BeeSmarter

BeeSmarter - 24 órás Mobil Programozó Csapatverseny
Budapest, 2013. március 2-3.

Főtámogatónk:

Arany fokozatú támogatónk:

Szakmai partnerünk:

A verseny szervezői:

Magyar Villamosmérnök- és Informatikus-hallgatók Egyesülete
www.eestec.hu

fides et ratio

PPKE HÖK ITK

SJC

Magyar Művészeti Akadémia

# BeeSmarter

## A verseny célja

A programozói verseny feladata három különböző alkalmazás lefejlesztése (PulsePlayer, ARDodgeball, QRTicketer), melyek együtt széles mobil programozói  területet fednek le. A három alkalmazás tökéletes elkészítésére a 24 órás versenyidő kevésnek bizonyulhat, ugyanakkor az értékelési pontrendszerben a részleges megoldásokra részpontszámok szerezhetőek.

A verseny során hangsúlyos szerepet kap a hatékonyság: a részfeladatokra bontás, és a programozói erőforrások helyes allokációja kritikus ebből a szempontból. A verseny kezdetén érdemes ennek megtervezésére kellő időt szánni.

A programozói versennyel párhuzamosan dizájner verseny is zajlik, ahol a dizájner csapatok a programozók által kifejlesztett alkalmazásokhoz készítenek grafikai elemeket. A mobil alkalmazások esetén kulcskérdés a felhasználói élmény maximalizálása, ami a dizájnerek és a programozók szoros együttműködését igényli. Ez a kapcsolat azonban nem mindig kiegyensúlyozott, a versenyen 24 órába sűrítettük azt a gyakran hónapokig tartó egyeztető folyamatot, ami végül a tökéletes felhasználói élményt nyújtó alkalmazást eredményezi.

### A verseny lebonyolítása

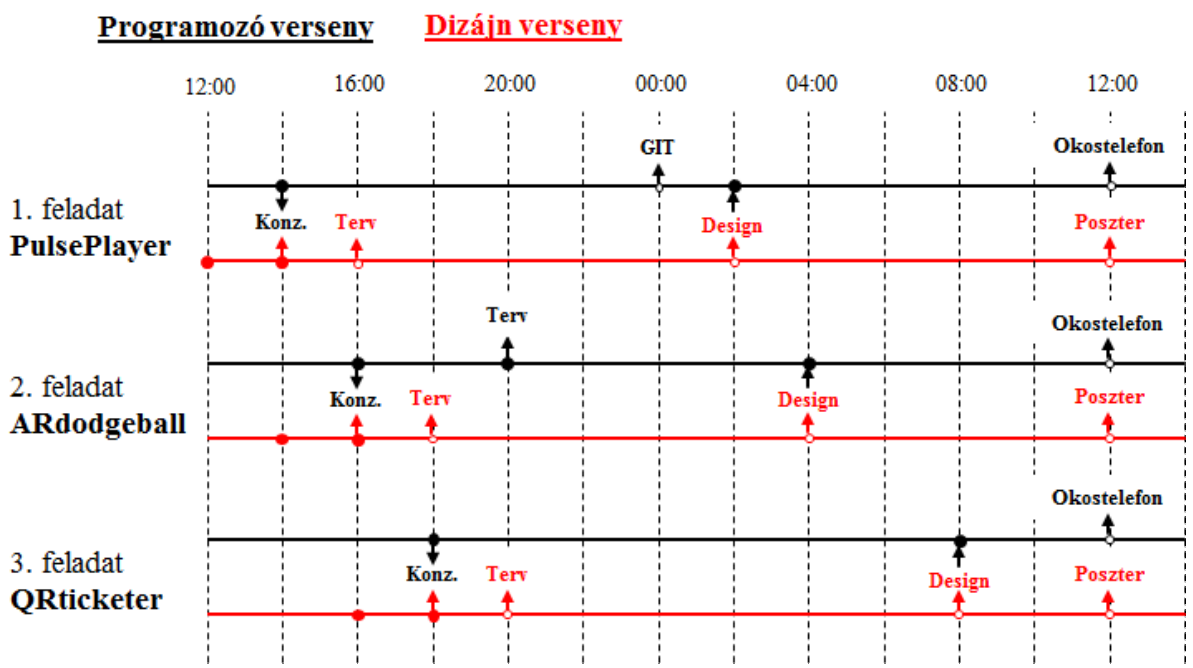
A verseny során több programozói csapat mellett lesz egy-egy dizájner csapat. A dizájner csapat alakítja ki mindhárom alkalmazás végső UI koncepcióját, de előtte egyeztetni illetve konzultálni fog a programozó csapatokkal, akik elmondják a programozó oldali véleményüket és elképzeléseiket. Az együttműködés sikeressége is értékelési szempont (a részleteket az Értékelési szempontok szakaszban írtuk le)!

