Joc de dames multiplataforma per a dispositius mòbils

Memòria de Projecte Final de Màster

Màster Universitari en Aplicacions Multimèdia

Informàtica, Multimèdia i Telecomunicació

Autor: Rubén Amaro Parrado

Consultor: Sergio Schvarstein Liuboschetz

16 de juny de 2014

Crèdits/Copyright

Aquest projecte ha estat íntegrament realitzat per l'autor d'aquesta documentació, tant la part corresponent a l'aplicació com el text que a continuació s'exposa.

Dedicatòria/Cita

A la meva família ja que si sóc com sóc és gràcies a ells, i als meus amics ja que no he pogut estar tot el temps que voldria amb ells degut a la realització d'aquest projecte.

Agraïments

Al Arnau Olesti Recasens per ajudar-me en la realització del disseny de l'aplicació i al David Plaza Balagué, projectista d'aquest màster també, per compartir tot el camí que ens ha portat a l'entrega dels nostres respectius projectes.

Abstract

The project involves the analysis, design and implementation of a checkers game for different mobile platforms performing a mashup tools using HTML, CSS and Javascript.

For the realization of this project we have studied different softwares that handle make hybrid applications, these applications (Titanium, 2014) (PhoneGap) or (Intel XDK). The last one has been chosen to the performance of the application because it has all the tools needed since the creation of the code to the encapsulation for each operating system.

As already mentioned, the project has been divided into three phases: analysis, design and implementation.

In the analysis phase we have studied the different types of game that has the checkers, we have chosen the most appropriate for the project and also analyzed different features to add as might be possible to play with more than one device at a same time.

The design was made at low level, and as the project progressed have become more high-level design up to the final design made with proper tools for such tasks as the Adobe Illustrator.

Resum

El projecte consisteix en l'anàlisi, disseny i implementació d'un joc de dames per a diferents plataformes mòbils realitzant una aplicació híbrida utilitzant les eines HTML, CSS i Javascript.
Per a la realització d'aquest projecte s'han estudiat diferents programaris per a la realització d'aplicacions híbrides entre els que destaquen (Titanium, 2014), (PhoneGap) o (Intel XDK) essent aquest últim l'escollit per a la realització de l'aplicació ja que disposava de totes les eines que necessitàvem des de la creació del codi fins a l'encapsulament per a cada sistema operatiu.

Com ja s'ha comentat el projecte ha estat estructurat en tres fases: Anàlisi, disseny i implementació.
A la fase d'anàlisi s'han estudiat les diferents modalitats de joc que té les dames, s'ha escollit la més apropiada per a la realització del projecte i a més s'han analitzat diferents funcionalitats a afegir com podria ser la possibilitat de jugar amb més d'un dispositiu a la vegada.

Per al disseny és va realitzar un primer esbós a baix nivell, i a mesura que anava avançant el projecte s'han anat fent disseny més a alt nivell fins a arribar al disseny final realitzat amb eines pròpies per a aquest tipus de tasques com es l'Adobe Illustrator.

Paraules clau

Joc, Dames, Multijugador, HTML5, CSS, Javascript, Cross-Platform.

Notacions i Convencions

Tipus de text	Font	Mida	Decoració
Títol 1	Arial	20	Negreta
Títol 2	Arial	13	Negreta
Títol 3	Arial	10	Negreta i cursiva
Text normal	Arial	10	
Codi	Calibri	9	

Índex

Figures i taules

Llista d'imatges, taules, gràfics, diagrames, etc., numerades, amb títols i les pàgines en les quals apareixen.

Índex de figures

Índex de taules

Capítol 1: Introducció

1.1. Introducció/Prefaci

Aquest treball ha consistit en la creació d'un joc de dames multiplataforma utilitzant eines actuals i deixant de banda les natives de cada sistema operatiu, això vol dir la creació d'un joc per al sistema operatiu (IOS) o (Android) sense fer ús de les (API's) pròpies dels mateixos. D'aquesta manera amb un mateix codi obtenim el resultat per a diferents plataformes.

Aquesta decisió és presa degut a experiències de desenvolupament de jocs anterior en la qual es va intentar fer servir les eines natives i el resultat va ser la no creació d'aquest joc degut a problemes derivats de no poder ajustar bé les resolucions dels dispositius i eines complexes per a un fi senzill.

L'interès principal en la realització d'aquest projecte és el d'aplicar eines de programació web i dispositius mòbils apreses tant al màster com a món laboral i plasmar-ho en la creació d'un joc.

1.2. Descripció/Definició

El projecte comença en la voluntat de crear una aplicació multi dispositiu i amb l'interès de poder aplicar els coneixements de HTML5, CSS3 i Javascript apresos durant l'etapa d'estudiant d'aquest màster i els adquirits durant la trajectòria laboral.

Amb aquesta aplicació dues persones podran jugar al joc de dames amb un sol dispositiu i sentir gairebé la mateixa sensació com si estiguessin jugant amb un taulell tradicional.

La intenció principal d'aquesta aplicació és que es puguin jugar partides ràpides de dames sense la necessitat d'estar pensant ni de muntar i desmuntar el taulell tradicional i tampoc amb la necessitat de registrar-se en cap aplicació.

El fet que sigui multiplataforma és un tema molt interessant degut a que avui en dia a l'hora de realitzar qualsevol tipus d'aplicació mòbil és gairebé obligatori realitzar-ho per a les dues plataformes majoritàries que trobem avui dia al mercat (iOS i Android) i el fet de poder-ho realitzar amb un sol codi facilita les coses en el sentit d'una millora en el temps d'entrega de l'aplicació (no has de picar el codi de dues aplicacions) i posteriorment en la resolució dels problemes ja que una vegada es soluciona l'error es vàlid per a totes les plataformes.

1.3. Objectius generals

Com s'ha comentat al resum, l'objectiu d'aquest projectes és l'anàlisi, el disseny i la implementació d'un joc de dames per a dispositius mòbils, utilitzant eines gratuïtes i llibreries de codi obert per a facilitar el treball en la implementació del mateix i intentant emfatitzar en que el disseny no quedi antiquat de manera ràpida.

1.3.1 Objectius principals

Els objectius principals de l'aplicació són els següents:

- Poder jugar partides de dames modalitat angleses.
- Eficiència en la càrrega dels apartats.
- Disseny intuïtiu i actual

Els objectius per a l'usuari són molt similars al de l'aplicació, son els següents:

- Saber que en el seu dispositiu, per petit o gran que sigui, l'aplicació es veurà bé
- Poder fer una partida de dames ràpida sense haver de registrar-se a cap lloc i sense haver de configurar opcions, engegar i jugar o '*Plug & Play*'.

Els objectius personals, com autor d'aquest treball, són els següents:

- Creació d'una eina multiplataforma.
- Creació d'un projecte des de zero fins a l'entrega.
- Aprofundiment dels coneixements de programació i disseny responsiu.

1.3.2 Objectius secundaris

Els objectius secundaris d'aquesta aplicació son aquells objectius els quals no s'han pogut implementar en la data d'entrega del treball però que son importants de cares a la millora futura de l'aplicació:

- Poder jugar contra una intel·ligència artificial i poder escollir el nivell de la mateixa.
- Poder jugar dues persones amb diferents dispositius.
- Adaptar el disseny per a poder jugar en horitzontal.
- Poder llevar un control de les estadístiques de diferents partides jugades.

1.4. Metodologia i procés de treball

La metodologia i el procés de treball emprats per a la realització d'aquest treball ha sigut la típica d'un projecte tècnic i la separació de les fites en fases ben definides i separades ha estat clau. Una vegada seleccionada la temàtica de l'aplicació i del projecte els passos que s'han seguit per al desenvolupament han estat els següents:

1.4.1 Cerca de l'eina de desenvolupament

Per al desenvolupament d'una aplicació híbrida disposem de moltes aplicacions i de tots els colors que ens ofereixen múltiples funcionalitats en diferents llenguatges i entorns. Una vegada feta la recerca i de descartar algunes, les tres aplicacions que van quedar com a possibles solucions van ser, tal i com s'explica al resum, (Titanium, 2014), (PhoneGap) i (Intel XDK) seleccionant finalment aquesta última com a eina definitiva per al desenvolupament de l'aplicació.

1.4.2 Estudi de les modalitats de joc de dames

Una vegada conegut que l'aplicació seria un joc de dames es procedeix a l'estudi de les diferents de modalitats de joc que disposa aquest antic joc de taula, i es fa un estudi de les modalitats més populars i quina podria encaixar millor com a aplicació en el mercat d'aplicacions dels principals sistemes operatius.

1.4.3 Anàlisi de connectivitat entre dispositius

En un primer moment es va valorar l'opció de poder fer una partida amb diferents dispositius, i gran part del temps dedicat a l'anàlisi es va dedicar en aquesta tasca, i malgrat que en un moment va semblar que es podria implementar en un temps que ens permetés entregar el projecte a temps, finalment es va haver de descartar la idea i només implementar l'opció de jugar amb un sol dispositiu.

1.4.4 Disseny dels elements gràfics

A l'inici del projecte es va fer un disseny del que serien les diferents pantalles a baix nivell utilitzant l'aplicació (Balsamiq Mockups) però finalment, amb el pas del temps en el projecte s'ha anat ajustant i modificant el disseny quedant molt diferent a la idea inicial.

1.4.5 Implementació de l'aplicació

Per a realitzar la implementació de l'aplicació va caler un estudi exhaustiu de l'eina i de com realitzar les diferents funcionalitats que necessitaríem per a la realització de l'aplicació.

L'estudi d'aquesta eina ha consistit en mirar exemples d'aplicacions realitzades amb aquesta eina i que els propis desenvolupadors d'Intel ofereixen als usuaris d'Intel XDK.

Una vegada obtingut el coneixement necessari per a la creació de l'App només quedava ficar-se mans a l'obra i picar codi per tal d'implementar l'algorísmica del joc i l'ajust del disseny fet per tal de que es visualitzés correctament en la gran majoria de dispositius.

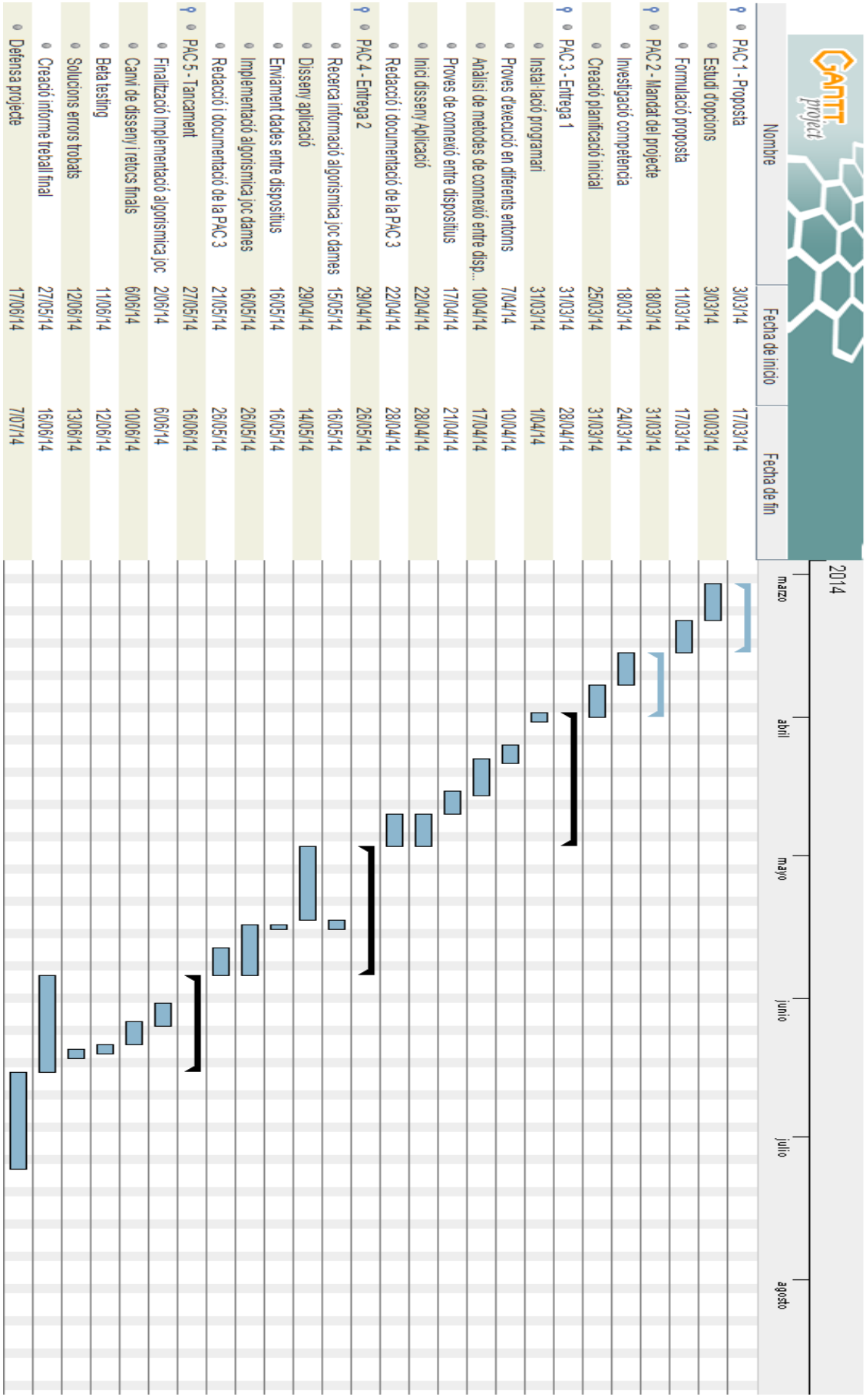
1.4.6 Proves finals

Per a la realització de les proves s'han fet servir dos mètodes:
- Utilitzar a gent aliena al projecte per a que fessin partides i comprovar que realment totes les funcionalitats han estat correctament implementades i que no hi ha cap error.
- Fer servir aplicacions online de joc de dames per generar moviments reals. És a dir, que una persona fes servir les fitxes marrons, fer aquest moviment en algun joc de dames online que tingui intel·ligència artificial per generar el moviment de les blanques i fer aquest moviment manual en l'aplicació.

1.5. Planificació

La planificació d'aquest projecte ha estat marcat per les entregues parcials que s'han anat fent durant el transcurs de l'assignatura del projecte final.

En la pàgina següent trobem el diagrama de Gantt referent a les diferents etapes del projecte. Aquesta etapes s'explicaran a continuació de manera més detallada degut a la qualitat de la imatge del diagrama complet.



Il·lustració 1 - Planificació final

1.5.1 Lliurament de la proposta

La fase de lliurament de la proposta va començar el 3 de març del 2014 just després de realitzar un debat amb tots els companys sobre possibles propostes.

