értünk a végére, választani kell egyet a hátralévő feladatok közül és betölteni azt.

   b. Ha a feladatsor végére értünk, át kell navigálni az eredményeket megjelenítő oldalra. Innen a felhasználó visszatérhet a feladatsorok listájára és az 1-es ponttól folytathatja.

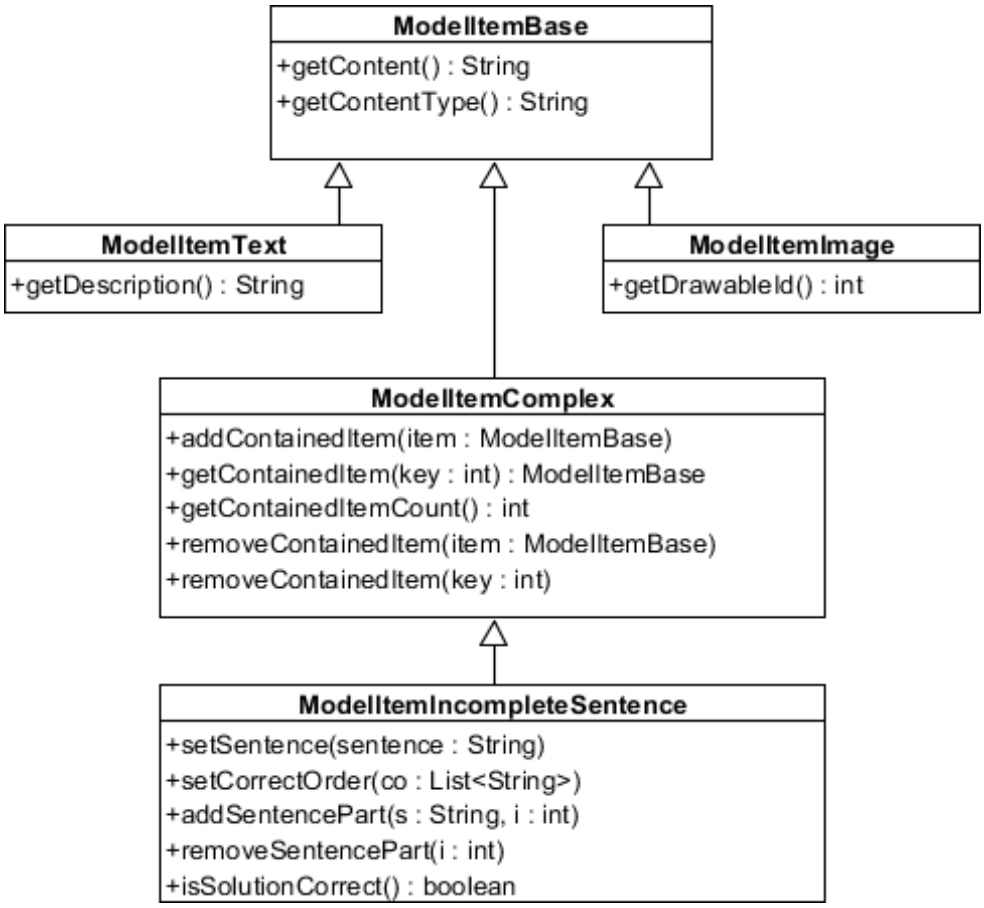
Az implementálandó feladattípusokat bővebben a 2.2 fejezetben fejtettem ki. Az Android alkalmazásban minden feladattípus működését egyenként implementálni kell. A feladatok ebben az esetben is az említett fejezetben taglalt elemi egységekből épülnek fel, ezért ezeket nagyon fontos újrahasznosíthatóra tervezni. A konkrét vizuális reprezentációja ezen elemeknek az adott feladattípustól is függ, a mögöttes modell elemek viszont alapvetően megegyeznek minden esetben.

3.2.1 Modell elemek

A feladatok logikája modell elemekből épül fel. A feladatok darabjainak közös absztrakt ősosztálya a `ModelItemBase`, az egyes elemi egységeket reprezentáló modell osztályok ebből származnak:

- `ModelItemText`: szöveges elemek

- `ModelItemImage`: kép elemek
- `ModelItemComplex`: csoportok
- `ModelItemIncompleteSentence`: hiányos mondatok



3.1. ábra: modell osztályok hierarchiája

Ezen osztályok hierarchiáját és pontosabb leírásár mutatja a 3.1. ábra.

A csoportokat megvalósító `ModelItemComplex` tárolhat akárhány `ModelItemBase` leszármazottat. A hiányos mondatokat illetően itt egy apró különbség van a feladatkészítő alkalmazáshoz képest, ugyanis a `ModelItemIncompleteSentence` a csoportokat jelképező `ModelItemComplex` leszármazottja. Ez a kapcsolat logikus az Android alkalmazás esetén, elvégre a hiányos mondat tulajdonképpen a rendezett csoport egy speciális esete. Ezt a fajta kapcsolatot nem lehetett volna a feladatkészítő alkalmazásban kényelmesen megtenni, mivel elkészítésük élesen eltér egymástól.

3.2.2 Feladatok

Minden feladattípushoz tartozik:

- Egy GameModel
- Egy ViewManager
- Egy vagy több Fragment
- Opcionálisan egyéb kiegészítő osztályok

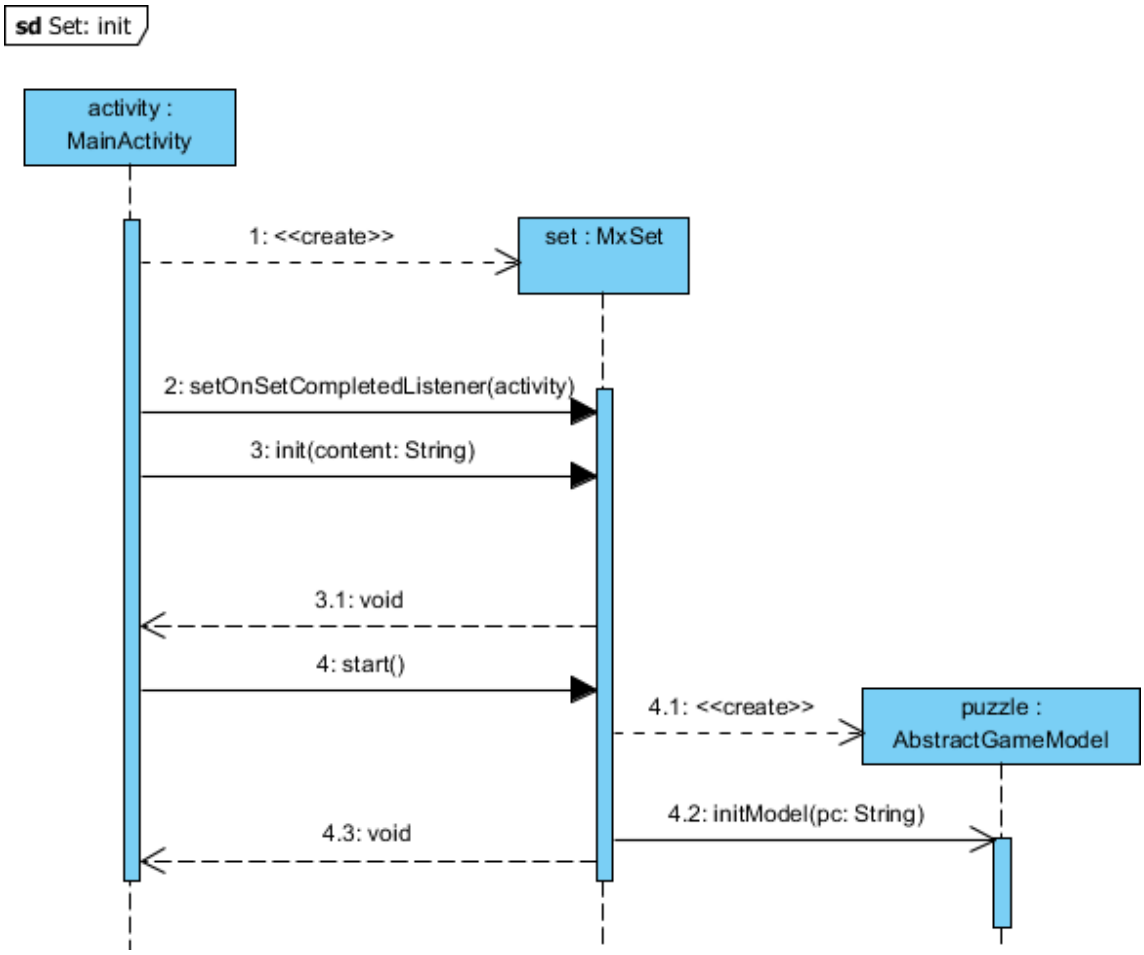
A GameModel osztályok felelősek az adott feladattípus logikai felépítéséért és működtetéséért, ősosztályuk az `AbstractGameModel`. Ezen keresztül lehet betölteni, elindítani a feladatot, manipulálni a benne szereplő modell elemeket, és ellenőrizni a megoldás helyességét.