A verseny közben a dizájnercsapattal történő egyeztetéseken kívül két mérföldkő van:

- az 1. feladat (PulsePlayer) pulzusdetekció részének értékelése 00:00-kor,
- a 2. feladat (ARDodgeball) esetén a kommunikációs protokoll (KommTerv.pdf) és a játékmodell (JatekTerv.pdf) leírását kell beadni 20:00-kor.

A verseny végén az okostelefonra kell telepíteni az alkalmazásokat, a kiértékelés az okostelefonokon futó API-kon történik.

A verseny ütemterve a következő oldalon látható.




## Értékelési szempontok

A beadott okostelefonon futó alkalmazások az egyes feladatoknál részletezett módon - az automatikusan kiértékelhető pulzusdetekción (1. feladat: PulsePlayer) kívül - manuálisan lesznek pontozva. Ugyanakkor a programozó - dizájner együttmüködés sikeressége is helyet kap a pontozási rendszerben az alábbiak szerint:

| Értékelési szempont | Rövid leírás | Elérhető pontszám |
|---|---|---|
| 1. feladat (PulsePlayer) | GIT és manuális kiértékelés | 1000 pont |
| 2. feladat (ARdodgeball) | manuális kiértékelés | 1100 pont |
| 3. feladat (QRticketer) | manuális kiértékelés | 400 pont |
| Dizájner értékelés (a verseny végén kitöltött űrlap alapján) | A programozók együttmüködési és kommunikációs képességeit méri dizájner szempontból. | 100 pont |
| A dizájnnal megvalósított alkalmazások száma (feladatonként 0 vagy 50 pont) | | 150 pont |
| | **Összesen:** | **2750 pont** |

A dizájner értékeléshez hasonlóan a programozók is értékelik a designereket ugyanolyan együttműködési és kommunikációs szempontok szerint.

# 1. Feladat: PulsePlayer

A feladat egy olyan zenelejátszó elkészítése, ami a felhasználó pulzusának megfelelő zenét játszik le.

Az alkalmazás indulásakor a felhasználó a telefonjával 15 másodpercig méri a pulzusát a készülék mikrofonjával. Az alkalmazás indulásakor a felhasználó a telefonjával 15 másodpercig méri a pulzusát a készülék mikrofonjával. A mérést a "*Scan heart rate*" gombbal indítja, a kezdőképernyőn csak ez látható. Ha a mérés sikertelen volt, akkor hibaüzenet jelenik meg a képenyőn, majd újból visszatérünk kezdőképernyőre.

Sikeres mérés után megjelenik a mért pulzus átlaga és a három opció választó gomb: "*Speed up*", "*Match it*", "*Calm down*". Egy "*Rescan*" gombbal lehet visszajutni a kezdőképernyőre.

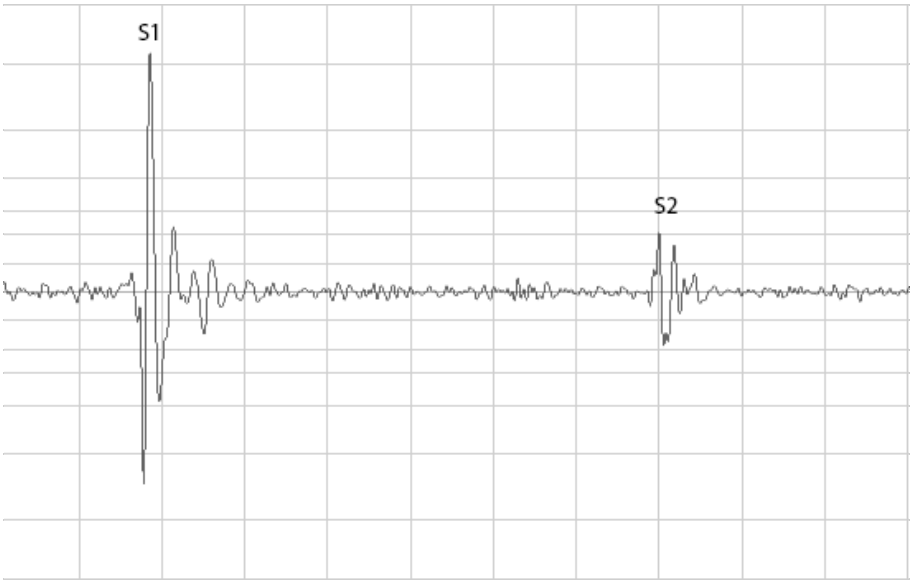
Kiválasztás után a lejátszó elindít egy olyan számot, ami a korábbi 3 opciónak megfelelően a felhasználó pulzusához viszonyítva gyorsabb, lassabb, vagy közel azonos tempójú. A szám elindulásával megjelennek a hagyományos lejátszó vezérlők vezérlők ("*Play*", "*Stop*" / "*Pause*", "*Next*", "*Last*") valamint egy "Back" lehetőség, amivel visszajuthatunk a tempó választó képernyőre (ekkor a zene leáll).