Aquesta fase va durar dues setmanes i en ella s'havia de realitzar una proposta forma del que es voldria realitzar com a projecte final. De les coses que s'havien de fer en aquesta entrega destaquen la proposta d'un títol, un petit resum de la idea i fins i tot una motivació pel qual es volia realitzar aquest projecte i no l'altre.

Finalment la planificació d'aquesta entrega va quedar així:

Il·lustració 2 - Planificació primera entrega

1.5.2 Mandat del projecte

Aquesta fase va començar al dia següent de terminar la fase de lliurament de la proposta. Aquesta fase, per tant, va començar el 18 de març i tenia una durada d'uns 12 dies.

En aquesta fase tan curta el que es va realitzar va ser una investigació de la competència en els mercats d'iOS i d'Android, valorar els seus punts forts i febles de cares a no caure en els mateixos problemes i veure quins son els punts obligatoris que havia de tenir l'aplicació.

Una vegada estudiada la competència es va crear una planificació inicial de com s'havia d'estructurar el projecte per poder arribar a bon port. Aquesta planificació s'ha anat canviant a mesura que passaven els dies ja que com a tot projecte sempre surten problemes.

Il·lustració 3 - Planificació mandat del projecte

1.5.3 Primer lliurament parcial

L'etapa del primer lliurament parcial del projecte va començar l'1 d'abril i va acabar el 28 del mateix mes.
Durant aquesta etapa es va haver de realitzar una part important del projecte ja que equival aproximadament al 33% de la planificació temporal del projecte.

En aquesta etapa es va realitzar la instal·lació del programari i diferents proves de petites aplicacions en diferents entorns per comprovar que l'eina realment feia la seva funció i que la corba d'aprenentatge de la mateixa no era gaire pronunciada i que els dispositius s'adapten al contingut sense ser natiu.

En aquesta entrega es va malgastar molt de temps en la recerca d'informació relativa a la connexió entre dispositius, cosa que va fer enrederir i modificar la planificació ja que finalment aquesta funcionalitat no es va implementar, i també es va començar a fer un petit esbós del que seria l'aplicació (tot això a baix nivell, sense entrar en detall).

La planificació temporal d'aquesta fase va ser la següent:

PAC 3 - Entrega 1	31/03/14	28/04/14
Instal·lació programari	31/03/14	1/04/14
Proves d'execució en diferents entorns	7/04/14	10/04/14
Anàlisi de metodes de connexió entre disp...	10/04/14	17/04/14
Proves de connexió entre dispositius	17/04/14	21/04/14
Inici disseny Aplicació	22/04/14	28/04/14
Redacció i documentació de la PAC 3	22/04/14	28/04/14

Il·lustració 4 - Planificació primer lliurament

1.5.4 Segon lliurament parcial del projecte

Aquesta entrega també va tenir una durada aproximada d'un mes, que va des de el 29 d'abril fins al 26 de maig.

Es en aquesta fase a on es va haver d'acabar el disseny de l'aplicació que es va començar a la fase anterior, també va haver-hi una fase de documentació de les modalitats de joc, intent de enviament de dades entre dispositius i per últim la implementació de l'algorísmica.

En aquesta fase han hagut molts problemes, sobretot derivats en l'intent de enviament de dades entre dispositius, ja que com s'ha comentat en altres apartats de la planificació no ha estat possible i ha suposat una pèrdua de temps que es podria haver emprat en altres fases del projecte.

Altres problemes trobats en aquesta fase ha sigut amb el disseny de l'aplicació. La impossibilitat d'adaptar el disseny creat a baix nivell va suposar fer un canvi d'estil. Quan es va acabar aquest canvi de disseny i al fer proves reals amb dispositius es van trobar problemes amb la resolució dels textos i va fer canviar per tercer cop el disseny (aquest cop en la següent fase).

La planificació temporal d'aquesta fase va quedar així:

PAC 4 - Entrega 2	29/04/14	26/05/14
Recerca informació algorismica joc dames	15/05/14	16/05/14
Disseny aplicació	29/04/14	14/05/14
Enviament dades entre dispositius	16/05/14	16/05/14
Implementació algorismica joc dames	16/05/14	26/05/14
Redacció i documentació de la PAC 3	21/05/14	26/05/14

Il·lustració 5 - Planificació segon lliurament

1.5.5 Tancament

Aquesta última fase, de durada tres setmanes i que ha anat des del 27 de maig fins al 14 de juny ha suposat el tancament del projecte.

En aquesta suposadament i segons la planificació inicial el que s'havia de fer era els últims retocs a l'aplicació i fer una ultima recerca en cerca de possibles 'bugs' i solucionar-ho. Res més allunyat de la realitat.

En aquesta fase s'ha hagut d'acabar fites que s'han anat arrastrant d'altres lliuraments i ha comportat que en aquesta fase s'hagi hagut de refer per complet el disseny de l'aplicació, adaptar-ho mitjançant codi i terminar d'implementar l'algorísmica de lloc a més de totes les tasques pròpies d'aquesta entrega.

Finalment la planificació d'aquesta tasca ha estat la següent:

PAC 5 - Tancament	27/05/14	16/06/14
Finalització Implementació algorismica joc	2/06/14	6/06/14
Canvi de disseny i retocs finals	6/06/14	10/06/14
Beta testing	11/06/14	12/06/14
Solucions errors trobats	12/06/14	13/06/14
Creació informe treball final	27/05/14	16/06/14

Il·lustració 6 - Planificació lliurament final

1.6. Pressupost

Aquest projecte és i serà un projecte gratuït, però si s'hagués de monetitzar les hores emprades en la realització d'aquesta aplicació sortirien uns números semblants als següents:
Si es manca d'un pressupost fixat per un tercer o pel propi autor, s'haurà de facilitar una estimació del cost que suposaria (o ha suposat) el projecte.

Objecte	Nº de hores	Descripció	Preu unitari	Descompte	Total de línia
Programadors	30	Implementació de l'aplicació	20€/hora	0	600€
Dissenyadors	5	Disseny de l'aplicació	15€/hora	0	75€
				Subtotal	675
				Impuesto sobre ventas	21%
				Total	816.75

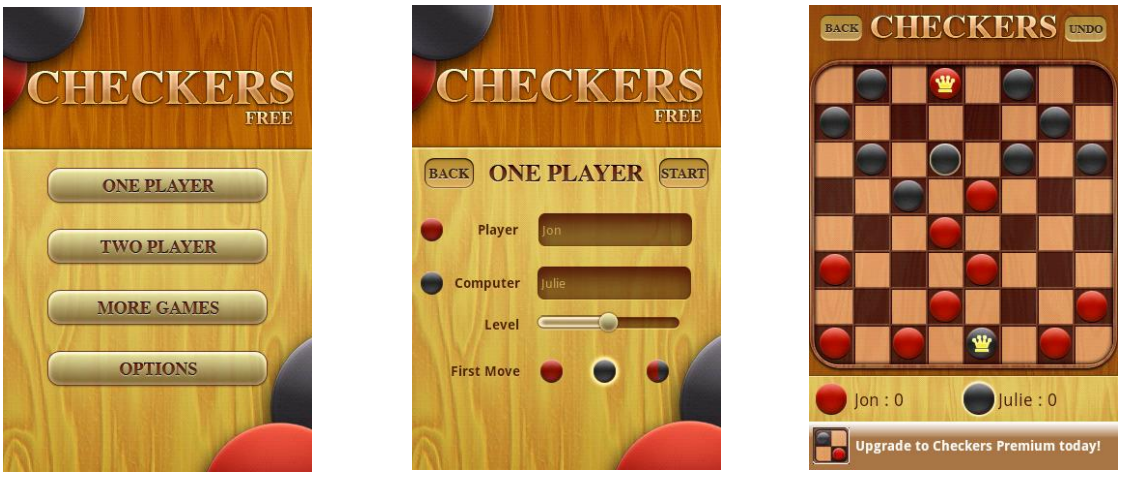
Capítol 2: Anàlisi

2.1. Estat de l'art

Per a la realització d'un bon projecte, si aquest no és del camp de la recerca (és a dir, estem creant un objecte o una temàtica la qual no existeix) com és aquest cas, és molt important saber quins altres projectes similars s'han fet, com s'han fet i en que es pot diferenciar el que es farà amb el que ja hi ha al mercat, és per això que s'ha fet una recerca de quines aplicacions hi ha actualment en matèria de jocs de dames per a aplicacions mòbils o web i quines característiques tenen. S'han classificat en funció del sistema operatiu al qual pertanyen per a un millor enteniment del que s'està explicant.

2.1.1 Jocs de dames a la play store d'android

Nom: Checkers Free
Companyia: Optime Software
Descripció: És el joc de dames més descarregat de la Play Store, amb un disseny molt clàssic i amb dos versions diferents: una versió gratuïta amb publicitat i un altre de pagament . Si t'endinses en el joc es pot veure que hi ha dos tipus de modes de joc, un per a jugar sol contra la intel·ligència artificial de la qual pots escollir diferents nivells de dificultat, o multijugador amb un sol dispositiu. Després disposa de dos opcions la qual una es treure el so i l'altre es utilitzar unes regles per als salts a l'hora de menjar la peça del contrari o no.
Comentaris: El funcionament de l'aplicació es prou correcte, i el disseny de l'apartat del joc és molt visual. La part dels menús i els botons son molt clàssics i en pantalles relativament grans (provat en un Nexus 5 amb pantalla de 4.95") els botons es veuen molt més petits que a les captures i a vegades es fan difícils de polsar. El multijugador és amb un sol dispositiu.
Captures:



Il·lustració 7 - Captures del Joc Checkers Free

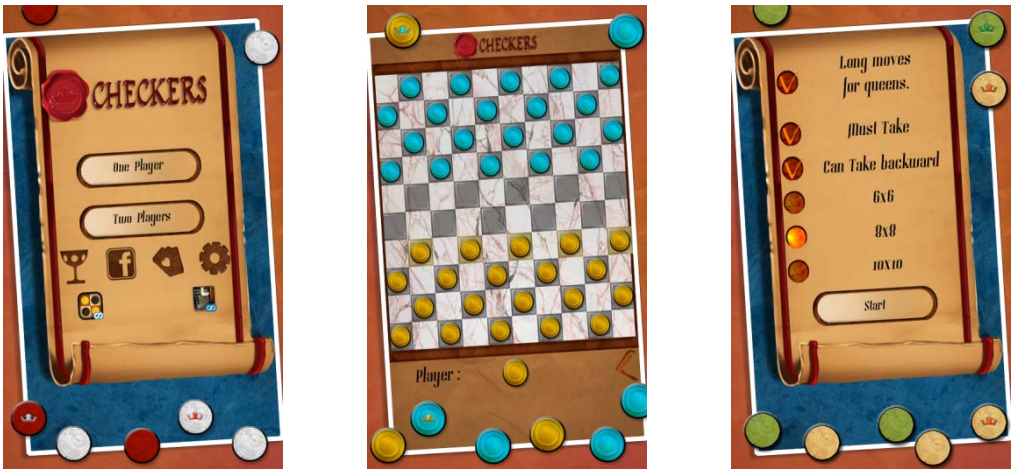
Nom: Damas (Checkers)
Companyia: Magma Mobile

Descripció: Es troba a la segona posició de la llista quan cerquem 'checkers' a la Play Store, disposa d'un disseny molt més elaborat que el de la companyia Optime i en aquest cas només es disposa d'una versió gratuïta amb publicitat (molta publicitat) .

Dins del joc es pot veure que té moltes més possibilitats tant de regles, disseny de taulers o sons. Al igual que l'anterior comentat disposa de dos modes de joc, individual o multijugador (amb un sol dispositiu).

Comentaris: Les dimensions dels botons son prou grans com per a poder clicar sense problemes. L'aplicació detecta automàticament l'idioma del dispositiu, disseny molt temàtic i amb possibilitat de canviar els colors del tauler i les fitxes, i possibilitat de jugar amb molts tipus de regles diferents (internacional, americana, personalitzada...).

Captures:



Il·lustració 8 - Captures de l'aplicació Damas (Checkers)

2.1.2 Jocs de dames a l'App Store d'Apple

Nom: Checkers

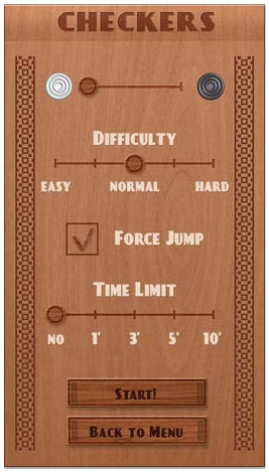
Companyia: Byterun

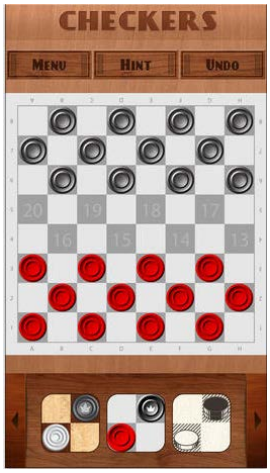
Descripció: Primer resultat que trobem quan cerquem checkers a l'App Store. El disseny de l'aplicació es molt semblant a la resta d'aplicacions, però en línies generals és molt uniforme i està molt aconseguit.

En aquest cas només disposa d'una versió gratuïta i sense publicitat fent que la interfície del joc quedi molt neta. Disposa de les mateixes modalitats de joc que la resta de jocs comentats, versió per a un jugador per a jugar contra la IA del joc, o possibilitat de jugar dos jugadors amb un mateix dispositiu.

Comentaris: Disposa d'opcions que no s'havien vist en les altres aplicacions com la gestió del temps entre moviments, que l'aplicació et proporciona consells sobre quin moviment es pot realitzar. En línies generals el disseny està mes aconseguit que als altres.

Captures:





Nom:

Il·lustració 9 - Captures del joc Checkers

Nom:Damas españolas

Companyia: Itchigoo

Descripció: Joc de dames de l'empresa Itchigoo, el qual ha creat una aplicació per a cada tipus de dames existents (espanyoles, internacional,...). La interfície és molt simple i les opcions estan amagades al començament. Disposa de tres modalitats de joc, 1 jugador contra la intel·ligència artificial, dos jugadors amb un mateix dispositiu o cercar un altre jugador a traves del Game Center d'Apple

Comentaris: No és dels mes 'usables' ja que té els menús molt amagats i molt malament distribuïts però és un dels més complerts en quant a normes i opcions de joc es refereix. A més, ofereix la opció de moure la fitxa movent-la amb el dit (arrastrant) cosa que no fa cap dels anteriors.