A ViewManager osztályok az `AbstractViewManager` leszármazottai. Ahogy az nevükből is kikövetkeztethető, ezek az osztályok felelősek a felhasználói felület kezeléséért. A ViewManager határozza meg, hogy elemek összekötésénél és szétválasztásánál mi történjen, bizonyos eseményekre hogyan kell válaszolni, továbbá lehetővé teszi a modell elemek drag&drop mozgatását.

A feladattípushoz tartozó fragment-ek közvetlenül tudják kezelni a felhasználói felületet. Eseményeket továbbítanak a ViewManager felé és annak utasításai alapján manipulálják a kirajzolt elemeket.

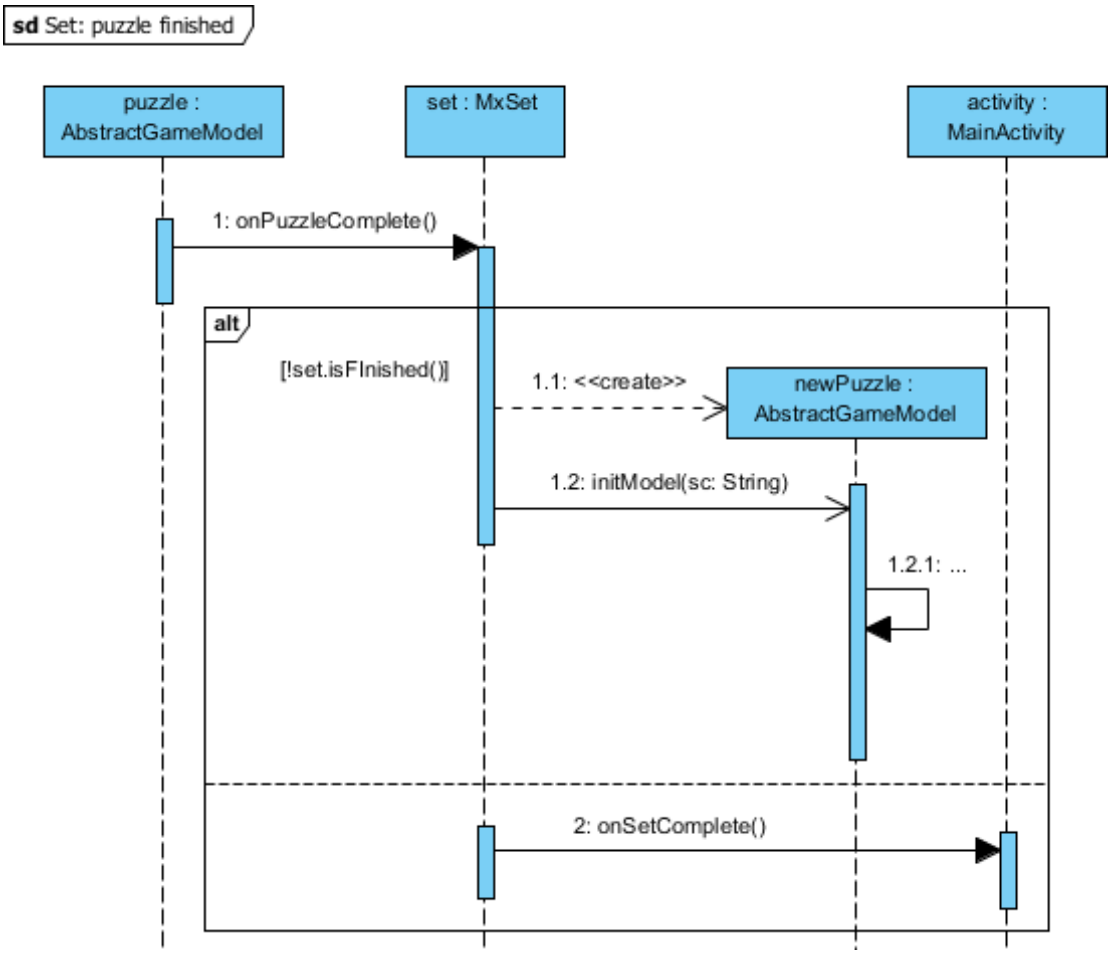
3.2.3 Feladatsorok

Az `MxSet` osztály feladatsorokat reprezentál. Feladatsort tartalmazó XML állománnyal lehet inicializálni, az ebből kiolvasott feladatok tartalmát eleinte nyers formában tárolja. A feladatsorok elindításának folyamatát a 3.2. ábra mutatja be.



3.2. ábra: feladatsor indítása

Indítás után (tehát az `MxSet.start()` hívásának hatására) létrejön egy modell a választott feladat típusának megfelelően. Ezt az XML-ből olvasott tartalommal inicializáljuk aszinkron módon, mivel ennek során több viszonylag erőforrás igényes műveletet is végrehajtunk (például Fragment-ek készítése, View-k felfújása). Ennek lefutása után a feladathoz tartozó ViewManager és Fragment-ek veszik át az irányítást és csak akkor szólnak ismét a feladatsorhoz, ha a felhasználó jó megoldást vitt be. Ennek folyamatá mutatja meg a 3.3. ábra. A feladat a `PuzzleActionListener` interface-en keresztül szól a feladatsornak, hogy a felhasználó elkészült. Amennyiben a feladatsornak nem értünk még a végére, az inicializáláskor látott móddal megegyezően egy új feladatot kell betölteni. Ha vége a feladatsornak, az `OnSetCompleteListener` megfelelő hívásán keresztül értesítjük erről az activity-t.



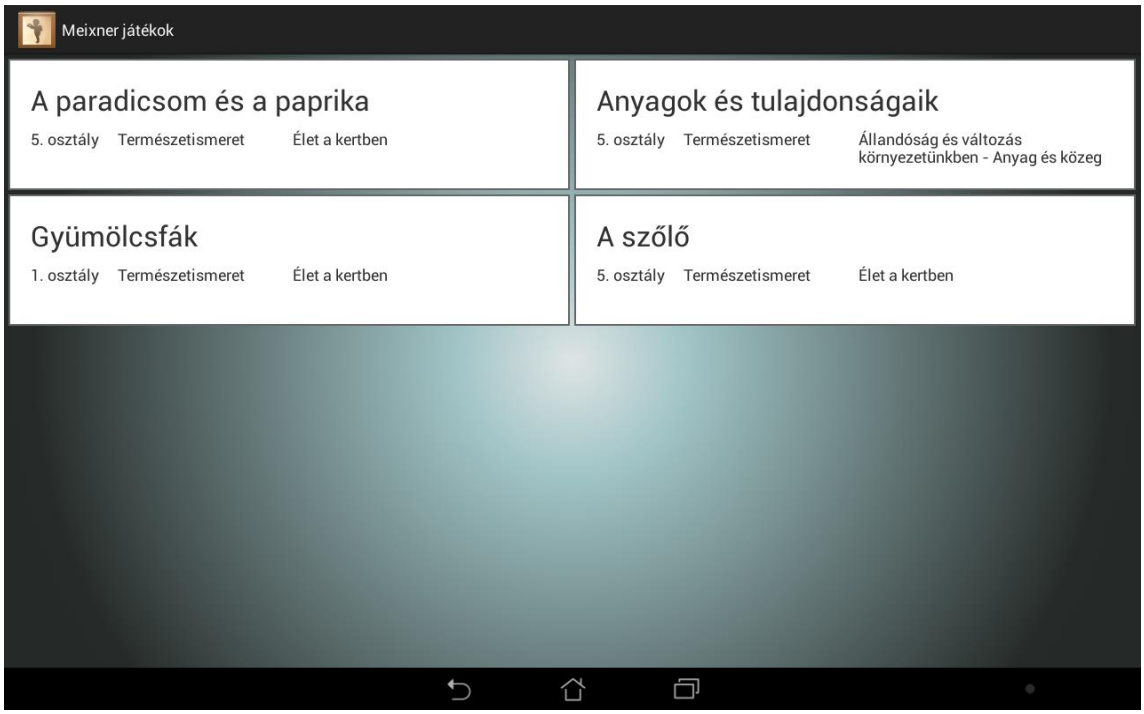
3.3. ábra: feladat befejezése

3.3 Felhasználói felület

Az alkalmazás felületét egy ViewPager kezeli. Ez tartalmaz egy üdvözlő képernyőt, a feladatsorok listáját, illetve minden egyes feladattípushoz egy dedikált oldalt. Az üdvözlő képernyőt mutatja a 3.4. ábra.