### Előkészületek:

A PulsePlayert implementáló csapatoknak regisztrálnia kell a The Echo Nest API honlapján (https://developer.echonest.com/account/register), hogy a megvalósításhoz szükséges érvényes API kulccsal rendelkezzenek.

### I. blokk: Pulzusdetekció – deadline Szombat 23:59

A feladat a készülék mikrofonjával, vagy mikrofon bemenetén készített felvételen a szívritmus megállapítása. Ehhez egy egyszerű módszert ismertetünk, aminek implementálása során lehetőség van annak javítására, paraméterváltoztatásokkal való optimalizálására. A feladat implementálásához segítségül 3 szívhang felvételt tettünk elérhetővé a http://contest.beesmarter.org/bs13sc/pulseplayer/heartsounds/ címen. Minden felvételből egy eredeti és egy szűrt verzió található, valamint a felvételhez tartozó az algoritmustól elvárt kimenet.




*Szívütés egy mobileszközzel készített felvételen*

### 1. A felvétel szűrése

A szívhangok tipikusan a 30-400 Hz-es tartományban hallhatóak, ezért egy digitális sáváteresztő szűrővel megszabadulhatunk a más tartományokba eső zajoktól. Erre a legyegyszerűbb megoldás egy FIR (Finite Impulse Response) szűrő használata.
Tájékoztatók és emlékeztetők a FIR szűrő működéséről:

- FIR szűrő általánosan: http://cnx.org/content/m12009/latest/
- FIR szűrő alkalmazása: http://cnx.org/content/m13904/latest/
- FIR struktúrák: http://cnx.org/content/m11918/latest/

A szűréshez szükséges együtthatók a következő oldalon számolhatóak ki: http://contest.beesmarter.org/bs13sc/pulseplayer/filterdesign/

### 2. A szívütések távolságának meghatározása a szűrt felvételen

A készülék mikrofonjáról érkező jelen az ábrán látható úgynevezett S1 és S2 szívhangok ismétlődnek periodikusan. A feladat két egymást követő S1 szívhang távolságának meghatározása.
A szűrt felvételt 3 másodperces szakaszokon vizsgáljuk, minden szakaszon a következőket kell elvégezni:
Másoljuk le a szakasz első másodpercét, és végezzük el az így kapott két vektor csúszóablakos normalizált keresztkorrelációját.

$$C_n = \frac{\sum_{p=0}^{P-1} A_{n+p} B_p}{\sqrt{\sum_{p=0}^{P-1} A_{n+p}^2 \cdot \sum_{p=0}^{P-1} B_p^2}}$$

$A$ : *az eredeti jelszakasz*

$B$ : *az eredeti szakasz első másodperce*

$C$ : *a csúszóablakos normalizált korreláció eredményvektora*
$P$ : *a B vektor hossza*
$N$ : *a C vektor hossza*

Ezt követően egy 60%-os threshold függvénnyel meghatározható a második S1 ütés helye. Az C eredményvektor eleje értelemszerűen azt mutatja majd, hogy az első pozícióban ütés található, ami természetes, hiszen az egymásodperces B szakasz megegyezik az A első másodpercével. Ezért a threshold függvénnyel való vizsgálatot célszerű a C vektor az elejét figyelmen kívül hagyva elvégezni.
Az így megtalált első 60%-nál nagyobb értékű elem időbeli távolsága a 0. elemtől lesz a két ütés távolsága ($\tau -$ *a két ütés közötti mintavételek száma*).

### 3. Szívritmus kiszámolása

A szívritmus mértékegysége BPM (azaz beats per minute), ez a következőképpen számolható:

$$BPM = round(\frac{60\,fs}{\tau}) \qquad fs : a\ mintavételi\ frekvencia$$

Az így kapott BPM értéknél fontos ellenőrizni, hogy érvényes legyen, azaz körülbelül 60 és 140 BPM közé essen. Nyugalmi állapotban, ha nem egy jó kondícióban lévő atlétát mérünk, akkor ez az érték 60 és 100 BPM között változik. Az ezen intervallumokon kívül eső értékeket érdemes érvénytelennek teinteni. Ez a pontatlanság az ára az algoritmus egyszerűségének.

## Megkötések és megjegyzések az Android implementációhoz

A fenti algoritmust az `org.beesmarter.pulseplayer.bpm` csomag `BSPulsePlayerBPMSolution.java` osztályában a `BSHeartSoundSubmitInterface.java` implementálásával a `calculateBPMWithRecord` metódusban kell megvalósítani. A metódus tetszőleges hosszúságú tömböt kaphat bemenetként és a bemenettel megegyező elemszámú tömbbel kell visszatérnie. Az eredmény tömb elemei legyenek a bemenet tömb minden egyes eleméhez tartozó BPM értékek. A szükséges fájlok a http://contest.beesmarter.org/bs13sc/pulseplayer/ címen elérhetőek el.

A mikrofonról való beolvasáshoz segítségül a `BSAudioIn.java` és `BSAudioListeningInterface.java` használhatóak.