Captures:

Il·lustració 10 - Captures del joc Damas Españolas

Com a conclusió es pot observar que no hi ha cap aplicació que ho reuneix tot, interfície amigable, possibilitat de poder jugar multijugador en diferents dispositius, arrastrar la fitxa, etc... i aquí es on entra l'objectiu d'aquesta aplicació, aunar tots aquests conceptes en una única aplicació i que a més sigui multi plataforma.

2.2. Objectius i Abast

Els objectius d'aquest projecte és senzill, és realitzar el joc de dames com a una aplicació que pugui ser executada en maquines amb diferents sistemes operatius, majoritàriament mòbils i amb sistemes operatius IOS o Android (entre d'altres) a més de poder-se executar via web.

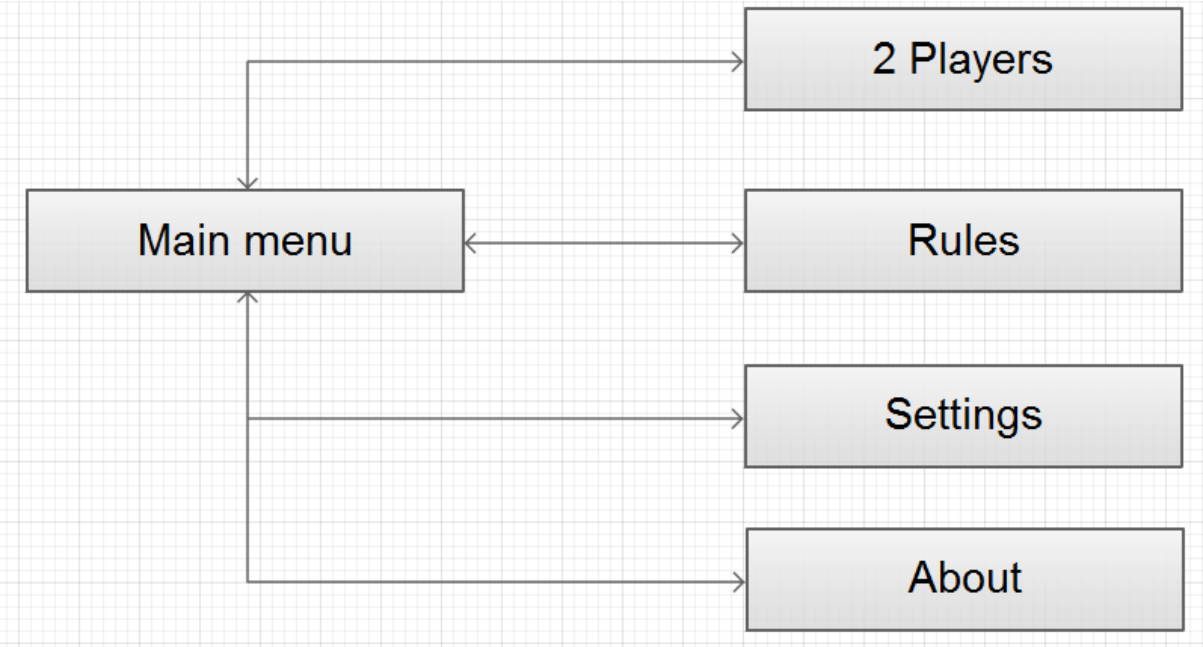
Per tant amb aquesta aplicació es podrà:

- Jugar a les dames amb un dispositius IOS.
- Jugar a les dames amb un dispositius Android.
- Jugar a les dames amb un navegador web.

Capítol 3: Disseny

3.1. Arquitectura de la informació i diagrames de navegació

L'aplicació en si es un joc molt senzill sense gaires pantalles i la navegació entre elles no pot comportar cap problemes però si considero important la creació d'un diagrama de navegació per il·lustrar els possibles moviments que pot realitzar un usuari dins del joc.



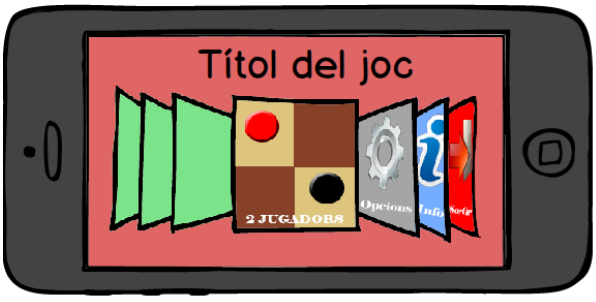
Il·lustració 11 - Diagrama de navegació de l'aplicació

Bàsicament l'aplicació consta d'un menú principal amb quatre opcions i des de aquestes opcions podem fer les accions corresponents i tornar enrere.

3.2. Disseny gràfic i interfícies

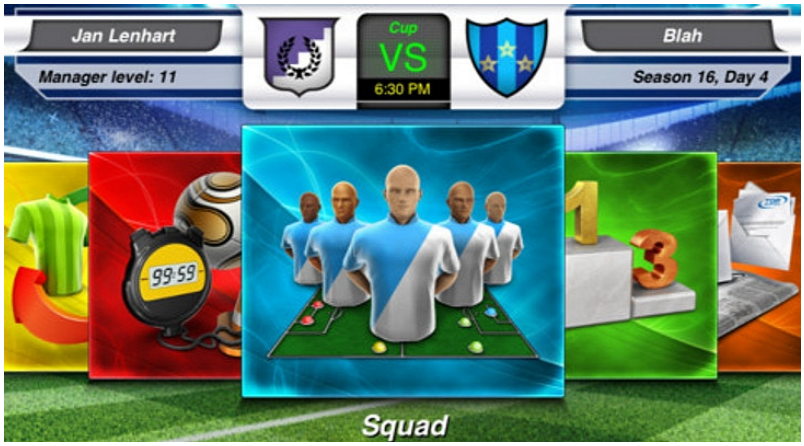
3.2.1 Disseny del menú principal

La idea inicial del projecte era realitzar un disseny de menús similar a la idea que va crear Apple anomenada (Cover Flow) i així es va intentar fer en l'esbós a baix nivell creat, tal i com es pot veure a les següents figures:



Il·lustració 12 - Menú principal cover flow

Aquest disseny està present en alguns jocs actuals com per exemple el famós (Top Eleven Be A Football Manager) i que es pot veure una imatge en la següent figura:



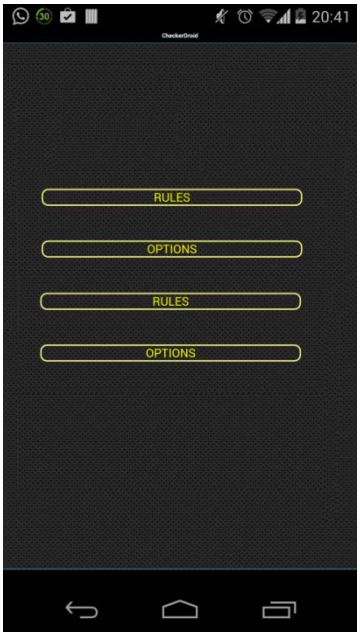
Il·lustració 13 - Menú Cover Flow al joc Top Eleven

Una vegada arribat el moment de la implementació del disseny des de l'aplicatiu d'Intel es va comprovar que no era viable, tant per temps d'implementació com per importància dins del projecte i es va crear un més senzill.

La següent versió que es realitza es la següent:

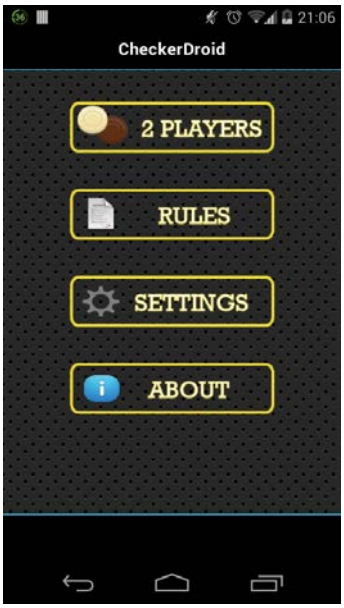
Il·lustració 14 - Segon menú alternatiu

Menú prou acceptable i funcional, el problema que va sorgir va ser que així era com es visualitzava quan ens trobavem al mode debug de l'intel XDK, quan fèiem servir la versió test que suposadament és la versió de com es veurà una vegada generéssim l'APK o l'IPSW es veia com a la següent figura:



Il·lustració 15 - Problemes amb el segon Menú

Es pot observar com els requadres son molt més petits, la capçalera no es pot ni llegir i l'aplicació en si perd molta qualitat. Per tant es decideix fer un tercer canvi en el disseny i realitzar-ho amb imatges i ajustar-les via CSS per tal de que es pogués veure bé en qualsevol moment i el resultat final va ser el següent:

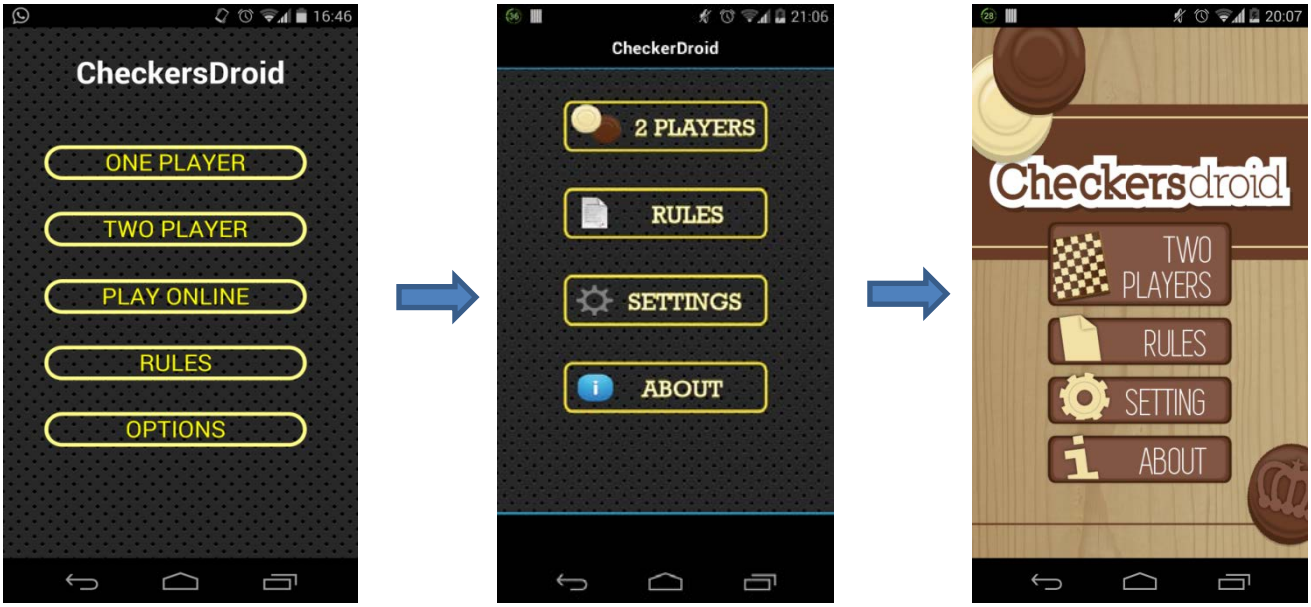


Aquest menú era millor que el primer però no acabava de tenir ni la serietat d'una aplicació ni la plasticitat d'un joc i per tant es decideix canviar per tercera i ultima vegada el disseny del menú de l'aplicació i es crea mitjançant l'aplicació (Adobe Illustrator).

Finalment el resultat és el següent:

Il·lustració 16 - Versió final del menú

L'històric de versions del menú és aquest:

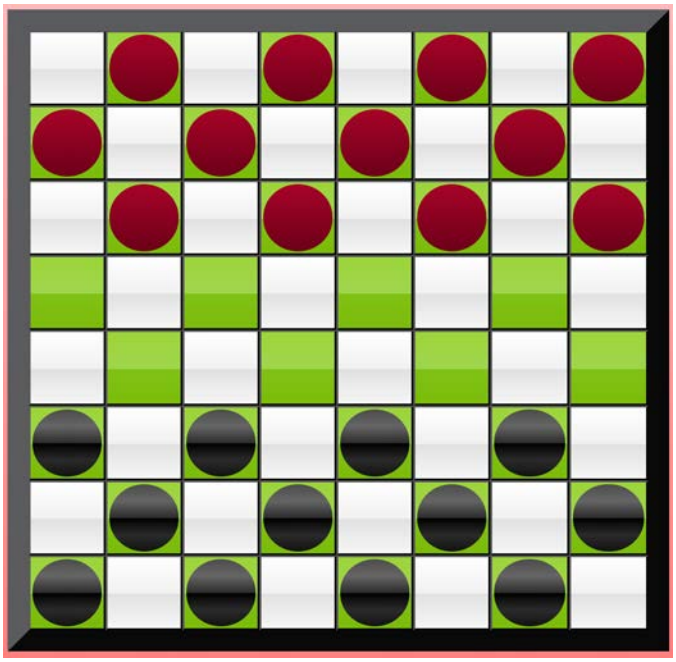


Il·lustració 17 - Evolució del menú principal

3.2.2 Disseny del taulell i peces

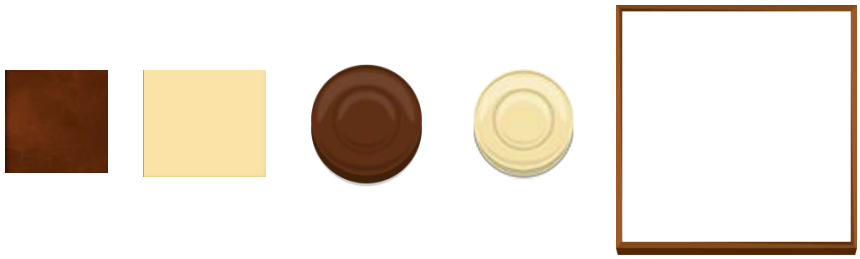
El taulell no va ser dissenyat a baix nivell en entregues anteriors i per tant es va procedir al disseny una vegada es va començar a implementar a dissenyar l'aplicació.