A start gomb megnyomása után a feladatsorok listája látható, a felhasználó ezek közül kiválaszthatja, melyiket szeretné megoldani. A Meixner Alapítvány munkatársai folyamatosan töltenek fel feladatokat, ezek közül néhányat importáltam az Android alkalmazásba, hogy legyen benne tesztadat. Az összes feladat elkészítése viszonylag nagy egyszeri munka a tanárok számára, ebben a pillanatban még nem végeztek vele. Az említett listát demonstrálja a 3.5. ábra.

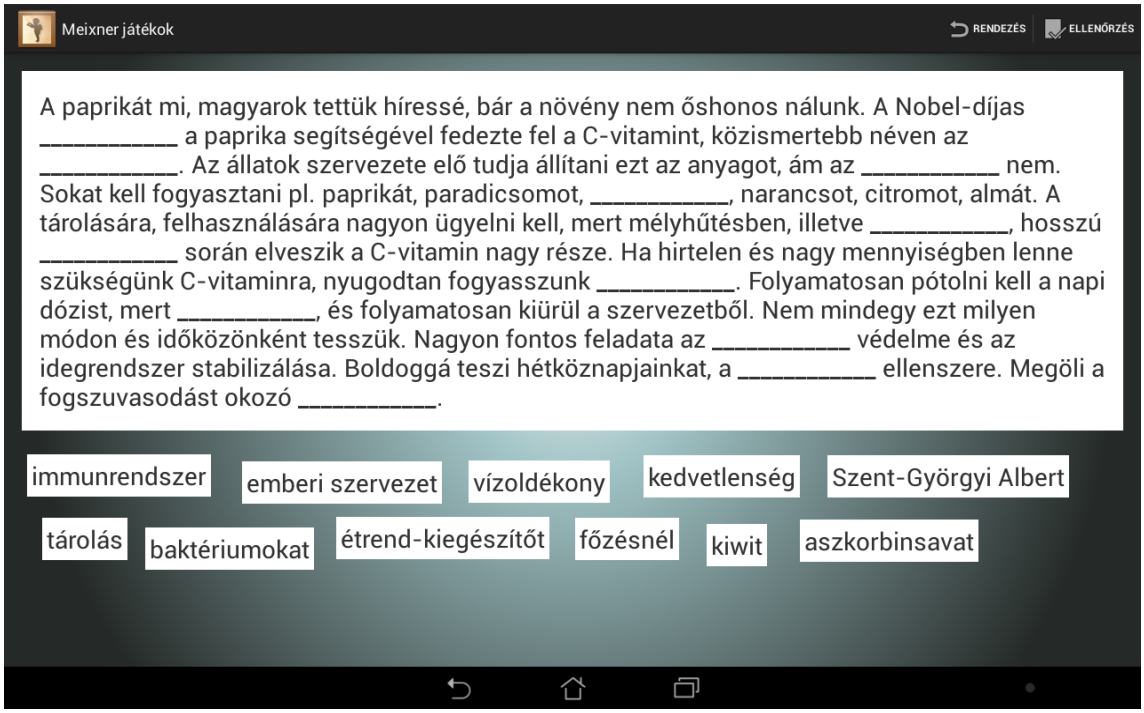
3.4. ábra: üdvözlő képernyő



3.5. ábra: feladatsorok listája

Egy feladatsor kiválasztása után betöltődik az első feladat. Ez a javasolt nehézségi szint alapján történik, mely egy [0; 100] intervallumon mozgó egész szám. Jelentése, hogy a következő feladat választásakor az esetek milyen százalékában essen a választás nehéz feladatra. Amennyiben egy feladatsorban csak nehéz vagy könnyű feladatok vannak, a választás teljesen véletlenszerű. Ennek a számnak az értékét tudja a

keretrendszer a mért adatok alapján befolyásolni javaslataival. A 3.6. ábra egy példát mutat a mondat kiegészítés típusú feladatra.



3.6. ábra: mondat kiegészítés feladat

A szavakat drag&drop módon lehet elhelyezni a mondatban. Az ellenőrzés gomb lenyomására visszajelzést kap a felhasználó az eddig behelyettesített szavak helyességéről.

4 Az AdatpED keretrendszer

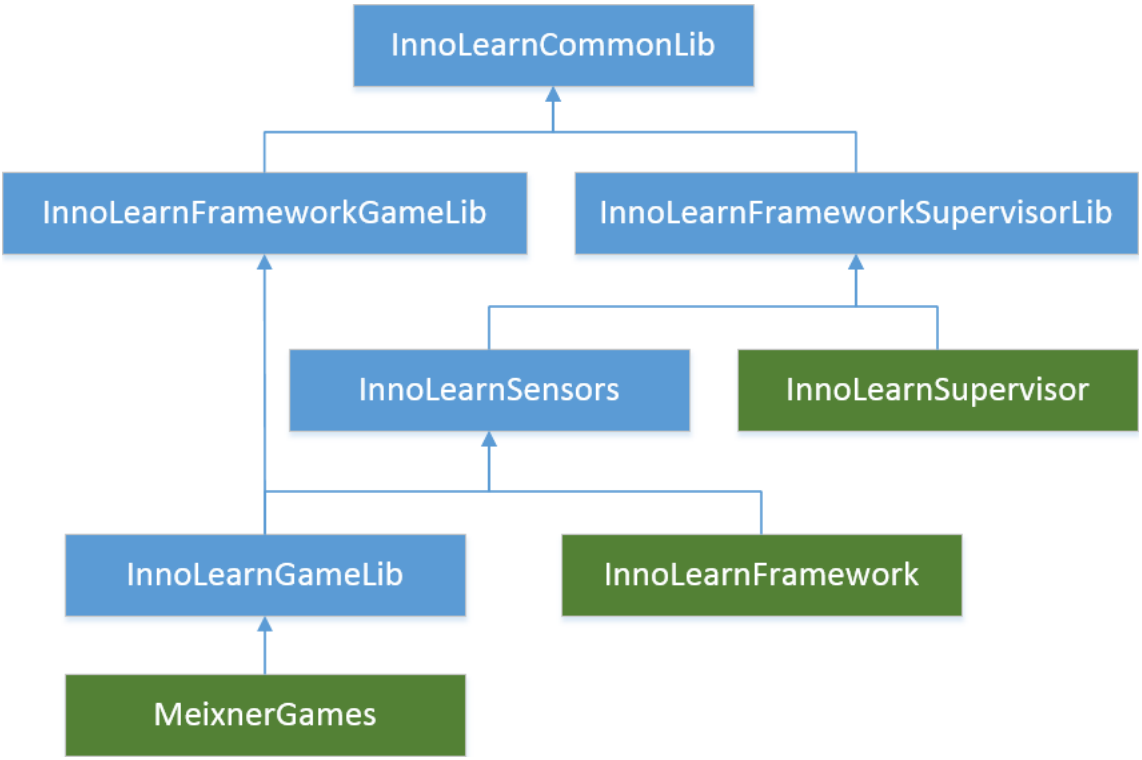
Az AdaptEd keretrendszert Angeli Róbert és Fekete András készítette, Szita Ádám pedig felügyelő alkalmazást írt hozzá. Feladata, hogy különböző bioszenzorok segítségével adatokat gyűjtsön a felhasználóról, majd ezen adatok analízise alapján különböző következtetéseket vonjon le és javaslatot tegyen jutalmazást és a kliens alkalmazás nehézségét illetően. Munkám során csatoltam egy új, egycsatornás EEG készüléket a keretrendszerhez. Ennek előnye, hogy viszonylag olcsó, könnyen felhelyezhető, és a nyers adatok mellett származtatott adatokkal is tud szolgálni. A belőle olvasott adatok alapján következtethetünk többek között a felhasználó figyelmi szintjére és nyugodtságára.

4.1 Felépítés

A keretrendszer a következő elkülöníthető részből áll:

- **Felügyelő alkalmazás**: Ennek segítségével a tanteremben jelen lévő pedagógus közvetlenül manipulálhatja a tanulók készülékein futó játékokat. Ez egy különálló alkalmazás, amely a tanárnál lévő készüléken fut és helyi hálózaton keresztül kapcsolódik a többi táblagéphez. A játékok szempontjából tulajdonképpen sok szempontból úgy viselkedik, mint egy mérvadó szenzor, mivel a keretrendszer által tett javaslatokat lehetséges vele befolyásolni.
- **Keretrendszer**: Ez a komponens felelős az adatok fogadásáért a többi rendszer és a szenzorok felől. Az adatokat feldolgozza, a hozzá kapcsolódó játékokat pedig értesíti az új javaslatokról. Bizonyos időközönként az akkumulált adatokat feltölti egy szerverre.
- **Játékok**: Olyan oktatójátékok, melyek képesek eseményeket küldeni a keretrendszer felé, illetve annak visszacsatolására képesek reagálni.
- **Szerver**: A keretrendszer erre a szerverre tölti fel bizonyos időközönként a mérési adatokat, ezzel lehetővé téve azok utólagos analízisét, melynek köszönhetően hosszú távon új módszereket fejleszthetünk ki keretrendszer által olvasott adatok értelmezésére.