## Megkötések és megjegyzések az iOS implementációhoz

A fenti algoritmust a `BSHeartSoundSubmitProtocol.h`-ban leírtaknak megfelelően a `calculateBPMWithRecord` metódusban kell megvalósítani. A metódus a `BSPulsePlayerBPMSolution.h` és `BSPulsePlayerBPMSolution.m` fájlokban legyen, az osztály a fenti protokollt valósítsa meg. A metódus tetszőleges hosszúságú tömböt kaphat bemenetként és a bemenettel megegyező elemszámú tömbbel kell visszatérnie. Az

eredménytömb elemei legyenek a bemeneti tömb minden egyes eleméhez tartozó BPM értékek.

A mikrofonról való beolvasáshoz segítségül a http://contest.beesmarter.org/bs13sc/pulseplayer/ címen elérhető `MixerHostAudio.h`, `MixerHostAudio.m` és `BSAudioListeningDelegate.h` használhatóak.

### Az első blokk megoldásának értékelése

A beküldött megoldásokat egy referenciahalmazon teszteljük az algoritmusok erdménye és egy referencia eredmény összehasonlításával. Az első blokk pontszámai a következőek szerint alakulnak:

$$P_1^{basic} = \frac{2}{1+\frac{1}{N}\sum_{i=0}^{N-1} f(|s_{ref_i} - s_i|)} - 1 \qquad f(x) = \{0 : x = 0 \mid 1 : x > 0\}$$

$$P_1 = 400 \cdot P_1^{basic} + 200 \cdot \frac{1}{t+1}$$

$P_1$ : *első blokkban elért pontszám*
$s_{ref}$ : *a referencia eredmény*
$s$ : *a beküldött algoritmus eredménye*
$t$ : *futási idő*

Így az első blokkban maximálisan elérhető pontszám 600 pont.
Az elkészült megoldást a git@repo.beesmarter.org:[csapat_azonosito].git repositoryba kell feltölteni. A repository-k a verseny kezdetétől 12 órán át lesznek elérhetőek (24.00-ig).

### II. blokk: Lejátszó elkészítése – deadline Vasárnap 12:00

*Az alkalmazás felépítése*

***1. Kezdőképernyő***

A kezdőképernyőn két lehetősége van a felhasználónak: elindítani a pulzusmérést vagy megnyitni a beállításokat. A pulzusmérés kétféle üzzemmódban működjön: normál és tesztmódban. Ezeket a Beállítások képernyőn lehet beállítani. Normál módban az első blokkban megvalósított hangfelvétel és jelfeldolgozás induljon el a "*Scan heart rate*" érintésére. Az indítástól kezdve 15 másodpercig füleljen a készülék mikrofonja, majd pedig számolja ki a 15 másodperces felvétel BPM átlagát. Nem valid eredmény esetén jelezze a problémát, majd maradjon a kezdő képernyőn és várjon új mérés indítására.
Teszt módban a fenti algoritmus helyett azonnal a beállítások képernyőn megadott BPM érték legyen az eredmény. Ha a számolás lefutott, akkor jelenjen meg a Tempó választó képernyő.

***2. Tempó választó képernyő.***

A tempó választó képernyőn jelenjen meg a mért pulzus, valamint 3 opció: "*Speed up*",

"*Match it*", "*Calm down*". Ez utóbbiakkal felhasználó kiválaszthatja, hogy az alkalmazás milyen tempójú zenét keressen lejátszásra.

| | |
|---|---|
| Speed up | pulzus + 15 < tempó < pulzus + 35 |
| Match it | pulzus - 10 < tempó < pulzus + 10 |
| Calm down | pulzus - 35 < tempó < pulzus - 15 |

Ha az alkalmazás nem talál a kérésnek megfelelő zenét, akkor ezt jelezze vissza és maradjon a tempó választó képernyőn. Legyen lehetőség visszalépni a kezdőképernyőre egy "*Rescan"* gombon keresztül.

A zenéket a következő címen található fájlokból kell kiválasztani:
http://contest.beesmarter.org/bs13sc/pulseplayer/songs/

A zenék tempójának meghatározásához a The Echo Nest API-t kell használni, az ehhez szükséges API hívások leírása a http://developer.echonest.com/docs/v4/track.html oldalon található. Az alkalmazástól elvárjuk, hogy a zenefájlok megváltozása után is megfelelően működjön (a kiértékeléskor ugyanezen fájlnevek alatt más felvételek lesznek). Ha több szám is megfelel a kritériumoknak, akkor a lejátszó egymás után játssza le mindet.

A megfelelő szám/számok kiválasztása után a lejátszó képernyő jelenjen meg és induljon el a lejátszás.