Primerament, es va fer un primer disseny en paral·lel mentre s'anaven aplicant l'algorísmica bàsica del joc per tal de provar el bon funcionament d'aquest. Aquest disseny tenia una vista similar al de la següent imatge:



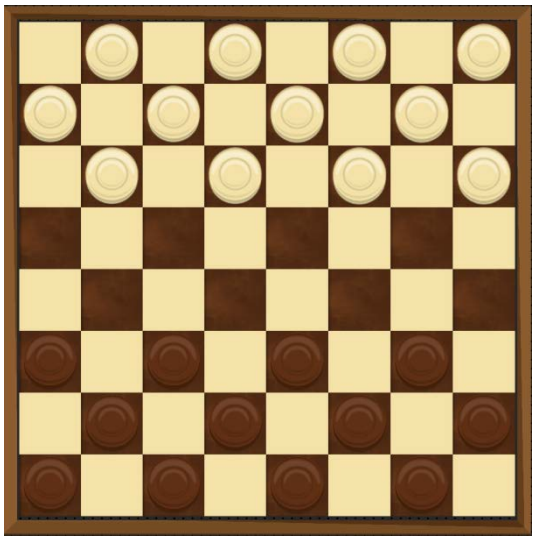
Il·lustració 18 - Primera versió del taulell

Si bé era un taulell aparent i funcional, no donava una sensació ‘seria’. Es per això que es va procedir a la realització d’unes peces, un quadrat groc, un marró i un marc amb (Adobe Illustrator)



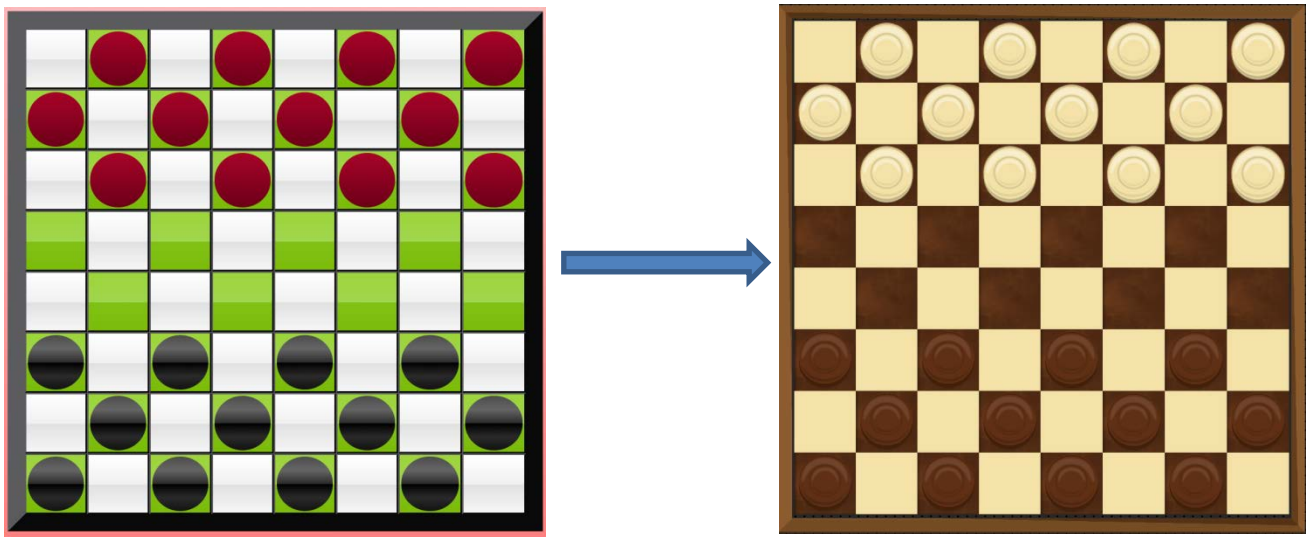
Il·lustració 19 - Contingut i peces del taulell

El resultat obtingut és a la figura següent:



Il·lustració 20 - Versió final del taulell

Per tant, l'evolució del taulell durant el transcurs del projecte ha estat el següent:



Il·lustració 21 - Evolució del taulell de joc

3.3 Estils

A l'hora de crear l'estil de l'aplicació s'ha buscat complir dues premisses principals:

- Crear uniformitat entre totes les pantalles del joc i,
- Donar-li un estil '*Retro*'.

Per fer això s'han fet un seguit d'accions:

3.3.1 Fons de pantalla

Es va crear una textura molt similar a la fusta per a utilitzar-la com a fons en la majoria de pantalles, aquesta textura és la següent:

Il·lustració 22 - Textura de fusta

3.3.2 Logotip

Creació d'un logotip per fer-se servir tant a la pantalla de benvinguda com per al menú principal, el logotip és el següent:

Il·lustració 23 - Logotip del joc

3.3.3 Botons

En quant a botons, s'han intentat fer botons aprofitant les eines multiplataforma que ofereix l'Intel XDK i en colors indicatius del tipus de tasca que es realitzarà. És a dir, per a guardar botons de color verd o per rendir-se de la partida un color vermell.
Algun exemple de botó:

Il·lustració 24 - Exemples de botons

3. 4. Llenguatges de programació i APIs utilitzades

El llenguatge a utilitzar no va portar cap mena de dubte. Si el que es volia realitzar era una aplicació híbrida i deixar de banda els llenguatges natius de les principals plataformes (Objective C, Java, C#...) s'opta per utilitzar els llenguatges de programació web per a la tasca de crear l'aplicació mòbil híbrida. Aquests llenguatges no son altres que la combinació de HTML5, CSS3 i Javascript.

Després de fer un estudi de les diferents solucions que permeten realitzar aquest tipus d'aplicacions, trobem tres solucions que ens permeten realitzar aquestes aplicacions amb garanties, aquestes són: (Titanium, 2014), (PhoneGap) i la escollida (Intel XDK) de l'empresa Intel.

A continuació s'explica els motius pels quals no s'han escollit uns i perquè s'ha escollit l'altre:
En el moment de fer la recerca de Titanium s'observa que, si bé sembla una eina molt potent per a la realització d'aquest tipus d'apps, es treballa en un llenguatge anomenat NodeJS, la qual cosa faria perdre temps en aprendre l'estructura bàsica de com funciona aquest llenguatge. A més, el mòdul principal de Titanium es de pagament.

La segona opció va ser PhoneGap, programari amb molt bona documentació, moltes opcions gratuïtes ja implementades i també treballant amb llenguatges coneguts (HTML5,CSS i Javascript). Però finalment la interfície de treball i la manera de realitzar les tasques no va ser gaire agradables i es va decidir continuar explorant possibles solucions.

La selecció d' (Intel XDK) és degut a que segons el que s'ha pogut llegir a la documentació conté tot el necessari per a la correcta implementació d'aquest projecte i a més té una interfície amigable, té una corba d'aprenentatge petita i a més incorpora totes les funcionalitats que es necessiten durant totes les fases del projecte: Escriure el codi, emular diferents dispositius de diferents sistemes operatius, fer proves amb dispositius reals (si disposo d'ells), fer '*debug*', generar l'aplicació definitiva o fins i tot si vull monetitzar l'aplicació.

L'intel XDK és molt fàcil de fer servir, com a mostra tenim aquests primers passos per comprovar que, en molt poc temps ja es pot començar a treballar amb ell.

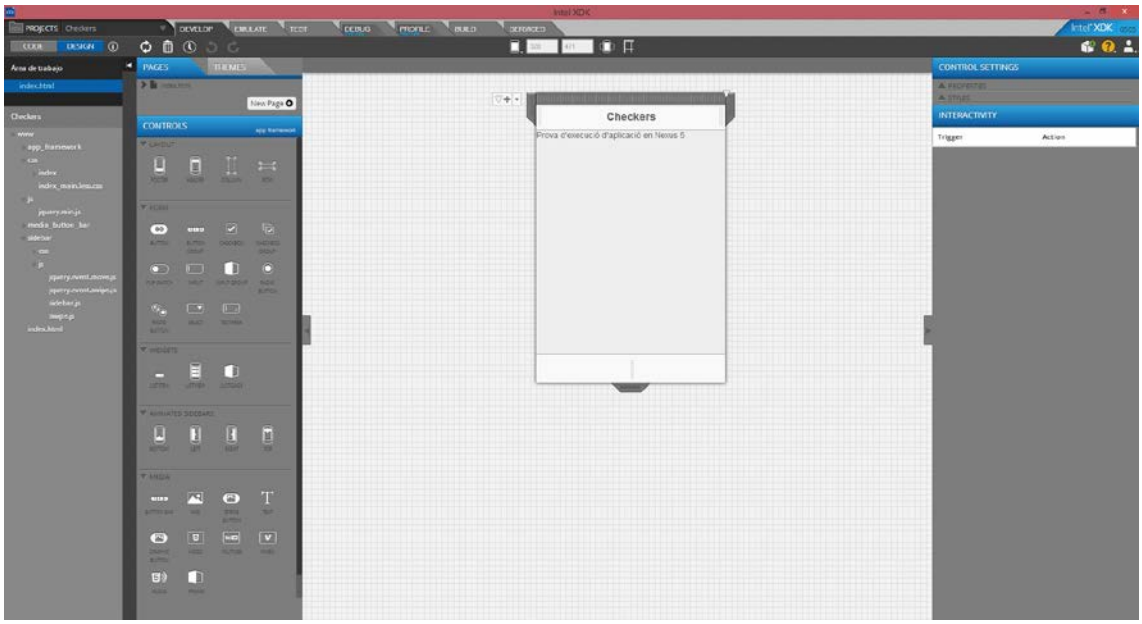
3.4.1 Instal·lació de l'Intel XDK

L'entorn de desenvolupament es el XDK d'intel, i la seva instal·lació i registre és molt fàcil i és com qualsevol programari.

Es realitzen diferents proves sense estressar gaire al programa i en algunes ocasions dona un error que fa que s'hagi de reiniciar. A priori no és greu perquè retorna al seu estat inicial però si és una mica molest pel fet d'haver de tornar a reiniciar el programa. Tot i que aquest error passa molt poc sovint, amb les posteriors actualitzacions del programari es va solucionar.

El programa conté tot el necessari per a la realització de la part de desenvolupament del treball ja que des de dins incorpora totes les funcionalitats necessàries, tant les de desenvolupament com posterior publicació, tal i com es pot observar en la següent captura:

Es pot observar com a la part superior disposa de diferents pestanyes en funció de la tasca a realitzar, des de desenvolupament com l'empaquetament en format apk o ipsw.

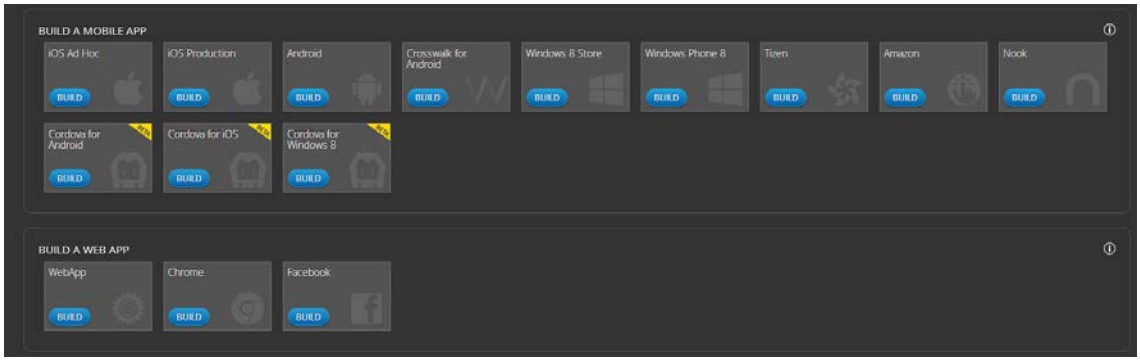


Il·lustració 25 - Pantalla principal d'Intel XDK

3.4.2. Proves en diferents entorns

Com comentàvem abans, amb un mateix codi es pot generar l'aplicació per a diferents plataformes des de IOS o Android fins a una web app.

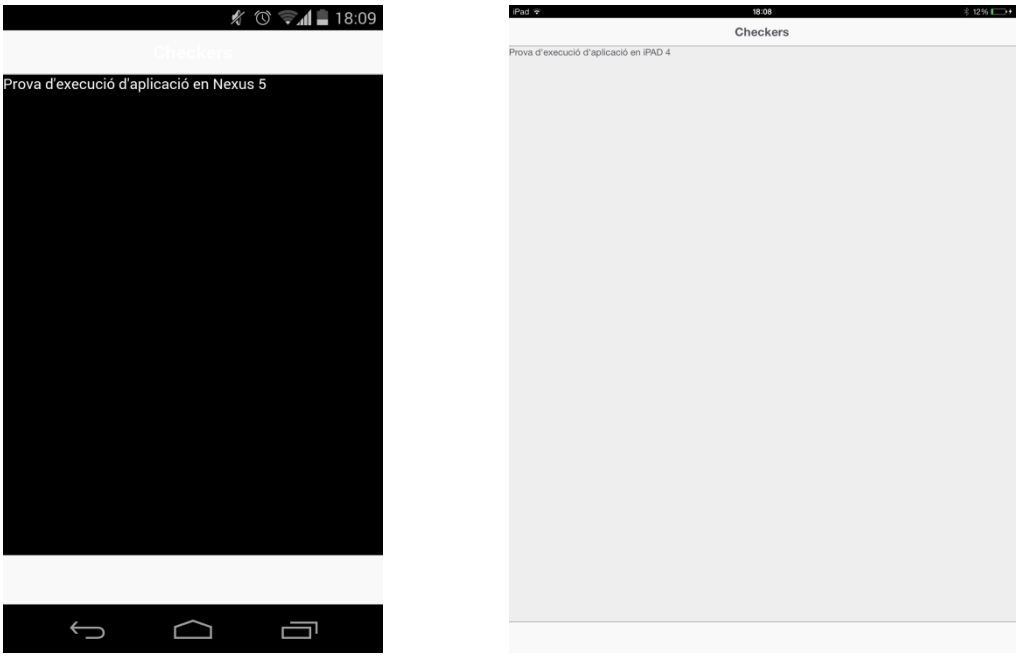
Les opcions son les següents:



Il·lustració 26 - Opcions de desplegament d'aplicació a Intel XDK

Les proves realitzades han sigut el típic '*Hello World*' i no s'ha generat una aplicació com a tal, sinó que s'ha realitzat a través de l'aplicació que té Intel per aquests casos.

Aquesta app s'anomena Intel App Preview i la trobem a (XDK App Preview a l'App Store) com (XDK a la Play Store). Es tan fàcil com fer login amb les teves dades d'Intel a l'aplicació, escanejar un codi QR amb el mòbil i ja s'instal·la l'aplicació al mòbil. Com a resultats es poden observar dos proves: una en un Google Nexus 5 i una altre en un iPAD4.



Il·lustració 27 - Exemples d'aplicacions creades

Es pot veure com amb un mateix codi s'han generat dos aplicacions totalment distintes i per a dos plataformes diferents.

Una vegada dins del projecte, amb (Intel XDK) es va descobrir una documentació d'un (Framework) propi per a poder treballar les interfícies de manera ràpida i senzilla i que funciona per a tots els

sistemes operatius, aquest Framework s'anomena (App Framework UI) i és aquí on es troba tota la documentació necessària per a realitzar tant una interfície adaptativa com comandes per a fer servir el hardware dels dispositius (com per exemple activar la vibració.