Az egymásra épülő könyvtárak és alkalmazások hierarchiáját mutatja a 4.1. ábra.



4.1. ábra: a keretrendszer könyvtárai és alkalmazásai

A kékkel jelölt elemek könyvtárak, a zöldek pedig futtatható alkalmazások. Ahhoz, hogy egy alkalmazás igénybe tudja venni a keretrendszer szolgáltatásait, futnia kell az adott készüléken az InnoLearnFramework-nek. A projekt neve eleinte InnoLearn volt, amit később AdaptEd-re változtattunk. Egyelőre nem történt meg az összes erőforrás refaktorálása, ezért szerepel még néhány helyen az InnoLearn név. A könyvtárak rövid magyarázata:

- **InnoLearnCommonLib**: olyan alapvető osztályokat tartalmaz, amik mind a keretrendszerben, mint a felügyelő rendszerben, mind a játékokban szükségesek.
- **InnoLearnFrameworkGameLib**: Olyan osztályokat tartalmaz, amelyekre szüksége van a játékoknak és a keretrendszernek is.
- **InnoLearnFrameworkSupervisorLib**: A felügyelő alkalmazás és a keretrendszer által egyaránt felhasznált osztályokat tartalmazza.

- **InnoLearnSensors**: Szenzor perifériák csatolásáért és az azok által küldött jelek feldolgozásáért, valamint megfelelő formátumba való konvertálásáért felelős. A jelfeldolgozó algoritmusok egy része C++-ban íródott, ezért szükséges hozzá az Android NDK [7].
- **InnoLearnGameLib**: A keretrendszer szolgáltatásait a rá épülő játékok ezen a könyvtáron keresztül érhetik el.
- **InnoLearnSupervisor:** Futtatható felügyelő alkalmazás.
- **InnoLearnFramework:** Futtatható keretrendszer alkalmazás. Saját felhasználói felülettel nem rendelkezik, egy háttérben futó szolgáltatást tartalmaz.
- **MeixnerGames:** A Meixner Alapítvány számára készülő alkalmazás.

4.2 Kommunikáció

4.2.1 Játékoktól a keretrendszer felé

Minden, a keretrendszerre épülő játék az InnoLearnGameLib könyvtárra épül. Ahhoz, hogy a játék kommunikálni tudjon a keretrendszerrel, a GameBase nevű osztályt kell igénybe venni.

A GameBase az Activity-ből származik, minden olyan saját Activity-nek ősosztálya, amely üzeneteket küld a keretrendszer felé és javaslatokat fogad tőle. Az ehhez használt metódus:

```
public void sendGameEvent(GameEvent gameEvent)
```

Mint azt a metódus szignatúrájában látni lehet, játékbeli eseményeket tudunk továbbítani egy-egy GameEvent objektum formájában a keretrendszer felé. A különböző eseményekhez egy-egy dedikált osztály kell készítenünk, melyek a GameEvent-ből származnak.

A Meixner Játékok alkalmazás esetén két fontos eseményt határoztam meg, ezek a következők:

- **Feladat megoldása során az első ellenőrzés**:

  Ez az esemény akkor sül el, amikor a felhasználó először nyom rá a feladat megoldása során az ellenőrzés gombra, tehát először gondolja úgy, hogy helyesen oldotta meg a feladatot. Ebben az esetben a keretrendszer felé továbbított adat a helyesen megoldott feladatrészek százalékos aránya lebegőpontos számként a [0;1] intervallumból. Létrehoztam a `PuzzleFirstAttempEvent` osztályt:

```
@GameEventType(reward = PuzzleReward.class)
public class PuzzleFirstAttemptEvent extends GameEvent {

     @ParameterValue(minValue="0.0f", maxValue="1.0f")
     float mHitRatio;

     //...
}
```

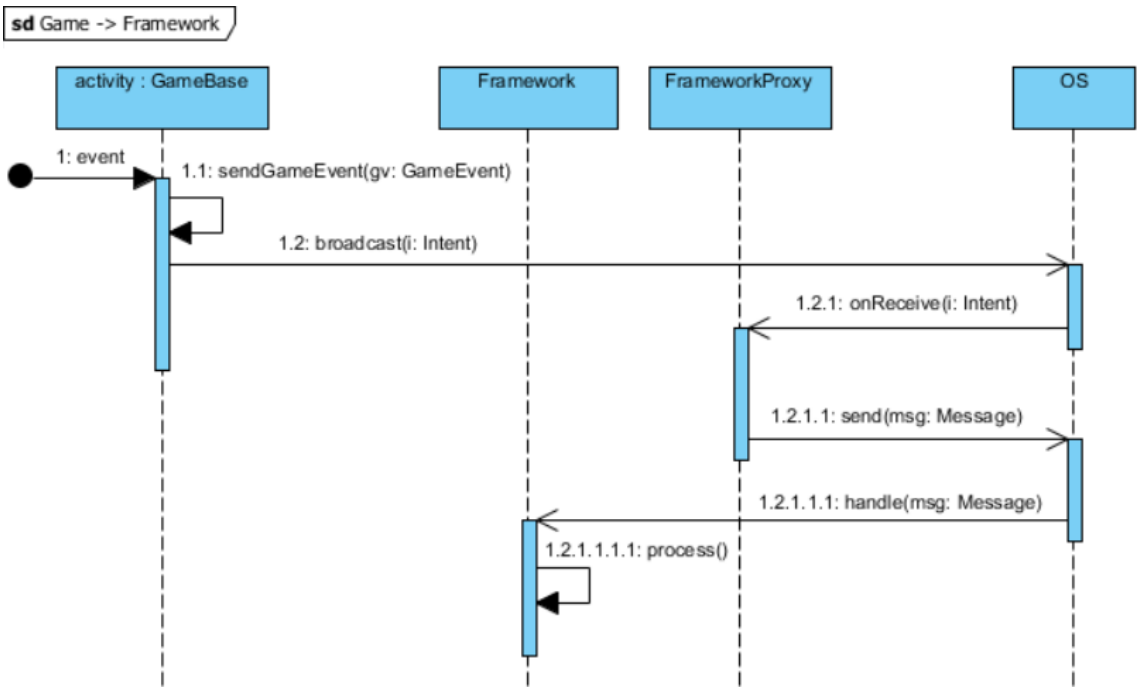
- **Feladat befejezése**:

  Ez az esemény akkor sül el, amikor a felhasználó helyes megoldást adott a feladatra. Ebben az esetben egy jó mutatója a felhasználó sikerességének az, hogy hányadik próbálkozásra született meg a helyes megoldás. Létrehoztam a `PuzzleCompletedEvent` osztályt:

```
@GameEventType(reward = PuzzleReward.class)
public class PuzzleCompleteEvent extends GameEvent {

     public static final int MAX_ATTEMPTS = 10;

     @ParameterValue(minValue = "0", maxValue = MAX_ATTEMPTS+"")
     int mAttemptCount = 0;

     //...
}
```

A GameEvent osztály leszármazottai tetszőleges tagváltozókat tartalmazhatnak, amelyekkel leírható az adott esemény. Ebben az esetben elég volt eseményenként 1-1 értéket tárolnom, de több is lehetne akár. Az esemény küldésekor az esemény objektumot a GameBase továbbítja a Framework felé ezen belül a FrameworkProxy képes GameEvent-ek fogadására. Ez a megfelelően annotált tagváltozók alapján számol egy

egyetemes értéket, az úgynevezett *goodnessValue*-t, mely a [0,1] intervallumon helyezkedik el. Ez az érték a megadott `minValue` és `maxValue` alapján, és minden esetben a nagyobb érték jelent jót. Ezután készül egy Event objektum a számított érték alapján. Az Event a keretrendszer osztálya, a belső kommunikációhoz szükséges. Az esemény küldésének menetét mutatja a 4.2. ábra.



4.2. ábra: kommunikáció a játékból a keretrendszer felé

Amikor a framework megkapja az üzenetet, elkezdi annak feldolgozását. Ehhez különböző modulokat vesz igénybe. Erről részletesebben írok a 4.3 fejezetben.