#### *3. Lejátszó képernyő*

A lejátszó képernyő megjelenésével elindul az alkalmazás által kiválasztott szám lejátszása. Megjelenik az aktuális számhoz tartozó előadó, cím, valamint a lejátszást vezérlő gombok.
"*Play* / *Stop* / *Pause*" - Lejátszás/Megállítás/Szünet
"*Next*" - Következő szám
"*Last*" - Előző szám
A tempó választó képernyőre egy "*Back*" gombbal lehet visszajutni.

#### *4. Beállítások képernyő*

A beállitások képernyőn két választható opciónak kell lennie: normál mód és teszt mód. Ezekkel lehet megadni, hogy a pulzus mérésének indításakor mi történjen a háttérben. Itt kell szerepeljen egy beviteli mező is, amellyel teszt módban beállítható, hogy pontosan milyen BPM értéket "mérjen" az alkalmazás.

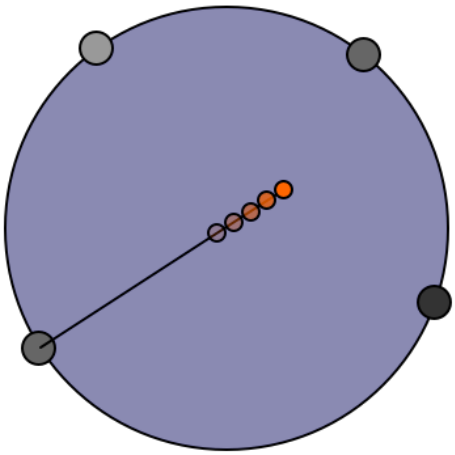
### A második blokk megoldásának értékelése

A megoldást a kiadott készülékre telepítve kell beadni a verseny végéig (12:00-ig). A második blokk implementálásával maximálisan 400 pont szerezhető, ehhez minden fent elvárt funkciót meg kell valósítson.

## 2. feladat: ARdodgeball

A feladat egy kidobós játék megvalósítása.

A klasszikus kidobóssal ellentétben ez a változat egy körpályán játszódik, ahol a pálya szélén mozoghatnak a játékosok, és a labdát mindig a középpontba célozva lehet elhajítani.



*Az ábrán látható a játék illusztrációja. A szürke pontok reprezentálják a játékosokat, a sárga az elhajított labdát. A célzás automatikusan a középpont felé irányul, illetve a játékosok kizárlóag a körpálya ívén mozoghatnak.*

A játék csakis többjátékos módban játszható (tehát legalább két játékos szükséges). A játék indításakor az egyik játékos a szerver szerepében indul, a többi kliens csatlakozik hozzá. A szervert futtató játékos szintén részt vesz a játékban.

### A megvalósítás

A játékot úgy kell megvalósítani, hogy a játékosok a pozíciójukat a készülék kamerájának segítségével határozzák meg úgy, hogy egy markert kell, hogy detektáljanak, amelynek a pozíciója határozza meg az orientációjukat. Igy ha a játékosok az eszközeikkel körbe-körbe járnak a marker körül, az őket reprezentáló alakok velük szinkronban mozognak a játéktérben.

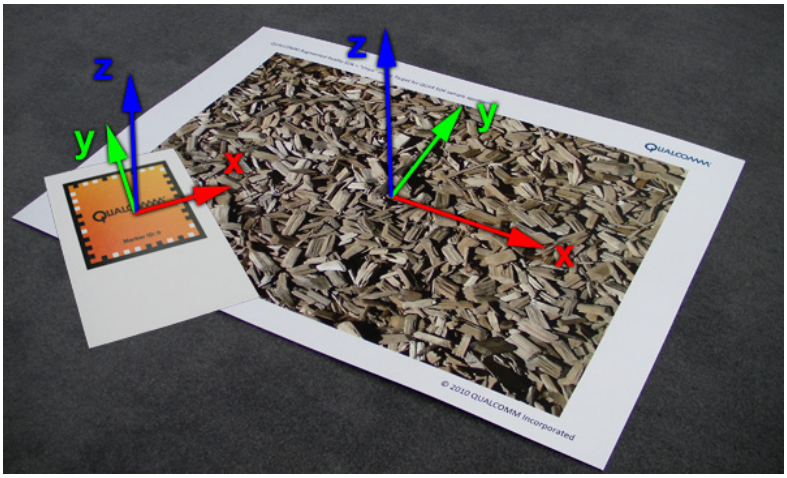
A detektáláshoz és a helyzetmeghatározáshoz a Vuforia SDK-t kell felhasználni.
https://developer.qualcomm.com/mobile-development/mobile-technologies/augmented-reality
http://www.youtube.com/watch?v=YhPsMn0s0rM

A **detektálás és pozíciómeghatározás** az alábbi módon valósítható meg Vuforia SDK-val:

1. Alkalmazás indításakor meghatározuk, hogy milyen típusú markereket (jeleket)

szeretnénk detektálni. Az SDK támogat **image markereket** és **frame markereket.** (lásd dokumentáció). Praktikusan a frame markert fogunk detektáltatni. (https://developer.vuforia.com/resources/dev-guide/frame-markers)