Bàsicament l'estructura del software i hardware final seria una cosa similar al diagrama següent:

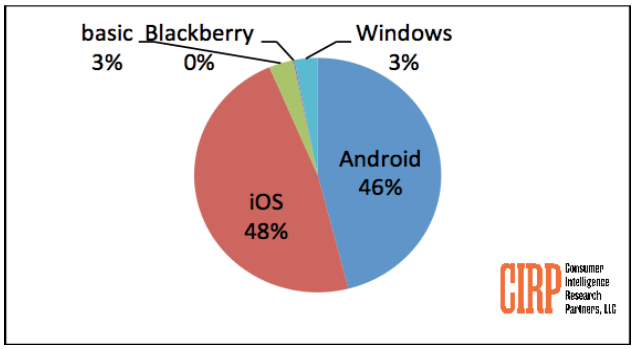
Il·lustració 28 - Estructura software + Hardware

Capítol 4: Implementació

Durant els capítols anteriors, s'ha anat veient com anava quedant l'aplicació a través de captures de pantalla i de petits fragment com el taulell o les peces del joc. És, en aquest moment de la memòria, on es veurà realment com ha quedat tota l'aplicació i desgranarem secció per secció per un millor enteniment de tot el procés que ha hagut fins arribar al final del camí.

4.1. Requisits d'instal·lació

Els requisits basics per a poder instal·lar l'aplicació es disposar d'un dispositiu amb Sistema Operatiu mòbil del tipus iOS, Android o Windows Phone, cosa molt senzilla si mirem el següent gràfic i comprovem la quota que tenen aquests sistemes operatius enfront a la resta.



Il·lustració 29 - Quota de mercat mòbil, font: http://bgr.com/2014/01/30/blackberry-us-market-share/

Com es pot comprovar, el 97% dels mòbils actuals disposen d'un d'aquests tres sistemes operatius per tant es molt probable que l'usuari final pugui instal·lar amb relativa facilitat el joc.

4.2. Instruccions d'instal·lació

En aquest cas es tracta d'una aplicació mòbil, per tant el que s'hauria de fer és anar al '*market*' de cada sistema operatiu i després cercar pel nom de l'aplicació i instal·lar-la al dispositiu en qüestió, com als exemples de la figura següent:

És un procés molt senzill ja que amb dos clics ja es té l'aplicació instal·lada i llesta per fer servir.

4.3. Exemple d'execució

Tot i ser un joc amb molt pocs apartats, es necessari mitjançant captures de pantalla i petites explicacions les transicions i possibles accions a realitzar dins de l'aplicació.

4.3.1 Menu principal

Menu principal amb quatre opcions amb disseny amb imatges vectorials per a una millor adaptació del contingut a les diferents pantalles i resolucions.

Tal i com s'ha comentat en apartats anteriors, s'han volgut recalcar dos aspectes fonamentals en el disseny de l'aplicació, aquests dos són:

- Mantenir l'aspecte clàssic de les dames amb textura tipus fustes i fitxes clàssiques i,
- Donar-li un toc modern amb fonts actuals i imatges de l'època.

Les quatre opcions que trobem i que explicarem més tard són:

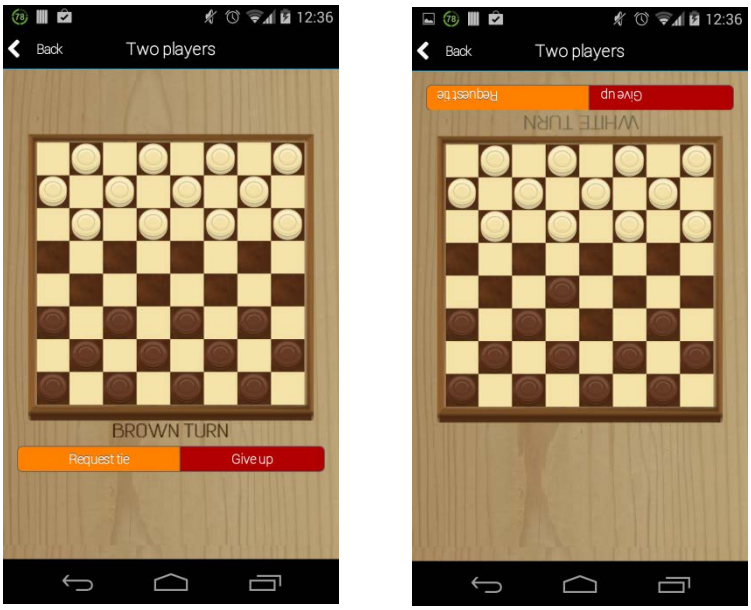
1. Two players: Pantalla principal del joc, es on es jugarà la partida de dames.
2. Rules: Pantalla on s'expliquen les regles principals del joc de dames i de la modalitat anglesa en concret.
3. Settings: Per canviar opcions bàsiques del joc.
4. About: Historial dels canvis realitzats en les diferents versions del joc

Il·lustració 31 - Pantalla principal del joc

Com es pot comprovar a aquesta captura i a les següents, l'idioma escollit per a la creació del joc ha sigut l'anglès, per així poder arribar a més públic en una primera versió inicial encara que no es descarta la traducció al Català i al Castellà en properes versions del joc

4.3.2 Pantalla de Two Players

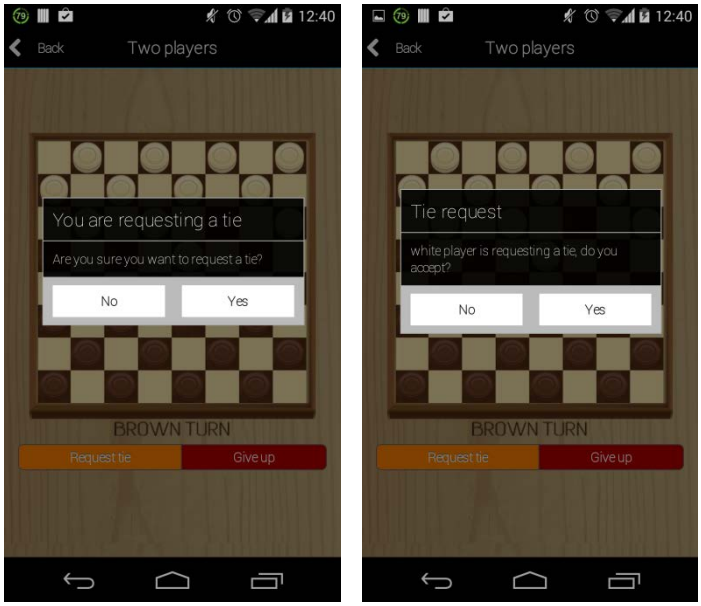
És la pantalla principal del joc tal i com s'ha comentat en apartats anteriors. En aquesta pantalla trobem el taulell al centre i en funció del torn uns botons a la part superior o inferior per demanar l'empat al jugador contrari o simplement per rendir-se si la partida va molt malament.



Il·lustració 32 - Captures pantalla two players

El text de la segona captura està del revés degut a que es vol provocar que cada jugador jugui a una part del taulell com si fos un taulell de veritat i no per torns en la mateixa posició. D'aquesta manera se li dona més realisme al joc.

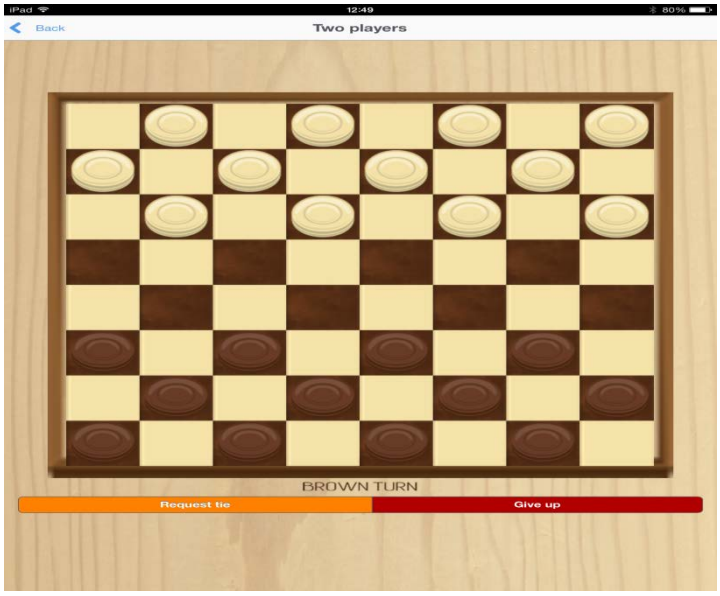
Des d'aquesta mateixa pantalla, tal i com es comentava es pot demanar un empat a l'altre usuari o directament rendir-se, tal i com es pot veure a les captures següents:



Il·lustració 33 - Alertes per demanar empat

Il·lustració 34 - Alerta per rendirse

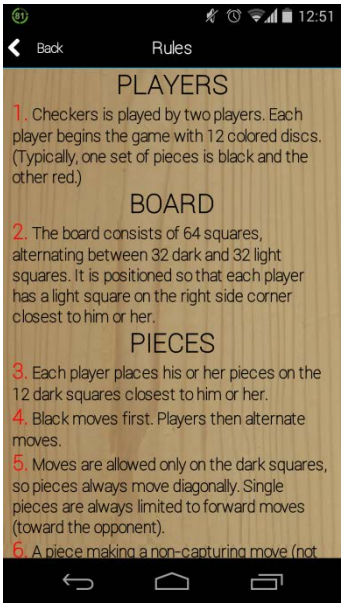
Altres captures per corroborar que l'aplicació funciona en dispositius molt diferents és la següent, captura realitzada en un iPad4 (les anteriors corresponen a un Nexus 5).



Il·lustració 35 - Captura de la pantalla two players a iPad

4.3.3 Pantalla Rules

Pantalla molt senzilla on s'enumeren les principals regles a seguir per a poder jugar correctament al joc.



Il·lustració 36 - Captura de la pantalla rules

4.3.4. Pantalla Settings

Pantalla per configurar 3 opcions molt bàsiques del joc, actualment es pot habilitar/deshabilitar els sons, habilitar/deshabilitar la vibració o seleccionar si és obligatori menjar la peça del contrari o no.
Aquesta pantalla està pensada per a versions futures anar afegint noves opcions, com poder seleccionar el color de les peces, qui comença primer, etc...

Il·lustració 37 - Pantalla Settings

4.3.5 Pantalla About

Pantalla que s'utilitzarà per anar actualitzant en ella totes les noves funcionalitats que es vagin afegint al joc, actualment, i a mode de broma s'explica que a la versió 1.0 ja es pot jugar a les dames:

Il·lustració 38 - Captura pantalla about

Capítol 5: Demostració

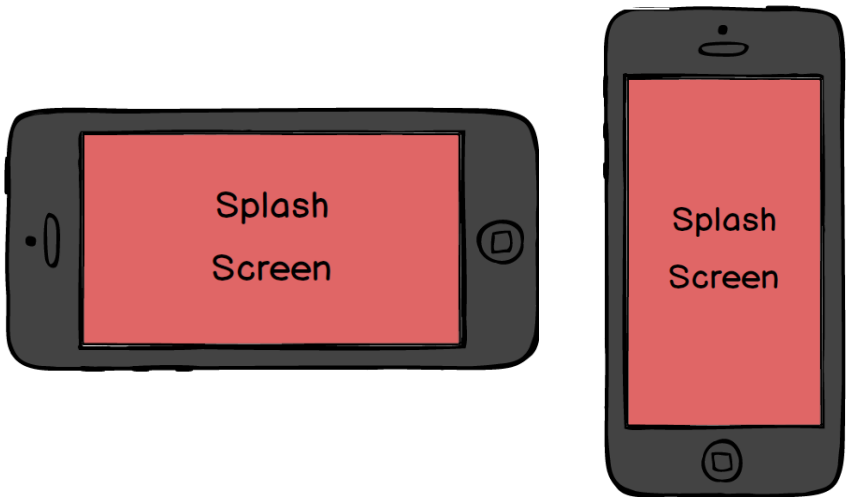
5.1. Prototips

Aquests van ser els prototips creats en començar el projecte i que han servit d'ajuda per a la realització final del disseny, encara que el resultat final no es ben bé igual al projectat als prototips

5.1.1 Prototips Lo-Fi

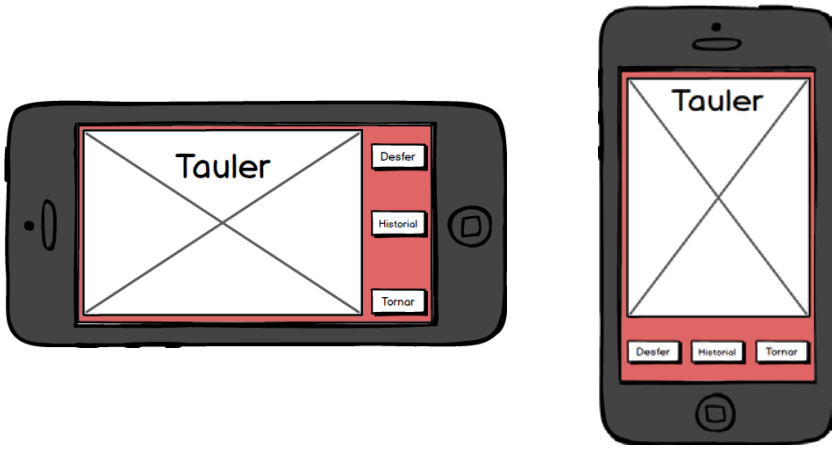
Es va realitzar un prototipus a baix nivell de l'aplicació en horitzontal i en vertical, intentant donar un disseny modern a un clàssic com son les dames. Finalment es va decidir eliminar la versió horitzontal del joc degut a problemes amb resolucions.

El disseny ha sigut realitzat amb Balsamiq Mockups, i és el següent:



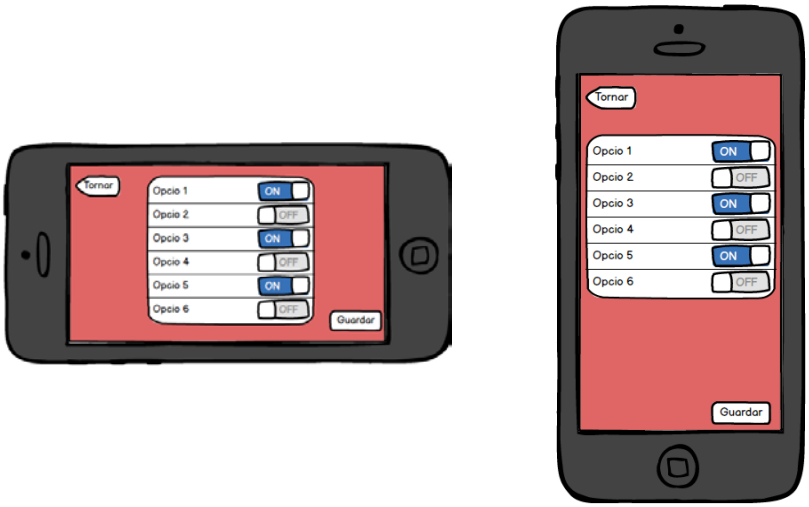
Il·lustració 39 - Prototipus de pantalla de benvinguda

Il·lustració 40 - Prototipus de menú principal



Il·lustració 41 - Prototipus de pantalla del taulell

En aquestes dos es pot comprovar com ja es valorava des d'un principi l'opció d'incloure botons però no es valorava l'opció de jugar dues persones amb un mateix dispositiu i es per això que la distribució final va canviar substancialment.