4.2.2 Keretrendszerből a játékok felé

A játékok és a keretrendszer kommunikációja leginkább egy szerver-kliens kapcsolatra hasonlít. A keretrendszer nem képes megszólítani a játékokat anélkül, hogy előbb jelezzék felé erre az igényüket. A keretrendszerből játékok felé folyó kommunikációt tehát szintén a játékok kezdeményezik a már említett `GameBase` használatával. Lehetséges ajánlatot kérni nehézségi szintre és jutalomra. Ennek folyamata a következő:

1. A játék jelzi, hogy igényel javaslatot. Ehhez értelemszerűen az alábbi két metódus használatos:

```
public void getSuggestedDifficulty()
public void getReward(Class<T> rewardType)
```

2. Az igények rögzítése után a keretrendszer kiszámolja a választ és a GameBase megfelelő callback-ein keresztül célba juttatja azt:

```
public void onRewardReceived(Reward reward)
public void onDifficultyChanged(float difficulty)
```

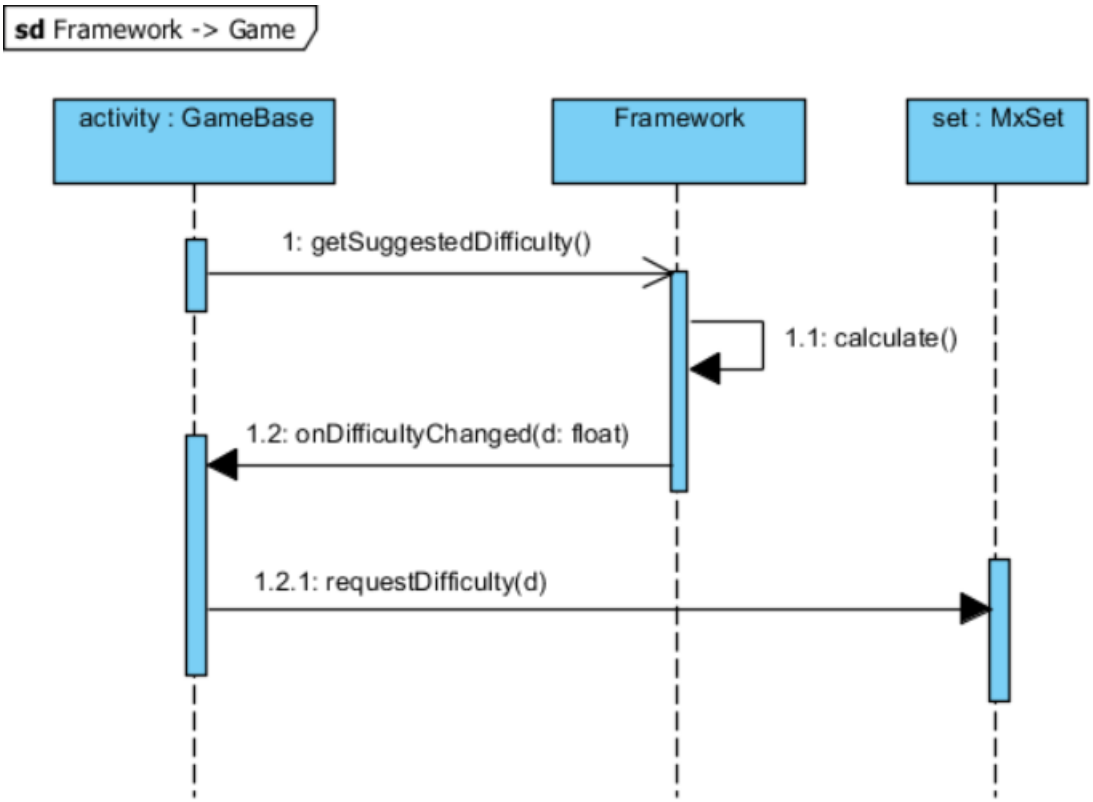
   Előbbi esetben egy jutalmat jelképező objektumot vesz át a GameBase, utóbbiban pedig egy lebegőpontos számot, melyet a jelenlegi nehézségi szint szorzójaként kell kezelni. Ez azt jelenti, hogy optimális nehézség esetén 1.0 lesz az értéke, csökkentés esetén egynél kisebb, növelés esetén pedig egynél nagyobb.

A jutalmazás a Reward osztály leszármazottain keresztül történik. A Meixner feladatokhoz egyetlen típusú jutalmat készítettem:

```
public class PuzzleReward extends Reward {

    public enum RewardType {
        NONE, ANIMATION, POINTS
    }

    private RewardType mRewardType;
    private float mMagnitude;

    //...
}
```

Jelenleg kétféle jutalom támogatott: pontok és animáció. A mMagnitude paraméter határozza meg a jutalmazás mértékét. Ez pontok esetén a konkrét pontszámot jelenti, míg animáció esetén pedig küszöbértékek szerint döntjük el az jelenet típusát. Mivel a jutalmakat aszinkron módon kapja meg az Activity, nem bízhatunk abban, hogy pont két feladat között érkezik meg. Amikor a felhasználó éppen megold egy feladatot és érkezik egy jutalom, nem szeretnénk megszakítani a folyamatát, ezért a jutalmat egy pufferbe kell helyezni és akkor lejátszani, amikor a felhasználó végzett a futó játékkal.

A javasolt nehézségi szint kérésének és feldolgozásának folyamatát mutatja meg a 4.3. ábra. Ezzel majdnem teljesen megegyező a jutalom feldolgozása.



4.3. ábra: javaslat kérése nehézségi szintre

4.2.3 A NeuroSky MindWave és a keretrendszer közt

Az EEG készülék bluetooth kapcsolaton keresztül kommunikál más eszközökkel. Induláskor a keretrendszer megpróbál csatlakozni a készülékhez, majd ha ez sikerült, nagyjából másodpercenként kétszer kap tőle adatot. Ezeket az adatokat a `NeuroSkyModule` osztály olyan formátumúra alakítja, hogy az továbbítható legyen a mérő modul felé. Jelenleg a következő információkat képes feldolgozni a keretrendszer:

- Jel erőssége. Csak akkor használjuk fel a készülékből érkező adatokat, ha azoknak megfelelő a megbízhatósága. A jelerősség nem a bluetooth összeköttetés minőségére vonatkozik, hanem az EEG által mért adatok pontosságára. Az érték a [0; 200] intervallumon mozog, ahol a 0 jelenti a kiváló jelet, a 200 pedig azt, hogy semmilyen megbízható mérés nem lehetséges.
- Figyelem mértéke. Egész szám a [0; 100] intervallumon, a nagyobb érték jelent magasabb szintű figyelmet.
- Nyugodtság mértéke. Szintén a [0; 100] intervallumból származó egész szám. Magasabb érték nyugodtabb felhasználóra enged következtetni.

- Nyers adatok: feldolgozatlan adatok a felhasználói agyhullámairól. A keretrendszer önmaga is tud analizálni ilyen adatokat, így nem muszáj kizárólag az EEG készlék által számított származtatott adatokra támaszkodni. A nyers adatokból ki tudom számítani, hogy az EEG viselőjének mely frekvenciatartományokban legaktívabb az agyműködése, amelyből a vonatkozó szakirodalom felhasználásával [16] szintén a mentális állapotra tudok következtetni.

A `NeuroskyModule`-nak a figyelemre és nyugodtságra vonatkozó adatokkal nincs különösebb dolga, ugyanis szerencsére a mérő modul fogadni tudja őket ebben a formátumban.

4.3 Javaslat számítása

A keretrendszer legfontosabb funkciója a különböző adatok feldolgozása, tárolása, majd javaslatok számítása ezek alapján. A jelen lévő szenzoroktól és a futó játék által kínált eseményektől függően rengeteg féle jelet tudunk fogadni. Ezen jelek az őket feldolgozó egységhez jellemzően nem nyers formában jutnak, hanem valamilyen egységes interface-re illesztik őket a hozzájuk tartozó modulok.

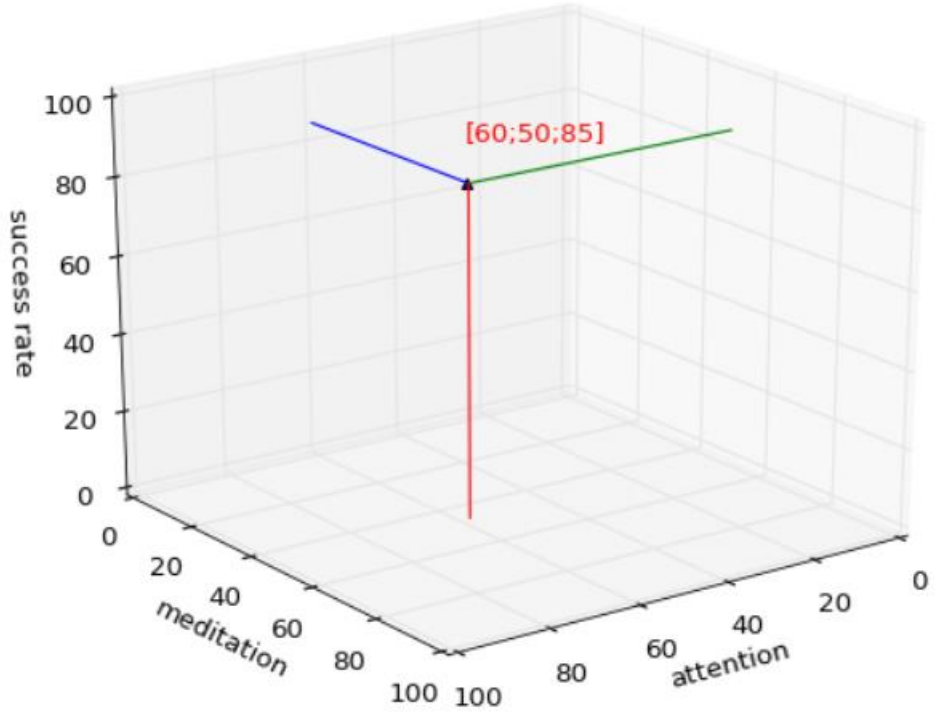
4.3.1 Elméleti háttér

A számítások során a következő származtatott paramétereket veszem figyelembe:

- *Figyelem*
- *Nyugodtság*

Ezen kívül a játékokból érkező üzeneteken keresztül a megoldások *sikerességéről* is megbízható képet tudok alkotni, ez lesz a harmadik paraméterem. Mindhárom paramétert százalékként kezelem, tehát a 0-100 intervallumon mozgó egész számokról van szó. Az itt leírt számítási mechanizmusok nem specifikusak erre az egy eszközre – amennyiben egy másik eszköz által szolgáltatott jelekből is számíthatók ezekre a mentális tulajdonságokra vonatkozó értékek (ilyen például a szívritmus variancia, mely alapján a figyelem szintjét lehet megállapítani), semmi akadálya, hogy ezek is hozzájáruljanak a számítás paramétereinek megfelelő beállításához. Dolgozatomban az új NeuroSky EEG készülékkel foglalkoztam, mely ára és elérhetősége miatt optimális tantermi használatra is a Meixner iskolában.

E három paramétert úgy érdemes elképzelni, mint egy teret meghatározó három tengelyt. A felhasználó minden pillanatban ennek a térnek egy adott pontjában helyezkedik el.



4.4. ábra: példa a paraméterek által alkotott térre

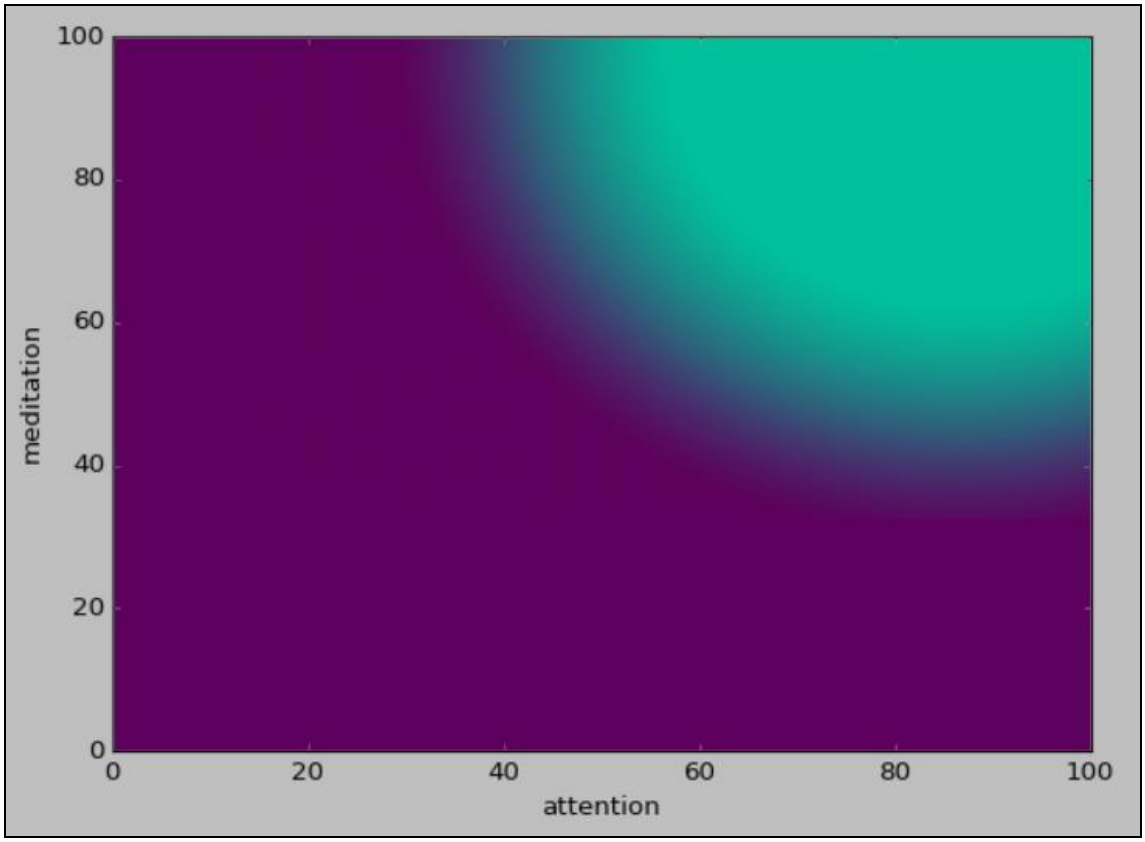
A 4.4. ábra az imént említett teret demonstrálja. A felhasználó a jelzett pontban helyezkedik el ezen a téren belül, amikor a figyelmének értéke 60%, nyugodtsága 50%, a feladatok megoldása során pedig 85% a sikeressége. A javaslatok számításához ennek a térnek az analízisére van szükség.

4.3.1.1 Figyelem és nyugodtság

Az első megközelítésben csak a figyelemmel és a nyugodtsággal fogunk foglalkozni. Ez a két paraméter közvetlenül jellemim a felhasználó agyát, ezért érdemes először a belőlük levonható, egyebektől független következtetéseket vizsgálni. Az optimális tanulás [8][9][10] állapota akkor lehetséges, ha:

- A felhasználó figyelme megfelelően magas. Amennyiben ez az érték alacsony, az optimális tanulás nem lehetséges, hiszen bármi történik, a felhasználó nem fogadja be a visszacsatolást.

- A felhasználó megfelelően nyugodt. A nyugodtság alacsony százalékos értéke idegességre, frusztráltságra enged következtetni, ez pedig akadályozza a tanulásban. Az EEG készülék által szolgáltatott érték szerint az 50 a normális szint, az ettől való távolság adja meg, hogy mennyire optimális a felhasználó nyugodtsági szintje. Ezt az értéket először konvertálnunk kell százalékosra.



4.5. ábra: figyelem és nyugodtság értékek vizualizációja

A 4.5. ábra mutatja, hogy milyen tartományban lévő felhasználóknál állhat fent optimális tanulás. A sík világoskék része az, ahova szeretnénk a felhasználót eljuttatni. A tartomány belsejében (tehát nagyjából 70% figyelem és 70% nyugodtság felett) van lehetőség a hatékony tanulásra.

4.3.1.2 Sikeresség

A megoldások sikeressége önmagában is egy nagyon hasznos indikátora a tanulási folyamat hatékonyságának. Ez az adat egyszerre hordoz információt arról, hogy a felhasználó mennyire megterhelőek a feladatok, és arról, hogy az eddig megoldottakat mennyire sikerült elsajátítani. Alacsony érték esetén arra következtethetünk, hogy a

feladatok nagyon nehezek, a felhasználó nem tudja őket jól megoldani. Ez egyrészt azt jelenti, hogy rosszul érzi magát, mivel a kapott jutalom nem egyezik meg elvárásaival, ennek nagyságát ugyanis a keretrendszer elsősorban a sikeresség értékéből számolja. Másrészt ha a felhasználó már oldott meg feladatsorokat, akkor arra következtethetünk, hogy a tanulás nem volt hatékony, ugyanis a tananyagot nem sikerült elsajátítania.

Az tanulási hatékonyság szempontjából optimális sikerességi tényező 65% közelében van [15][17][18]. A keretrendszer hosszú távú célja az is, hogy erről az optimális tartományról a szerver segítségével adatokat gyűjtsön, hogy ezek analízisén keresztül új, pontosabb modelleket állíthassunk fel.