2. Ha a kameraképen látható marker, akkor az lekérhető a rendszertől. Ilyenkor egy markert reprezentáló objektumot kapunk vissza, amelynek lekérhetjük a térbeli pozícióját. Ez egy 3x4-es mátrix irja le, mely egy 3x3-as forgatási mátrixból és egy eltolásból áll. Ez reprezentálja a virtális terünk transzformációját. https://developer.vuforia.com/resources/dev-guide/trackable-base-class
3. A játék számára csak a *z* tengely körüli forgatás lényeges, ez alapján határozható meg a játékosok elhelyezkedése




*A pose matrix, ami a marker elhelyezkedését reprezentálja*

A klasszikus Augmented Reality alkalmazásokkal ellentétben a pose mátrixot csak részlegesen használjuk, és a játékot nem illesztjük a marker képére.

A **<u>játékosok</u>** **<u>közötti</u>** **<u>kommunikáció</u>** konkrét megvalósítására nincs megkötés. Ez praktikusan egy egyszerű kliens-szerver architektúra, ahol a kliens folyamatosan kommuninkálja a szerver felé az állapotának változásait. A játék állapota a szerveren frissül, majd a friss állapotokat szétküldi a klienseknek. (Természetesen a klienseken is futhat a játéklogika, ahol a kliens predikálja az állapotokat, majd ezt szinkronizálja a szervertől kapott állapotokkal, ez a csapat tervezési döntése.)

iOS platformon a GameKit framework hashasználatával megvalósítható a kommunikáció
http://www.raywenderlich.com/3276/how-to-make-a-simple-multiplayer-game-with-game-center-tutorial-part-12

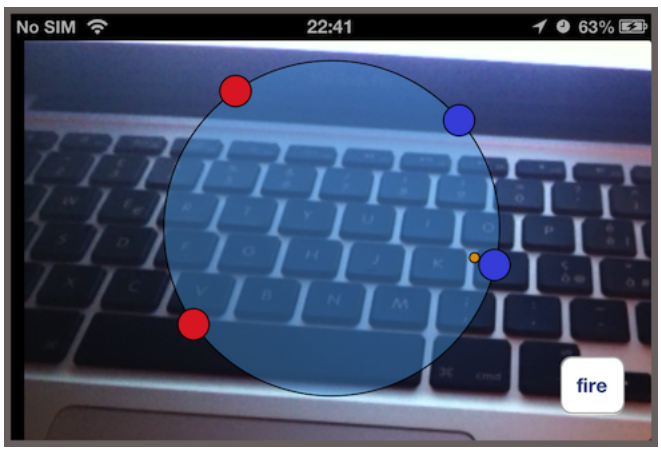
Az Enet könyvtár is hasznos lehet azoknak, akik alacsonyabb szinten szeretnék megvalósítani a kommunikációt:
http://enet.bespin.org/Tutorial.html

Java-s platform esetén a hagyományos socket-ek vagy a NIO rendszer használatával építhető

fel a rendszer. Természetesen ezektől eltérő megvalósítások is tökéletesen megfelelnek.

### Az alkalmazás felépítése

- **Kezdőképernyő:** az alkalmazás indítása után ez a képernyő fogadja a játékost, ahol eldönttheti, hogy szerver vagy kliens módban indul.
- **Szerver / Lobby képernyő:** ezen a képernyőn látjuk a szerver eléréséhez fontos információkat, az eddig csatlakozott kliensek listáját, és a játék indítására szolgáló gombot.
- **Kliens képernyő:** a szerver azonosításához szükséges információk bevitelére szolgál, illetve itt található egy csatlakozás gomb.
- **Játék képernyő:** magát a játékot reprezentáló képernyő. Egyrészt a kamera képét jelenítjük meg rajta, hogy a markert folyamatosan a kamera látóterébe tudja tartani a játékos, mástrészt a kidobós pálya amin a játék zajlik.




*A kamera képet a játéktér alá helyezve a markert bent tudjuk tartani a kamera látóterében és a játékteret is belátjuk. Természetesen nem kötelező ezt az elrendezést követni.*

### Egyedi megvalósítások

A játék logika megvalósításával semmilyen megkötés nincs. Azt hogy a játékosoknak egy vagy több élete legyen, vagy hogy egyszerre több labdával lehessen játszani a versenyzőkre van bízva. Ez igaz az alkalmazás felépítésére is. Az viszont elvárás, hogy a játékmenetet ponotsan definiálják a versenyzők, amit le is kell adniuk. Ez igaz a szerver kommunikációs protokolljának definiálására is.