Il·lustració 42 - Prototipus de la pantalla d'opcions

5.2. Tests

Els tests realitzats per a la millora del joc i la solució de problemes, i tal i com s'ha explicat en altres apartats de la memòria, han consistit en dos tipus de tests:

1. Jugar dues persones amb el dispositiu amb l'ànim de detectar possibles errors.
2. Jugar una persona amb el dispositiu i el segon jugador que sigui el moviment generat per la intel·ligencia artificial d'un altre joc de dames d'internet. En aquest cas s'ha fet servir la I.A. de

Han sigut un testos molt satisfactoris ja que ens ha permès trobar errors que treballant i fent proves normals no s'haguessin trobat i a més es veia a l'aplicació com responia a una partida normal.

Amb això hem pogut comprovar que en un primer moment l'aplicació no marcava que la partida havia acabat degut a que un dels jugadors no podia fer més moviments (totes les fitxes estaven bloquejades) o que en segons quin moviments les peces quedaven solapades (dues fitxes en una mateixa posició).

Capítol 6: Conclusions i línies de futur

6.1. Conclusions

Les conclusions generals sobre aquest treball són la satisfacció de realitzar un projecte de principi a fi, passant per totes les etapes de manera autònoma. Des de trobar la idea fins l'anàlisi o la implementació ha sigut obra meva.

Per altra banda he après a planificar un projecte, ha tenir en compte que poden sorgir problemes que et facin canviar la idea inicial que tenies però que amb ganes i treball es pot fer.

En quant a l'assoliment dels objectius es podria dir que no estic content al 100% potser perquè vaig ser massa optimista amb la feina a fer tenint en compte el temps disponible i s'han hagut de fer canvis de rumb per tal d'arribar a bon port, tot i així son experiències que t'ajuden a millorar en projectes futurs.

El seguiment de la planificació ha estat molt difícil de complir, entre altres coses, perquè hi havia tres assignatures més del màster i s'havia d'anar a treballar tot i així considero que el seguiment de la planificació ha sigut prou correcte.

En línies generals estic molt content amb el resultat obtingut, he après moltes coses relacionades amb la meva feina i amb els meus interessos i espero que aquest projecte sigui el primer de molts.

6.2. Línies de futur

Aquest projecte no acaba en l'entrega del projecte, aquest projecte té moltes coses a realitzar-se en temps futurs i entre ells, destaco les següents:

1. Poder jugar contra una intel·ligència artificial.
2. Poder jugar multijugador amb diferents dispositius.
3. Poder jugar amb la pantalla tant en vertical com horitzontal.
4. Monetitzar l'aplicació si tingués èxit als Markets.

Bibliografia

Adobe Illustrator. (sense data). Recollit de http://www.adobe.com/es/products/illustrator.html

Android. (sense data). Recollit de http://es.wikipedia.org/wiki/Android

API's. (sense data). Recollit de http://es.wikipedia.org/wiki/API

App Framework UI. (sense data). Recollit de http://app-framework-software.intel.com/documentation.php#afui/afui_about

Balsamiq Mockups. (sense data). Recollit de http://balsamiq.com/

Checkerschest. (sense data). Recollit de http://www.checkerschest.com/checkers/

Cover Flow. (sense data). Recollit de http://es.wikipedia.org/wiki/Cover_Flow

Framework. (sense data). Recollit de http://es.wikipedia.org/wiki/Framework

Intel XDK. (sense data). Recollit de http://xdk-software.intel.com/

IOS. (sense data). Recollit de http://es.wikipedia.org/wiki/IOS

PhoneGap. (sense data). Recollit de http://phonegap.com/

Titanium. (2014). Recollit de http://www.appcelerator.com/titanium/

Top Eleven Be A Football Manager. (sense data). Recollit de http://www.topeleven.com/es/

XDK a la Play Store. (sense data). Recollit de https://play.google.com/store/apps/details?id=com.intel.html5tools.apppreview&hl=es

XDK App Preview a l'App Store. (sense data). Recollit de https://itunes.apple.com/us/app/intel-app-preview/id725023841?mt=8)

Annexos

Annex A: Codi font (Extractes)

A continuació es troba la majoria del codi comentat, utilitzat per a la creació de l'algorísmica del joc.

Aquí tenim les variables globals utilitzades durant el transcurs del joc:

```
/* Global vars
* whitePieces - White pieces array
* brownPieces - Brown pieces array
* BROWN - Constant for check if is brown turn
* WHITE - Constant for check if is white turn
* turn - Var for check actual turn
*/
var whitePieces = [];
var brownPieces = [];
var BROWN = 1;
var WHITE = 0;
var turn = BROWN;
```

Tenim un array per guardar les peces blanques i marrons, cada peça tindrà 4 atributs:

1. pieceNumber: número de la peça.
2. x: Posició x al taulell
3. y: Posició y al taulell
4. isChecker: Per saber si aquesta peça és una dama o no.

Després tenim dos constants i una variable, les constants equivalen a marró o blanc per més tard fer comparacions sobre quin és el torn actual i poder fer-ho amb un literal és més visual que fer-ho amb números. Per últim la variable turn que ens indica el torn actual de la partida.

Funció addPieces

```
/*
* addPieces() - Method for addPieces in board
*/
function addPieces(){
    var numWhitePieces = 1;
    var numBrownPieces = 1;
    for(var i=1;i<4;i++){
        for(var j=1;j<9;j++){
            if($('#cell'+i+j).hasClass('green')){
                // Create one White piece, assign id, add piece to white pieces array
and add onclick event
                var pieceDiv = document.createElement('div');
                $(pieceDiv).addClass('piece');
                var piece= document.createElement('a');
                $(piece).addClass('whitePiece');
                $(piece).attr('id','whitePiece'+numWhitePieces);
                whitePieces.push({piece: numWhitePieces, x: i,y:j,isChecker: false});

$(piece).attr('onclick','selectPiece("whitePiece'+numWhitePieces+'",'+i+','+j+')');
                numWhitePieces++;
                $(pieceDiv).append($(piece));
                $('#cell'+i+j).append(pieceDiv);
            }
        }
    }
```

```
    for(var i=6;i<9;i++){
        for(var j=1;j<9;j++){
            if($('#cell'+i+j).hasClass('green')){
                // Create one Brown piece, assign id, add piece to brown pieces array
and add onclick event
                var pieceDiv = document.createElement('div');
                $(pieceDiv).addClass('piece');
                var piece= document.createElement('a');
                $(piece).addClass('brownPiece');
                $(piece).attr('id','brownPiece'+numBrownPieces);
                brownPieces.push({piece: numBrownPieces, x: i,y:j,isChecker: false});

$(piece).attr('onclick','selectPiece("brownPiece'+numBrownPieces+'","'+i+','+j+'")');
                numBrownPieces++;
                $(pieceDiv).append($(piece));
                $('#cell'+i+j).append(pieceDiv);
            }
        }
    }
}
```

Aquesta funció el que fa és col·locar les peces blanques i marrons al taulell, als quadrats marrons a les tres primeres files començant per on es troba cada jugador, i a més, emplenant l'array amb totes les fitxes.

Funció selectPiece

```
/*
* selectPiece(piece,x,y) - Method for check if one piece can be selected, and if can be selected
* select the piece
*/
function selectPiece(piece,x,y){
    if(turn == BROWN && ($("#"+piece).hasClass('brownPiece') ||
$("#"+piece).hasClass('brownCheckerPiece'))){
        if($('.selectedPiece').length>0){

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('onclick');
        }else{
            var numNeighbors = 0;
            var possibleKill = weCanKill(piece,x,y);
            if(possibleKill[0].killMovement == true &&
possibleKill[0].killerPieces.indexOf(piece)!=-1 && forceJumpEnabled){
                var numPiece = piece.replace("brownPiece","");
                numPiece = parseInt(numPiece);
                var boardPiece = findPiece(x,y,numPiece,BROWN);
                showKillMovement(boardPiece,BROWN);
            }else if(possibleKill[0].killMovement == false || !forceJumpEnabled){
                if($("#"+piece).hasClass('brownCheckerPiece')){
                    numNeighbors = calcCheckerNeighbors(x,y,BROWN);
                }else{
                    numNeighbors = calcNeighbors(x,y,BROWN);
                }
            }
            if(numNeighbors>0){
                $("#"+piece).parent().addClass('selectedPiece').removeClass('green');
            }
        }
    }else if(turn == WHITE && ($("#"+piece).hasClass('whitePiece') ||
$("#"+piece).hasClass('whiteCheckerPiece'))){
        if($('.selectedPiece').length>0){

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('onclick');
        }else{
            var numNeighbors = 0;
            var possibleKill = weCanKill(piece,x,y);
            if(possibleKill[0].killMovement == true &&
possibleKill[0].killerPieces.indexOf(piece)!=-1 && forceJumpEnabled){
                var numPiece = piece.replace("whitePiece","");
                numPiece = parseInt(numPiece);
                var boardPiece = findPiece(x,y,numPiece,WHITE);
                showKillMovement(boardPiece,WHITE);
```

```
            }else if(possibleKill[0].killMovement == false || !forceJumpEnabled){
                if($("#"+piece).hasClass('whiteCheckerPiece')){
                    numNeighbors = calcCheckerNeighbors(x,y,WHITE);
                }else{
                    numNeighbors = calcNeighbors(x,y,WHITE);
                }
            }
            if(numNeighbors>0){
                $("#"+piece).parent().addClass('selectedPiece').removeClass('green');
            }
        }
    }
}
```

Aquesta funció comprova el torn en el que ens trobem, comprova si la fitxa és una fitxa normal o una dama, i comprova si la fitxa que hem clicat té veïns a on saltar, si és així, seleccionarà la peça i marcarà les posicions a on es pot moure la peça. Si pel contrari ja havia una peça seleccionada, desmarca la seleccionada, i si tampoc havia cap seleccionada no farà res.

Funció findPiece

```
/*
* findPiece(x,y,numPiece,actualTurn) - Method for find a piece in array
*/
function findPiece(x,y,numPiece,actualTurn){
    var found = false;
    var piece = "";
    if(actualTurn == BROWN){
        var i = 0;
        while(i<brownPieces.length && !found){
            found = brownPieces[i].x == x && brownPieces[i].y == y &&
brownPieces[i].piece == numPiece;
            if(!found){
                i++;
            }
        }
        if(found){
            piece = brownPieces[i];
        }
    }else{
        var i = 0;
        while(i<whitePieces.length && !found){
            found = whitePieces[i].x == x && whitePieces[i].y == y &&
whitePieces[i].piece == numPiece;
            if(!found){
                i++;
            }
        }
        if(found){
            piece = whitePieces[i];
        }
    }
    return piece;
}
```

Funció a la qual li passem la posició, x,y i numero de peça i ens retorna la peça sencera. Funció molt important per comprovar si una peça és dama o no.

Funció showKillMovement

```
/*
* showKillMovement(boardPiece,actualTurn) - When a force jump is enabled and we select
a piece that can kill
* this method shows which movement can do
*/
function showKillMovement(boardPiece,actualTurn){
    if(actualTurn == BROWN){ //is brown piece
        var leftNeighborX = 0;
        var leftNeighborY = 0;
        var x = boardPiece.x;
        var y = boardPiece.y;
        if(boardPiece.isChecker){
            // we check the left neighbor
            leftNeighborX= x+1;
            leftNeighborY= y-1;
            // we check that is a valid position && is not occupied
            var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                //if left neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                leftNeighborX= leftNeighborX+1;
                leftNeighborY= leftNeighborY-1;
                arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
                if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                }

            }
            // we check the right neighbor
            rightNeighborX= x+1;
            rightNeighborY= y+1;
            // we check that is a valid position && is not occupied
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                //if right neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                rightNeighborX= rightNeighborX+1;
                rightNeighborY= rightNeighborY+1;
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
                if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                }
            }
        }
```

```
        leftNeighborX= x-1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition = positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != BROWN ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX-1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition = positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
            }

        }
        // we check the right neighbor
        rightNeighborX= x-1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition = positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
        if(rightNeighborX >= 1 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != BROWN ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX-1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
            }
        }
    }else{ // is white turn
        var leftNeighborX = 0;
        var leftNeighborY = 0;
        var x = boardPiece.x;
        var y = boardPiece.y;
        if(boardPiece.isChecker){ // is checker
            // we check the left neighbor
            leftNeighborX= x-1;
            leftNeighborY= y-1;
            // we check that is a valid position && is not occupied
            var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                //if left neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                leftNeighborX= leftNeighborX-1;
                leftNeighborY= leftNeighborY-1;
                arrayPosition =
```

```
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
                if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                }
            }
            // we check the right neighbor
            rightNeighborX= x-1;
            rightNeighborY= y+1;
            // we check that is a valid position && is not occupied
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                //if right neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                rightNeighborX= rightNeighborX-1;
                rightNeighborY= rightNeighborY+1;
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
                if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                }
            }
        }
        // we check the left neighbor
        leftNeighborX= x+1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition = positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != WHITE ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX+1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition = positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
            if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
            }

        }
        // we check the right neighbor
        rightNeighborX= x+1;
        rightNeighborY= y+1;
```

```
        // we check that is a valid position && is not occupied
        arrayPosition = positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
        if(rightNeighborX <= 8 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != WHITE ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX+1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
            }
        }
    }
}
```

Funció utilitzada quan els salts (matar) és obligatori. Aquesta funció ens comprova per a cada tipus de peça i torn els possibles moviments de matar que pot fer una peça seleccionada.