4.3.1.3 Aggregált képlet

A cél természetesen az, hogy mind a három paramétert felhasználjuk a javaslatok számításánál. Ehhez két függvényre van szükségem: egy a nehézségi szintet, egy pedig a jutalmat fogja megadni. Ehhez először kiszámolom, hogy az egyes paraméterek szerint külön-külön milyen értéket kellene javasolni:

- figyelem: $$d_a(a) = \begin{cases} 1, & ha\ a \geq K_a \\ K_a\ /\ a, & egyébként \end{cases}$$
- nyugodtság: $$d_m(m) = \begin{cases} 1, & ha\ m \geq K_s \\ m\ /\ K_s, & egyébként \end{cases}$$
- sikeresség: $$d_s(s) = K_s\ /\ s$$

Ezek a függvényeket úgy konstruáltam meg, hogy eredményük az optimális érték felé mozdítsa a megfelelő mutatókat. A képletekben szereplő K_a, K_m és K_s együtthatók segítségével lehet a számítási algoritmust finomhangolni. Előbbi kettő rendre a figyelemi és nyugodtsági szintek küszöbértékei. Amennyiben az adott fiziológiai jel eléri a hozzá tartozó küszöbértéket, nem szükséges a nehézséget változtatni, egyébként az eltéréssel arányosan módosítom azt. Nyugtalanság, frusztráltság jellemzően akkor jelentkezik, amikor a feladatok túl nehezek, így ebben az esetben csökkenteni kell a nehézséget. A figyelmetlenség azonban a túl könnyű feladatok indikátora, így a küszöbértéket el nem érő értékek esetében növelni fogjuk a nehézségi szintet. A K_s együttható jelöli az optimális sikerességet. Bármilyen fiziológiai jel esetén az optimális érték felé szeretnék

mozogni. Ezeket az együtthatókat az implementált algoritmusban különböző kutatások eredményei alapján [9][10][15] a következő módon választottam meg:

$$K_a = 60$$

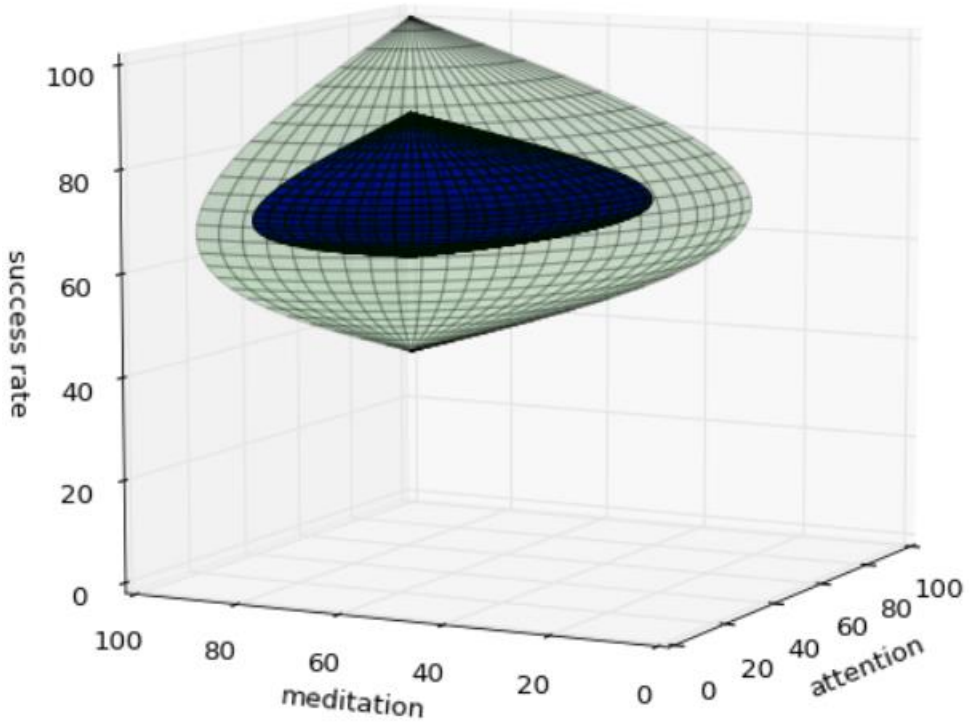
$$K_s = 50$$

$$K_m = 65$$

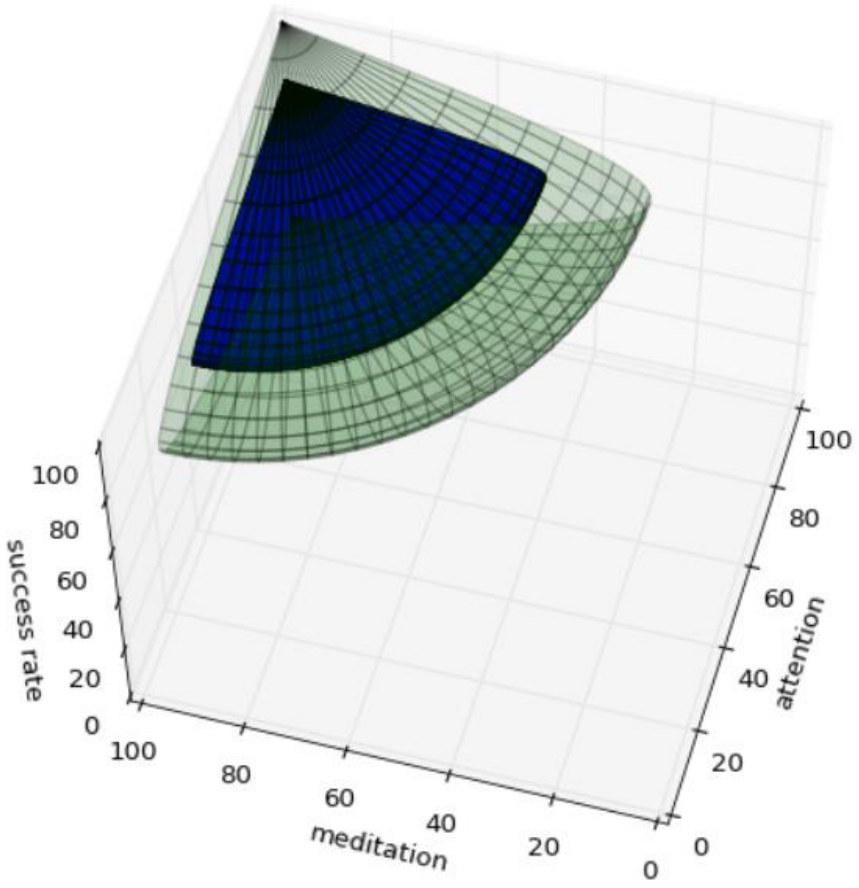
Ezek jelenleg konstans értékként jelennek meg a számítási algoritmusban, hosszú távon azonban használhatók arra, hogy még jobban az adott felhasználóra szabjuk a feladatokat. Ehhez szükség van egy-egy felhasználóról nagyobb mennyiségű adatra, ezért egyelőre az implementációba nem került bele. Az aggregált javaslat ezek után könnyen megkapható a következő képlettel:

$$diff(a, m, s) = d_a(a) * d_m(m) * d_s(s)$$

Ez az érték az, amit a játékok felé továbbítani kell. A 4.6. ábra és 4.7. ábra megmutatja, hogy a 3 paraméter által feszített térben hol helyezkednek el az optimális tanulást lehetővé tevő pontok. Amennyiben a külső, zöld burkon belül van a felhasználó, akkor nagy a valószínűsége az optimális tanulást lehetővé tévő mentális állapotnak, a felhasználó ilyenkor úgynevezett *flow* állapotban van. Az alkalmazás célja, hogy a felhasználó a kék burkon belülre tudjon jutni, ahol minden tényező optimális.



4.6. ábra: optimális tanulást lehetővé tévő pontok



4.7. ábra: optimális tanulást lehetővé tévő pontok, más szögből

A jutalom számítása kevésbé jól formalizálható, mint amit a nehézségi szint esetén láthattunk. Alapvetően kétféle jutalmat ajánlunk a felhasználónak:

- Figyelemfelkeltő jutalom: ha az olvasott értékek alapján azt látjuk, hogy a felhasználónak lankad a figyelme, érdemes egy-egy kisebb jutalmas javasolni, amely megragadja az ember tekintetét. Ez jellemzően egy kisebb animáció, beúszó kép, stb.
- Teljesítmény alapú jutalom: ilyen jutalmat a nehézségi szinthez hasonlóan tudunk számolni. Ebben az esetben játszik fontos szerepet a `PuzzleReward` osztály (4.2.2. fejezet) `magnitude` értéke, mely a jutalmazás mértékét tárolja. Hagyományosan, azaz biofeedback nélküli esetben a jutalom számítása a következő módon történik:

$$rwd_s(s) = C * s$$

C egy tetszőleges konstans, tehát a sikeresség a kapott pontszámmal egyenes arányos. A keretrendszer számára azonban egyéb adatok is rendelkezésre állnak, melyek alapján finomhangolhatjuk ezt az értéket. A legfrissebb kutatások alapján [9][10] a túlságosan alacsony figyelmi vagy nyugodtsági szint hatására érdemes csökkenteni a jutalmat. A következő képletet konstruáltam:

$$rwd_a(a) = \begin{cases} 1, & ha\ a \geq 30 \\ \frac{a}{30}, & egyébként \end{cases}$$

$$rwd_m(m) = \begin{cases} 1, & ha\ m \geq 30 \\ \frac{30}{m}, & egyébként \end{cases}$$

$$rwd(a, m, s) = rwd_s(s) * rwd_a(a) * rwd_m(m)$$

Kezdeti küszöbértéknek a 30%-ot jelöltem ki mind a figyelem, mint a nyugodtság esetén. Amennyiben a két paraméter közül bármelyik ez alá csökken, a jutalom arányosan csökkentésre kerül. Ez a választás jelenleg önkényes, az eddigi, nem reprezentatív méréseken alapszik. Pontos értékét a csoportos tesztek alapján, pedagógusi részvétellel fogjuk meghatározni.