### Az alkalmazás pontozása

A játék három fő modulból áll. Ezek midegyike önállóan is értékelhető. A maximális pontszámra értékelt alkalmazás feltételezi mindhárom modul megvalósítását és ezek együttműködését. Az alábbiakban szerepelnek a modulok és azok részfeladatai:

| Kommunikáció | Jelfeldolgozás | Játék |
|---|---|---|
| 1. Kommunikációs protokoll megtervezése (**KommTerv**)<br>2. Kommunikációs protokoll implementálása, használata<br>3. Kliens oldali állapottér predikció* | 1. POSE mátrix feldolgozása, orientáció meghatározása<br>2. A beérkező jel szűrése (POSE mátrixból feldolgozott forgatások sora)<br>3. Forgatás simítása interpolációval** | 1. A játék megtervezése, formális definiálása (**JatekTerv**)<br>2. játék modell implementálása<br>3. fizika megvalósítása |

*Kliens oldali állapottér predikció: a valós idejű kliens szerver architektúráknál általában a szerver számolja az állapotot, majd elküldi az eredményt a szervereknek. Azonban két állapotfrissítés között sok idő telhet el, ezért ilyenkor a kliens próbálja megjósolni, hogy mi lehet az aktuális állapot. Ezzel tudja eliminálni például az elveszett csomagok okozta zökkenéseket.

**A szűrésen túl akkor lehet szükség további simításra, ha két állapot közt nem ugrásszerűen váltunk, hanem folyamatosan konvergáltatjuk az új értékbe. A szűréssel elkerülhetjük a zajok okozta "remegést", a simítással a ugrásokat szüntethetjük meg.

Példarészlet a kommunikációs protokollra (beadandó: KommTerv.pdf):

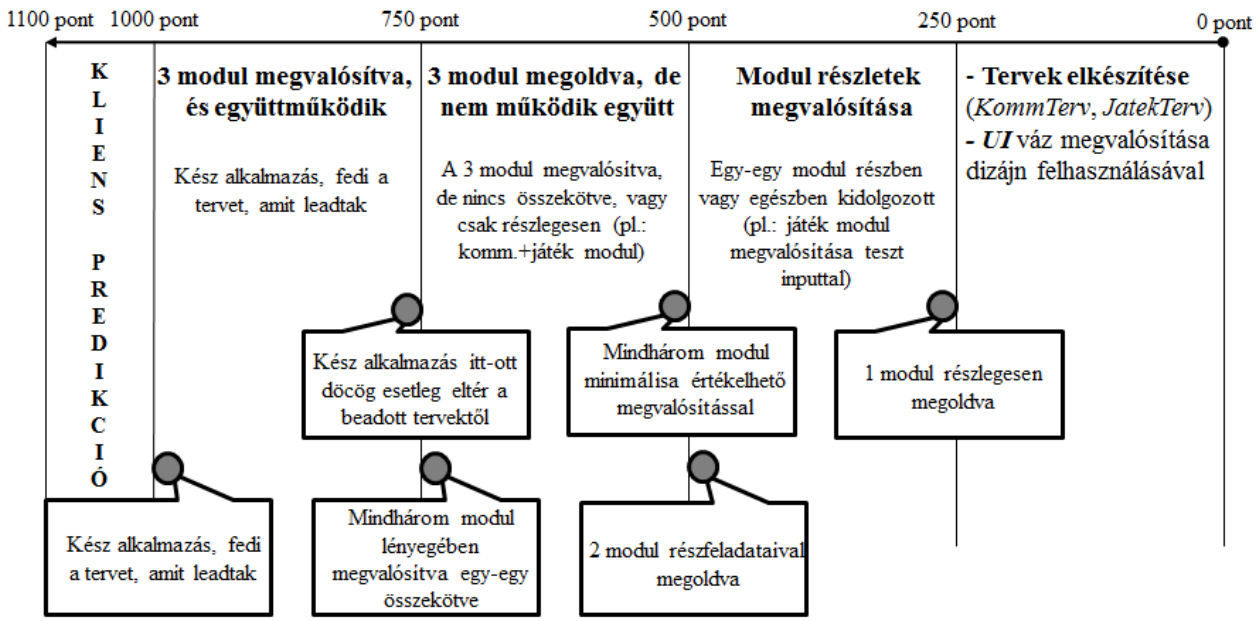
| Server | Kliens |
|---|---|
| ...<br>`S_ClientConnectionAck [ClientID]`<br>*visszajelzés a kliensnek, hogy kapcsolódhat, tartalmazza a kliens egyedi azonosítóját*<br><br>`S_UpdateState [GameStateRepresentation]`<br>*a játéktér aktuális modell reprenzentációját tartalmazza, ami alapján a kliensek frissíteni tudják a játékot*<br>... | ...<br>`C_ConnentMessage [Name]`<br>*kapcsolódási kérelem a szerver felé, egy "nickname"-et tartalmaz*<br><br>`C_UpdateState [InputState]`<br>*Kliens állapot küldése a szerver felé, hogy a játék állapotát frissíteni tudja*<br><br>... |

Példarészlet a Játék definiálására (beadandó: JatekTerv.pdf):

...
A játékban maximum 3 játékos vehet részt.
A játék egyetlen labdával játszódik, amit véletlenszerűen sorsolnak ki a játékosok közt.
He egy játékos eltatlál egy másik játékost a labdával akkor a dobó játékos 10 pontot kap.
A játéknak akkor van vége, ha valaki 60 pontot elért.
...