Funció weCanKill

```
/*
* weCanKill(piece) - Method that shows if one piece can kill or not
*/
function weCanKill(piece){
    var possibleKill = [];
    var killMovement = false;
    var killerPieces = [];
    if(piece.indexOf('brown')!=-1){ //is brown piece
        for(var i=0;i<brownPieces.length;i++){
            var x = brownPieces[i].x;
            var y = brownPieces[i].y;
            var leftNeighborX = 0;
            var leftNeighborY = 0;
            if(brownPieces[i].isChecker){
                // we check the left neighbor
                leftNeighborX= x+1;
                leftNeighborY= y-1;
                // we check that is a valid position && is not occupied
                var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
                if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                    //if left neighbor is occupied by a piece of the enemy we check
if we can kill that piece
                    leftNeighborX= leftNeighborX+1;
                    leftNeighborY= leftNeighborY-1;
                    arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
                    if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                        //if the neighbor of the enemy's piece is not occupied we can
kill this piece
                        killMovement = true;
                        if(killerPieces.indexOf(brownPieces[i].piece)==-1){
                            killerPieces.push("brownPiece"+brownPieces[i].piece);
```

```
                        }
                    }

                }
                // we check the right neighbor
                rightNeighborX= x+1;
                rightNeighborY= y+1;
                // we check that is a valid position && is not occupied
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
                if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                    //if right neighbor is occupied by a piece of the enemy we check
if we can kill that piece
                    rightNeighborX= rightNeighborX+1;
                    rightNeighborY= rightNeighborY+1;
                    arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
                    if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                        //if the neighbor of the enemy's piece is not occupied we can
kill this piece
                        killMovement = true;
                        if(killerPieces.indexOf(brownPieces[i].piece)==-1){
                            killerPieces.push("brownPiece"+brownPieces[i].piece);
                        }
                    }
                }
            }
            leftNeighborX= x-1;
            leftNeighborY= y-1;
            // we check that is a valid position && is not occupied
            var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                //if left neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                leftNeighborX= leftNeighborX-1;
                leftNeighborY= leftNeighborY-1;
                arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,BROWN)[0];
                if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece
                    killMovement = true;
                    if(killerPieces.indexOf(brownPieces[i].piece)==-1){
                        killerPieces.push("brownPiece"+brownPieces[i].piece);
                    }
                }

            }
            // we check the right neighbor
            rightNeighborX= x-1;
            rightNeighborY= y+1;
            // we check that is a valid position && is not occupied
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != BROWN ){
                //if right neighbor is occupied by a piece of the enemy we check if
we can kill that piece
                rightNeighborX= rightNeighborX-1;
                rightNeighborY= rightNeighborY+1;
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,BROWN)[0];
                if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
```

```
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece
                    killMovement = true;
                    if(killerPieces.indexOf(brownPieces[i].piece)==-1){
                        killerPieces.push("brownPiece"+brownPieces[i].piece);
                    }
                }
            }
        }
    }else{ // is white turn
         for(var i=0;i<whitePieces.length;i++){
            var x = whitePieces[i].x;
            var y = whitePieces[i].y;
            var leftNeighborX = 0;
            var leftNeighborY = 0;
            if(whitePieces[i].isChecker){ // is checker
                // we check the left neighbor
                leftNeighborX= x-1;
                leftNeighborY= y-1;
                // we check that is a valid position && is not occupied
                var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
                if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                    //if left neighbor is occupied by a piece of the enemy we check
if we can kill that piece
                    leftNeighborX= leftNeighborX-1;
                    leftNeighborY= leftNeighborY-1;
                    arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
                    if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                        //if the neighbor of the enemy's piece is not occupied we can
kill this piece
                        killMovement = true;
                        if(killerPieces.indexOf(whitePieces[i].piece)==-1){
                            killerPieces.push("whitePiece"+whitePieces[i].piece);
                        }
                    }
                }
                // we check the right neighbor
                rightNeighborX= x-1;
                rightNeighborY= y+1;
                // we check that is a valid position && is not occupied
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
                if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                    //if right neighbor is occupied by a piece of the enemy we check
if we can kill that piece
                    rightNeighborX= rightNeighborX-1;
                    rightNeighborY= rightNeighborY+1;
                    arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
                    if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                        //if the neighbor of the enemy's piece is not occupied we can
kill this piece
                        killMovement = true;
                        if(killerPieces.indexOf(whitePieces[i].piece)==-1){
                            killerPieces.push("whitePiece"+whitePieces[i].piece);
                        }
                    }
                }
            }
            // we check the left neighbor
            leftNeighborX= x+1;
```

```
            leftNeighborY= y-1;
            // we check that is a valid position && is not occupied
            var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                //if left neighbor is occupied by a piece of the enemy we check if we
can kill that piece
                leftNeighborX= leftNeighborX+1;
                leftNeighborY= leftNeighborY-1;
                arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,WHITE)[0];
                if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece
                    killMovement = true;
                    if(killerPieces.indexOf(whitePieces[i].piece)==-1){
                        killerPieces.push("whitePiece"+whitePieces[i].piece);
                    }
                }

            }
            // we check the right neighbor
            rightNeighborX= x+1;
            rightNeighborY= y+1;
            // we check that is a valid position && is not occupied
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != WHITE ){
                //if right neighbor is occupied by a piece of the enemy we check if
we can kill that piece
                rightNeighborX= rightNeighborX+1;
                rightNeighborY= rightNeighborY+1;
                arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,WHITE)[0];
                if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                    //if the neighbor of the enemy's piece is not occupied we can kill
this piece
                    killMovement = true;
                    if(killerPieces.indexOf(whitePieces[i].piece)==-1){
                        killerPieces.push("whitePiece"+whitePieces[i].piece);
                    }
                }
            }
        }
    }
    possibleKill.push({killMovement:killMovement,killerPieces:killerPieces});
    return possibleKill;
}
```

Funció que comprova totes les fitxes per verificar si podem o no realitzar un moviment de matar, o no. Aquesta funció es important quan l'obligació de matar està activada ja que d'aquesta manera no podrem fer qualsevol moviment si no que haurem de fer un de matar si és possible.

Funció calcCheckerNeighbors

```
/*
* function calcCheckerNeighbors(x,y,actualTurn) - Method that shows checker neighbors
*/
function calcCheckerNeighbors(x,y,actualTurn){
    var numNeighbors = 0;
    numNeighbors = calcNeighbors(x,y,actualTurn);
    //Check neighbors of a checker brown piece
    if(actualTurn == BROWN){
        // we check the left neighbor
        leftNeighborX= x+1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX <=8 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== false){

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX+1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x+1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX <= 8 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== false){

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX+1;
            rightNeighborY= rightNeighborY+1;
```

```
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }else{
        // we check the left neighbor
        leftNeighborX= x-1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== false){

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX-1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x-1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX >= 1 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== false){

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+')');
            numNeighbors++;
```

```
        }else if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX-1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }
    return numNeighbors;
}
```

Funció que serveix per a comprovar els veïns d'una peça que és dama, i marcar el possibles moviments. A diferencia d'una peça normal, una dama pot fer moviments enrere i la comprovació es diferent als d'una peça normal.

Funció calcNeighbors

```
/*
* function calcNeighbors(x,y,actualTurn) - Method that shows normal piece neighbors
*/
function calcNeighbors(x,y,actualTurn){
    var numNeighbors = 0;
    //Check neighbors of a brown piece
    if(actualTurn == BROWN){
        // we check the left neighbor
        leftNeighborX= x-1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== false){

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX-1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
```

```
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x-1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX >= 1 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== false){

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX-1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }else{
        // we check the left neighbor
        leftNeighborX= x+1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX <=8 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== false){

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX+1;
```

```
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x+1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX <= 8 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== false){

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+')');
            numNeighbors++;
        }else if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX+1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }
    return numNeighbors;
}
```

Funció molt similar a l'anterior, la diferència és que només fa la comprovació dels veïns cap endavant i no endavant i enrere com la funció anterior.

Funció calcKillCheckerNeighbors

```
/*
* function calcKillCheckerNeighbors(x,y,actualTurn) - Method that shows checker
neighbors when force jump is enabled and we can kill a piece
*/
function calcKillCheckerNeighbors(x,y,actualTurn){
    var numNeighbors = calcKillNeighbors(x,y,actualTurn);
    //Check neighbors of a brown checker piece
    if(actualTurn == BROWN){
        // we check the left neighbor
        leftNeighborX= x+1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX+1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x+1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX <= 8 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX+1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }else{

        // we check the left neighbor
```

```
        leftNeighborX= x-1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX-1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
        // we check the right neighbor
        rightNeighborX= x-1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX >= 1 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX-1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }
    return numNeighbors;

}
```

Funció que s'utilitza per comprovar els veïns quan en un moviment ja hem matat. És a dir, si ja hem fet un moviment de matar, comprovem si es pot continuar matant o si pel contrari hem de passar turn. En aquest cas ho comprova per a una peça que és dama.

Funció calcKillNeighbors

```
/*
* function calcKillNeighbors(x,y,actualTurn) - Method that shows normal piece kill
neighbors
*/
function calcKillNeighbors(x,y,actualTurn){
  var numNeighbors = 0;
    //Check neighbors of a brown piece
    if(actualTurn == BROWN){
        // we check the left neighbor
        leftNeighborX= x-1;
        leftNeighborY= y-1;
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX-1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX >= 1 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
        // we check the right neighbor
        rightNeighborX= x-1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX >= 1 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX-1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX >= 1 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }else{
        // we check the left neighbor
        leftNeighborX= x+1;
        leftNeighborY= y-1;
```

```
        // we check that is a valid position && is not occupied
        var arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
        if(leftNeighborX >= 1 && leftNeighborY >= 1 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if left neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            leftNeighborX= leftNeighborX+1;
            leftNeighborY= leftNeighborY-1;
            arrayPosition =
positionOccupied(leftNeighborX,leftNeighborY,actualTurn)[0];
            if(leftNeighborX <= 8 && leftNeighborY >= 1 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+leftNeighborX+leftNeighborY).addClass('selectedPiece').removeClass('green
');

$("#cell"+leftNeighborX+leftNeighborY).attr('onclick','movePiece('+x+','+y+','+left
NeighborX+','+leftNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }

        }
        // we check the right neighbor
        rightNeighborX= x+1;
        rightNeighborY= y+1;
        // we check that is a valid position && is not occupied
        arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
        if(rightNeighborX <= 8 && rightNeighborY <= 8 && arrayPosition.positionOccupied
== true && arrayPosition.color != actualTurn ){
            //if right neighbor is occupied by a piece of the enemy we check if we can
kill that piece
            rightNeighborX= rightNeighborX+1;
            rightNeighborY= rightNeighborY+1;
            arrayPosition =
positionOccupied(rightNeighborX,rightNeighborY,actualTurn)[0];
            if(rightNeighborX <= 8 && rightNeighborY <= 8 &&
arrayPosition.positionOccupied == false){
                //if the neighbor of the enemy's piece is not occupied we can kill this
piece

$("#cell"+rightNeighborX+rightNeighborY).addClass('selectedPiece').removeClass('gre
en');

$("#cell"+rightNeighborX+rightNeighborY).attr('onclick','movePiece('+x+','+y+','+ri
ghtNeighborX+','+rightNeighborY+','+actualTurn+',true)');
                numNeighbors++;
            }
        }
    }
    return numNeighbors;
}
```

Funció similar a l'anterior però pel cas d'una peça normal. Comprova si després d'haver matat una peça es pot continuar matant.