4.3.2 Mérés és számítás

A keretrendszer különböző funkciói modulba vannak szervezve. Ilyen például a `GameProxyModule`, ami a játékokkal való kommunikációért felelős, vagy a `SupervisorProxyModule`, aminek segítségével a felügyelő alkalmazással tudunk kapcsolatot teremteni. A jelek mérését a `MeasurementModule` végzi, különböző *Calculator* osztályok segítségével.

4.3.3 SuggestionCalculatorStrategy

A `MeasurementModule` képes fogadni a figyelemre, nyugodtságra és sikerességre vonatkozó eseményeket a keretrendszeren belül. Ahhoz, hogy képes legyen ezek fogadása után javaslatokat tenni egy `SuggestionCalculatorStrategy` objektumot használ.

Ez az osztály 3 fontos tagváltozóval rendelkezik, melyek rendre eltárolják az aktuális figyelmet, nyugodtságot és sikerességet. Metódusain keresztül lehet ezeket az értékeket frissíteni. A konzisztens állapot fenntartásának érdekében ezek a változók nem egy-egy mérés eredményeit tárolják közvetlenül, hanem az eddig mért adatok aggregált értékét tartalmazzák. Ezzel biztosíthatjuk azt, hogy egy-egy hibás mérési érték ne eredményezhessen szélsőségesen hibás számítást, sokkal kisebb súlya lesz az új érték kiszámításánál. A figyelem új értékének kiszámítási módja:

$$Attention(a) = \frac{A_0 + C_{calc} * a}{1 + C_{calc}}$$

A képletben szereplő A_0 az eddigi számolt érték, a az új adat, C_{calc} pedig egy tetszőlegesen választható konstans a [0; 1] intervallumból. Minél nagyobb C_{calc} értéke, annál nagyobb mértékben képes egy-egy új adat a tárolt értéket megváltoztatni. A jelenlegi implementációban $C_{calc} = 0.1$. A nyugodtság és sikeresség értékének számítási módszere ezzel megegyező:

$$Meditation(m) = \frac{M_0 + C_{calc} * m}{1 + C_{calc}}$$

$$Success(s) = \frac{S_0 + C_{calc} * s}{1 + C_{calc}}$$

Mivel az adatok elég gyorsan érkeznek (másodpercenként 4-5 a figyelem és sikeresség esetében), nem szeretnénk minden egyes alkalommal hosszabb számításokat

végezni, csak minden n frissítés után. A jelenlegi implementációban az $n = 10$ értéket választottam. Minden 10. frissítés után a `SuggestionCalculator` küld egy javaslatot a keretrendszer felé, aki a játék felé akkor küldi ezt tovább, amikor az jelzi az igényét erre a 4.2.2 fejezetben bemutatott módon.

5 Összefoglalás

Eddigi munkámnak két fontos eredménye született. Sikerült kiépíteni a Meixner Alapítvány számára egy olyan rendszer, amely segítségével kényelmesen és hatékonyan lehet mind feladatokat készíteni, mind azokat tantermi körülmények között felhasználni az oktatásban. Másik fontos eredményem az új EEG készülék csatlakoztatása az AdaptEd keretrendszerhez, valamint egy új javaslat számítási algoritmus implementálása, mely a legújabb tudományos kutatások eredményeit felhasználva ajánl nehézségi szintet és jutalmat a játékoknak. Az így készített adaptív játékok hatékonnyá és élvezetessé teszik az oktatást. Jelenleg is folyik számos kutatás tanszékünkön belül és azon kívül is, melyek eredményeit fel fogom használni az algoritmus finomításához, amint azok rendelkezésre állnak. Az alapítvány számára kiépített alkalmazások és a keretrendszer új komponenseinek legmérvadóbb tesztje az éles körülmények közötti használat lesz, reményeim szerint ebből rengeteg következtetést le fogok tudni vonni, melyek segíteni fognak a fejlesztés irányát meghatározni.

Az alkalmazás eddigi tesztjeinken nagyszerűen teljesített, érezhetően jó döntéseket hozott a feladatok összeválogatásánál. A következő lépés az, hogy egy szélesebb körű tesztelésnek vessük alá éles környezetben.

5.1 Tervek és lehetőségek

A feladatkészítő alkalmazás végleges verzióját szeptember végén adtuk át, azóta a Meixner Alapítvány munkatársai elkezdték feltölteni a tananyagot. Ez egy hosszú folyamat, reményeim szerint néhány héten belül már elég tartalom lesz ahhoz, hogy készítsünk néhány Android alkalmazás verziót belőle.

Az alapítványon belül jelenleg is folyamatban van azoknak a táblagépeknek a beszerzése, amelyeket a tanulók a tanteremben igénybe tudnak majd venni feladatok megoldásához. Az alapítvány munkatársainak saját elmondása alapján egyébként a tanulók igen nagy százalékának van a családjában táblagép, így az otthoni, önálló munka is lehetséges legtöbbjüknél. Amint végére ér a beszerzési folyamat és implementálom az alkalmazás kinézetét, el tudjuk kezdeni szervezni először a rendszerek tesztjét az iskolában, végül pedig beüzemelhetjük azt éles használatra.

Hosszú távon a Meixner Alapítvány és a tanszék készen áll arra, hogy a szükséges tesztek elvégzése után az alkalmazást elérhetővé tegye mások számára is. Így például elmaradottabb régiókban, ahol nem áll rendelkezésre megfelelő pedagógus, lehetséges lenne a képességzavaros gyerekek megfelelő gondozása és fejlesztése távoli felügyelettel.

Irodalomjegyzék

[1] Meixner Alapítvány honlapja, http://meixner.hu/alapytvanyunk/

[2] R.S. Sutton, A.G. Barto, "Reinforcement Learning", MIT Press, Cambridge, MA., 1998.

[3] NeuroSky MindWave Mobile: http://store.neurosky.com/products/mindwave-mobile

[4] Bootstrap UI: http://getbootstrap.com/

[5] jQuery: http://jquery.com/

[6] jQuery UI: http://jqueryui.com/

[7] Android NDK: https://developer.Android.com/tools/sdk/ndk/index.html

[8] Felipe Gerhard, Szegletes Luca. „Spline- and wavelet-based models of neural activity in response to natural visual stimulation" (IEEE) New York: IEEE Press, pp. 4611-4614. ISBN: 978-1-4244-4120-4, San Diego, USA, 2012.

[9] Szegletes Luca, Forstner Bertalan: „Towards biofeedback-controlled self-rewarding learning with mobile devices", (IEEE) New York: IEEE, pp. 303-308. ISBN: 978-1-4673-5187-4, Kassa, Szlovákia, 2012.

[10] Szegletes Luca, Forstner Bertalan, „An Introduction to a Neurofeedback-based Framework in Developing Learning Ability on Mobile Devices" ISBN: 978-1-61804-147-0, Montreux, Svájc, 2012.

[11] P. Valdés, J. Bosch, R. Grave, J. Hernandez, J. Riera, R. Pascual, R. Biscay: Frequency domain models of the EEG, 1992

[12] Farwell LA, Donchin E, The truth will out: interrogative polygraphy with event-related brain potentials. 1991.

[13] M Kutas, G McCarthy, E Donchin: Augmenting mental chronometry: the P300 as a measure of stimulus evaluation time, 1977.

[14] Matteo Carandini, Jonathan B. Demb, Valerio Mante, David J. Tolhurst, Yang Dan, Bruno A. Olshausen, Jack L. Gallant, and Nicole C. Rust, "Do We Know What the Early Visual System Does?," Journal of Neuroscience, vol. 25, no. 46, pp. 10577–10597, Nov. 2005.

[15] Csíkszentmihályi Mihály, „Flow: The psychology of optimal experience", 1990

[16] T. Egner, J.H. Gruzelier, EEG Biofeedback of low beta band components: frequency-specific effects on variables of attention and event-related brain potentials, 2004.

[17] Paul A. Kirschner: Cognitive load theory: implications of cognitive load theory on the design of learning, 1997.

[18] Clark, Ruth C., Frank Nguyen, and John Sweller. Efficiency in learning: Evidence-based guidelines to manage cognitive load. John Wiley & Sons, 2011.