Mindkét leírás csak példa. A csapatoknak a JatekTerv.pdf és KommTerv.pdf dokumentumokat kell beadnia 20:00-kor. Erre azért van szükség, hogy a versenyzők előre átgondolják a feladatot és annak megvalósítását, ezzel megkönnyítve a verseny végén a modulok ill. részfeladatok összeillesztését.

A feladatra adható pontszám a megvalóstás mértékétől függ az alábbiak szerint:




Természetesen nem kell egy darab alkalmazásban bemutatni mindent, javasolt a feladatot alprojektekbe szervezni.

# 3. feladat: QRTicketer

Az alkalmazáson keresztül belépők és jegyek vásárolhatóak az aktuálisan akciós színházi előadásra vagy eseményekre (mozi, kiállítás, koncert stb.). A hirdetések megjelenhetnek utcán vagy újságokban, vagy bármilyen felületen, ahol elérhetőek a potenciális érdeklődök/jegyvásárlók.

Ezeken a hirdetéseken egy QR kódot (kétdimenziós vonalkód) helyeznek el, amit leolvasva az elkészítendő alkalmazással megkapjuk a kód alapján meghatározott kedvezményekkel az adott napi ajánlatokat.

Az alkalmazás több szempontból előnyös:

- nem kell minden nap más QR kódot elhelyezni a hirdetésekben, hiszen a felhasználó egy szervertől kapja az ajánlatokat,
- a fizikailag kihelyezett QR kóddal szűkíthető a célcsoport,
  (pl.: egyetemek közelében elhelyezett QR kódra, az egyetemistákat megszólító kedvezmények rendelhetők hozzá)

Ha a kapott ajánlatok elnyerik a felhasználó tetszését, akkor az alkalmazáson keresztül meg is vásárolhatja a jegyet, aminek ellenértéke a felhasználó telefonszámlájában kerül számlázásra. A megvett jegyet QR kóddal helyettesítjük, melyet az alkalmazás tárol, és szükség esetén megjeleníti.

## Elvárások:

- Tudnia kell kommunikálnia a szerverrel.
- Képesnek kell lennie a a szerver által küldött információk megjelenítésére.
- Tudjunk jegyet vásárolni vele.
- Meg kell tudnia jelenítenie a már megvásárolt jegyek QR kódját.
- Segítse a felhasználót a jegyvásárlásban, kerülje el a “véletlen” többszöri vásárlásokat, biztonságosan tárolja
- UI dizájn felhasználása
- Felhasználóbarát működés

## Az alkalmazás működése:

- QR kód beolvasása.
- A kód alapján, a szervertől kapott ajánlatok megjelenítése ezek kiválasztása, részletes megtekintése.
- Jegy vásárlása.
- Vásárolt jegy automatikusan tárolódik a saját jegyek között.
- A saját jegyeimet bármikor meg tudom tekinteni, azok minden adatával együtt, a jegy QR kódjával együtt.
- Az alkalmazás tudja a jegyek QR kódját leolvasás/azonosítás szempontjából megjeleníteni.
- naptárbejegyzés a saját naptárba

**Kommunikáció** a http://contest.beesmarter.org/bs13sc/qrticketing/ticket.php url-en a data változónak adott json string-gel. A beolvasott QR kódban lévő adat az “offerID” ezzel való lekérés:

{

```
  "requestID": "12345678987654",        //15 jegyű véletlen szám
  "msisdn": 36300000000,          //telefonszám
  "offerID": "x2t3johixe843lochgwh"       //ez jön a QR kódból
}
```

A jegy vásárlása során az offerID helyett az adott esemény azonosítóját kell küldeni.
A QR kódok olvasásához illetve generálásához használhatjátok a http://contest.beesmarter.org/bs13sc/qrticketing/ oldalon található ZXingObjC.framework-öt, illetve a .jar könyvtárat. Természetesen, ha van más kipróbált vagy saját könyvtáratok, akkor használhatjátok azokat is.

Példa QR

## Az alkalmazás pontozása

Az okostelefonra feltöltött alkalmazás értékelése manuálisan történik, és az alábbi funkciók megléte eredményez pontot:

| Funkció | Maximális pontszám |
|---|---|
| beolvassa a QR kódot és értelmezi | 60 pont |
| kommunikál a szerverrel, és értelmezi a válaszát | 60 pont |
| további kéréseket küld a szervernek | 40 pont |
| jegy vásárlása | 40 pont |
| vásárolt jegy QR megjelenítése | 40 pont |
| designer UI-t felhasznált | 60 pont |

| | |
|---|---|
| a jegy (és a korábban) megvásárolt jegyek tárolása | 30 pont |
| naptárbejegyzés | 30 pont |
| szubjektív: felhasználói élmény | 40 pont |
| **Összesen:** | **400 pont** |