Funció movePiece

```
function movePiece(oldX,oldY,newX,newY,actualTurn,killPiece){
    if(actualTurn == BROWN){
        var found = false;
        var i=0;
        var piecePosition;
        while(!found && i<brownPieces.length){
            found = (brownPieces[i].x == oldX && brownPieces[i].y == oldY && actualTurn
== BROWN);
            if(!found){
                i++;
            }
        }
        if(found){
            piecePosition = i;
            brownPieces[piecePosition].x = newX;
            brownPieces[piecePosition].y = newY;
            //we check if is a kill movement
            if(killPiece){

if($("#brownPiece"+brownPieces[piecePosition].piece).hasClass('brownPiece')){
                    if(newY - oldY < 0){
                        var enemyX = oldX-1;
                        var enemyY = oldY-1;
                    }else if(newY - oldY > 0){
                        var enemyX = oldX-1;
                        var enemyY = oldY+1;
                    }
                }else{ //is checker piece
                     if(newY - oldY < 0 && newX-oldX < 0){
                        var enemyX = oldX-1;
                        var enemyY = oldY-1;
                    }else if(newY - oldY > 0 && newX-oldX<0){
                        var enemyX = oldX-1;
                        var enemyY = oldY+1;
                    }else if(newY-oldY<0 && newX-oldX>0){
                        var enemyX = oldX+1;
                        var enemyY = oldY-1;
                    }else if(newY-oldY>0 && newX-oldX>0){
                        var enemyX = oldX+1;
                        var enemyY = oldY+1;
                    }
                }

                killSelectedPiece(enemyX,enemyY,WHITE);
                var numNeighbors = 0;

if($("#brownPiece"+brownPieces[piecePosition].piece).hasClass('brownPiece')){
                    numNeighbors = calcKillNeighbors(newX,newY,BROWN);
                }else{
                    numNeighbors = calcKillCheckerNeighbors(newX,newY,BROWN);
                }
                if(numNeighbors>0){

$("#brownPiece"+brownPieces[piecePosition].piece).parent().addClass('selectedPiece'
).removeClass('green');
                    var i=0;
                    while(i<brownPieces.length){
                        if(brownPieces[i].x != newX || brownPieces[i].y != newY){

$('#brownPiece'+brownPieces[i].piece).removeAttr('onclick'); // TODO - intentar no
eliminarlo, solo deshabilitarlo
                        }
                        i++;
                    }
                    // we move the piece to the selected position and we can do other
movement
```

```
$('#cell'+newX+newY).append($('#brownPiece'+brownPieces[piecePosition].piece).paren
t());
                    $('#cell'+oldX+oldY).html('');
                    checkCreateChecker(brownPieces[piecePosition].piece,BROWN);
                    $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#brownPiece'+brownPieces[piecePosition].piece).attr('onclick','selectPiece("brow
nPiece'+brownPieces[piecePosition].piece+'",'+newX+','+newY+')');
                }else{
                    // we move the piece to the selected position and change turn

$('#cell'+newX+newY).append($('#brownPiece'+brownPieces[piecePosition].piece).paren
t());
                    $('#cell'+oldX+oldY).html('');
                    checkCreateChecker(brownPieces[piecePosition].piece,BROWN);
                    $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#brownPiece'+brownPieces[piecePosition].piece).attr('onclick','selectPiece("brow
nPiece'+brownPieces[piecePosition].piece+'",'+newX+','+newY+')');
                    $('#turnInfoBrown').css('visibility','hidden');
                    $('#turnInfoWhite').css('visibility','initial');
                    if(soundsEnabled){
                        move_piece_sound();
                    }
                    turn = WHITE;
                    var i=0;
                    while(i<brownPieces.length){
                        if(brownPieces[i].x != newX || brownPieces[i].y != newY){

$('#brownPiece'+brownPieces[i].piece).attr('onclick','selectPiece("brownPiece'+brow
nPieces[i].piece+'",'+brownPieces[i].x+','+brownPieces[i].y+')');
                        }
                        i++;
                    }
                }
            }else{
                // we move the piece to the selected position and change turn

$('#cell'+newX+newY).append($('#brownPiece'+brownPieces[piecePosition].piece).paren
t());
                $('#cell'+oldX+oldY).html('');
                checkCreateChecker(brownPieces[piecePosition].piece,BROWN);
                $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#brownPiece'+brownPieces[piecePosition].piece).attr('onclick','selectPiece("brow
nPiece'+brownPieces[piecePosition].piece+'",'+newX+','+newY+')');
                $('#turnInfoBrown').css('visibility','hidden');
                $('#turnInfoWhite').css('visibility','initial');
                if(soundsEnabled){
                    move_piece_sound();
                }
                turn = WHITE;
                var i=0;
                while(i<brownPieces.length){
                    if(brownPieces[i].x != newX || brownPieces[i].y != newY){

$('#brownPiece'+brownPieces[i].piece).attr('onclick','selectPiece("brownPiece'+brow
nPieces[i].piece+'",'+brownPieces[i].x+','+brownPieces[i].y+')');
```

```
                    }
                    i++;
                }
            }
        }
    }else{
        var found = false;
        var i=0;
        while(!found && i<whitePieces.length){
            found = (whitePieces[i].x == oldX && whitePieces[i].y == oldY && actualTurn
== WHITE);
            if(!found){
                i++;
            }
        }
        if(found){
            piecePosition = i;
            // TODO - Actualizar datos de la pieza en el array
            whitePieces[piecePosition].x = newX;
            whitePieces[piecePosition].y = newY;
            //we check if is a kill movement
            if(killPiece){


if($("#whitePiece"+whitePieces[piecePosition].piece).hasClass('whitePiece')){
                    if(newY - oldY < 0){
                        var enemyX = oldX+1;
                        var enemyY = oldY-1;
                    }else if(newY - oldY > 0){
                        var enemyX = oldX+1;
                        var enemyY = oldY+1;
                    }
                }else{ //is checker piece
                     if(newY - oldY < 0 && newX-oldX < 0){
                        var enemyX = oldX-1;
                        var enemyY = oldY-1;
                    }else if(newY - oldY > 0 && newX-oldX<0){
                        var enemyX = oldX-1;
                        var enemyY = oldY+1;
                    }else if(newY-oldY<0 && newX-oldX>0){
                        var enemyX = oldX+1;
                        var enemyY = oldY-1;
                    }else if(newY-oldY>0 && newX-oldX>0){
                        var enemyX = oldX+1;
                        var enemyY = oldY+1;
                    }
                }
                killSelectedPiece(enemyX,enemyY,BROWN);
                var numNeighbors = 0;

if($("#whitePiece"+whitePieces[piecePosition].piece).hasClass('whitePiece')){
                    numNeighbors = calcKillNeighbors(newX,newY,WHITE);
                }else{
                    numNeighbors = calcKillCheckerNeighbors(newX,newY,WHITE);
                }
                if(numNeighbors>0){

$("#whitePiece"+whitePieces[piecePosition].piece).parent().addClass('selectedPiece'
).removeClass('green');
                    var i=0;
                    while(i<whitePieces.length){
                        if(whitePieces[i].x != newX ||whitePieces[i].y != newY){

$('#whitePiece'+whitePieces[i].piece).removeAttr('onclick'); // TODO - intentar no
eliminarlo, solo deshabilitarlo
                        }
                        i++;
                    }
```

```
                    // we move the piece to the selected position and we can do other
movement

$('#cell'+newX+newY).append($('#whitePiece'+whitePieces[piecePosition].piece).paren
t());
                    $('#cell'+oldX+oldY).html('');
                    checkCreateChecker(whitePieces[piecePosition].piece,WHITE);
                    $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#whitePiece'+whitePieces[piecePosition].piece).attr('onclick','selectPiece("whit
ePiece'+whitePieces[piecePosition].piece+'","'+newX+','+newY+')');
                }else{
                    // we move the piece to the selected position and change turn

$('#cell'+newX+newY).append($('#whitePiece'+whitePieces[piecePosition].piece).paren
t());
                    $('#cell'+oldX+oldY).html('');
                    checkCreateChecker(whitePieces[piecePosition].piece,WHITE);
                    $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#whitePiece'+whitePieces[piecePosition].piece).attr('onclick','selectPiece("whit
ePiece'+whitePieces[piecePosition].piece+'","'+newX+','+newY+')');
                    $('#turnInfoBrown').css('visibility','initial');
                    $('#turnInfoWhite').css('visibility','hidden');
                    if(soundsEnabled){
                        move_piece_sound();
                    }
                    turn = BROWN;
                    var i=0;
                    while(i<whitePieces.length){
                        if(whitePieces[i].x != newX ||whitePieces[i].y != newY){

$('#whitePiece'+whitePieces[i].piece).attr('onclick','selectPiece("whitePiece'+whit
ePieces[i].piece+'","'+whitePieces[i].x+','+whitePieces[i].y+')');
                        }
                        i++;
                    }
                }
            }else{
                // we move the piece to the selected position

$('#cell'+newX+newY).append($('#whitePiece'+whitePieces[piecePosition].piece).paren
t());
                $('#cell'+oldX+oldY).html('');
                checkCreateChecker(whitePieces[piecePosition].piece,WHITE);
                $('.selectedPiece').removeAttr('onclick');

$('.selectedPiece').addClass('green').removeClass('selectedPiece').removeAttr('oncl
ick');

$('#whitePiece'+whitePieces[piecePosition].piece).attr('onclick','selectPiece("whit
ePiece'+whitePieces[piecePosition].piece+'","'+newX+','+newY+')');
                $('#turnInfoBrown').css('visibility','initial');
                $('#turnInfoWhite').css('visibility','hidden');
                if(soundsEnabled){
                    move_piece_sound();
                }
                turn = BROWN;
                var i=0;
                while(i<whitePieces.length){
                    if(whitePieces[i].x != newX ||whitePieces[i].y != newY){
```

```
$('#whitePiece'+whitePieces[i].piece).attr('onclick','selectPiece("whitePiece'+whit
ePieces[i].piece+'",'+whitePieces[i].x+','+whitePieces[i].y+')');
                    }
                    i++;
                }
            }
        }
    }
}
```

Una de les funcions mes important del joc, mou la peça d'una posició a l'altre però comprova si es un moviment de matar, actualitzar el llistat de peces, canvia el torn, etc...

Funció checkCreateChecker

```
function checkCreateChecker(numPiece,actualTurn){
    var found = false;
    var i=0;
    if(actualTurn == BROWN){
        while(!found && i<brownPieces.length){
            found = (brownPieces[i].piece == numPiece);
            if(!found){
                i++;
            }
        }
        if(found){
            if(brownPieces[i].x == 1 &&  brownPieces[i].isChecker == false ){
                brownPieces[i].isChecker = true;

$('#brownPiece'+brownPieces[i].piece).removeClass('brownPiece').addClass('brownChec
kerPiece');
            }
        }
    }else if(actualTurn == WHITE){
        while(!found && i<whitePieces.length){
            found = (whitePieces[i].piece == numPiece);
            if(!found){
                i++;
            }
        }
        if(found){
            if(whitePieces[i].x == 8 &&  whitePieces[i].isChecker == false ){
                whitePieces[i].isChecker = true;

$('#whitePiece'+whitePieces[i].piece).removeClass('whitePiece').addClass('whiteChec
kerPiece');
            }
        }
    }
}
```

Funció que, després de realitzar un moviment, comprova si la posició de la peça es per a crear una dama o no. Si es possible, crea la dama.

Funció killSelectedPiece

```
function killSelectedPiece(enemyX,enemyY,colorPiece){
    if(colorPiece == BROWN){
        var found = false;
        var i=0;
        while(!found && i<brownPieces.length){
            found = (brownPieces[i].x == enemyX && brownPieces[i].y == enemyY);
            if(!found){
                i++;
            }
        }
        if(found){
            $('#brownPiece'+brownPieces[i].piece).parent().remove();
            brownPieces.splice(i,1);
            if(vibrationEnabled){
                intel.xdk.notification.vibrate();
            }
            if(brownPieces.length == 0){
                $.ui.popup( {
                   title:"Victory",
                   message:"White wins!",
                   cancelText:"Ok",
                   doneCallback: function(){

                   },
                   cancelOnly:true
                });
            }else if(!availableMovements(brownPieces,BROWN)){
                $.ui.popup( {
                   title:"Victory",
                   message:"Brown pieces can not make more movements. White wins!",
                   cancelText:"Ok",
                   doneCallback: function(){

                   },
                   cancelOnly:true
                });
            }
        }
    }else if(colorPiece == WHITE){
        var found = false;
        var i=0;
        while(!found && i<whitePieces.length){
            found = (whitePieces[i].x == enemyX && whitePieces[i].y == enemyY);
            if(!found){
                i++;
            }
        }
        if(found){
            $('#whitePiece'+whitePieces[i].piece).parent().remove();
            whitePieces.splice(i,1);
            if(vibrationEnabled){
                intel.xdk.notification.vibrate();
            }
            if(whitePieces.length == 0){
                $.ui.popup( {
                   title:"Victory",
                   message:"Brown wins!",
                   cancelText:"Ok",
                   doneCallback: function(){

                   },
                   cancelOnly:true
                });
            }else if(!availableMovements(whitePieces,WHITE)){
                $.ui.popup( {
                   title:"Victory",
```

```
                    message:"White pieces can not make more movements. Brown wins!",
                    cancelText:"Ok",
                    doneCallback: function(){

                    },
                    cancelOnly:true
                });
            }
        }
    }
}
```

Funció que a partir d'unes coordenades x,y i del color de la peça, elimina una peça del taulell.

Funció availableMovements

```
function availableMovements(pieces,nextTurn){
    var continueGame = false;
    var i = 0;
    while(i<pieces.length && !continueGame){
        if(pieces[i].isChecker){
            continueGame = (calcCheckerNeighbors(pieces[i].x,pieces[i].y,nextTurn) >
0);
        }else{
            continueGame = (calcNeighbors(pieces[i].x,pieces[i].y,nextTurn) > 0);
        }

        if(!continueGame){
            i++;
        }
    }
    return continueGame;
}
```

Funció que retorna si un jugador té moviments disponibles o no. Funció fonamental per saber si un jugador perd per no poder moure les peces.

Funció positionOccupied

```
function positionOccupied(x,y,actualTurn){
    var arrayNeighbor = [];
    var positionOccupied = false;
    var color = -1;
    var i = 0;
    while(!positionOccupied && i<whitePieces.length){
        positionOccupied = (x == whitePieces[i].x && y == whitePieces[i].y);
        if(!positionOccupied){
            i++;
        }else{
            color = WHITE;
        }
    }
    if(!positionOccupied){
        var i = 0;
        while(!positionOccupied && i<brownPieces.length){
            positionOccupied = (x == brownPieces[i].x && y == brownPieces[i].y);
            if(!positionOccupied){
                i++;
            }else{
                color = BROWN;
            }
        }
```

```
    }
    arrayNeighbor.push({positionOccupied:positionOccupied,color:color});
    return arrayNeighbor;

}
```

Funció per determinar si una posició x,y està ocupada.

Funció makeCells

```
function makeCells()
{
    resetBoard();
    $('#prueba').height($('#prueba').width());
    $('#checkerboard').height($('#checkerboard').width());
    var f = document.getElementById('field');
    for (var i=1; i<9; i++) {
        var row = document.createElement('tr');
        var drow = [];
        for (var j=1; j<9; j++) {
            var square = document.createElement('td');
            $(square).attr('id','cell'+i+j);
            if (i%2 == 0){
                        if (j%2 == 0){
                                square.setAttribute('class', 'cell slowchange
white');
                        }else{

                    square.setAttribute('class', 'cell slowchange green')
                        }
                    }
                    else{
                        if (j%2 == 0){
                                square.setAttribute('class', 'cell slowchange
green')
                        }else{

                    square.setAttribute('class', 'cell slowchange white');
                        }
                    }

            $(square).attr('id',"cell"+i+j);
            $(square).css('width',"12%");
            $(square).css('height',"12%");
            row.appendChild(square);
            drow[j] = { "dom": square, "color": '', "x": i, "y": j};
        }
        f.appendChild(row);
    }
    addPieces();
}
```

Funció inicial del lloc, reseteja la pantalla, crea el taulell i crida a la funció per afegir les peces.

Funció requestTie

```
function requestTie(turn){
    var nextTurn  = '';
    if(turn == 'white') nextTurn = 'brown'
    else nextTurn = 'white';
    $.ui.popup( {
       title:"You are requesting a tie",
       message:"Are you sure you want to request a tie?",
       doneText:"Yes",
       cancelText:"No",
       doneCallback: function(){acceptTie(nextTurn)},
       cancelOnly:false
     });
}
```

Funció per a demanar l'empat a l'altre jugador

Funció acceptTie

```
function acceptTie(nextTurn){
     $.ui.popup( {
       title:"Tie request",
       message:nextTurn + " player is requesting a tie, do you accept?",
       doneText:"Yes",
       cancelText:"No",
       doneCallback: function(){
            $.ui.popup( {
               title:"Game finish TIE!",
               message:"It's a TIE!",
               cancelText:"Ok",
               cancelCallback: function(){
                   resetBoard();
               },
               cancelOnly:true
             });
       },
       cancelOnly:false
     });
}
```

Funció per a acceptar o no l'empat de l'altre jugador.

Funció giveUp

```
function giveUp(turn){
     var nextTurn  = '';
     if(turn == 'white') nextTurn = 'brown'
     else nextTurn = 'white';
     $.ui.popup( {
       title:"Give up request",
       message:"Are you sure you want to give up?",
       doneText:"Yes",
       cancelText:"No",
       doneCallback: function(){
            $.ui.popup( {
               title:"Victory!",
               message: nextTurn + " player wins!",
               cancelText:"Ok",
               cancelCallback: function(){
                   resetBoard();
               },
               cancelOnly:true
             });
       },
       cancelOnly:false
```

```
    });
}
```

Funció que obre un popup d'alerta per confirmar que l'usuari es vol rendir.

Funció resetBoard

```
function resetBoard(){
    whitePieces = [];
    brownPieces = [];
    turn = BROWN;
    $('#field').html('');
    $('#turnInfoWhite').css('visibility','hidden');
    $('#turnInfoBrown').css('visibility','initial');
}
```

Funció per resetejar el taulell cada vegada que entrem a la pantalla.