**Cheessy**

De Adrian-Marian Popoviciu



**Cuprins**

1. Introducere

2. Reguli de joc

3. Cerințe de sistem

4. Implementare

5. Justificarea tehnologiilor folosite

6. Arhitectură și funcționalitate

7. Direcții viitoare de dezvoltare

8. Referințe și resurse externe

**1. Introducere**

Cea mai populară interpretare a originii jocului de șah este că acesta este originar din India din timpul imperiului Gupta. Timpul și-a pus amprenta pe acest joc prin nenumărate modificări la regulile de joc. Forma curentă a jocului de șah a apărut în sudul Europei, în a două jumătate a secolului al XV-lea.

Cheessy este este o reinterpretare a jocului de șah care păstrează nostalgia pieselor și a tablei de joc prin modele 3D, dar care modifică regulile de joc astfel încât acestea adoptă elemente moderne regăsite în jocurile video contemporane.

*Elemente distinctive:*

# -Rotirea direcțională în 360° a pieselor.

# -Fiecare piesă are acum abilități noi.

# -Matricea de 8x8 pătrate pentru piese a fost înlocuită de o matrice de 8x8 pătrate, unde fiecare pătrat este egal cu 100 unităţi în spaţiul 3D.

# -Fiecare piesă se deplasează un număr prestabilit de unităţi în direcţia în care a fost orientată. În cazul în care piesa întâlneşte un obstacol, va ricoşă pe reflecţia vectorului de provenienţă:C:\Users\silly\Desktop\InfoEducatie\vectors1.gifC:\Users\silly\Desktop\InfoEducatie\vectors2.gif

# V = Vectorul de provenienţă R = Reflexia vectorului de provenienţă

# N = Vectorul reprezentat de coordonatele punctului lovit.

# 2. Reguli de joc

# La începutul jocului, fiecărui jucător îi este atribuit un set familiar format din 8 pioni, 2 cai, 2 nebuni , 2 turnuri (ture), un rege şi o regină. Culoarea albă are dreptul la prima mutare asemenea jocului clasic de şah. Scopul jocului este de a distruge regele inamic prin utilizarea strategică a pieselor.

# *Mi*ş*carea pieselor* se realizează pe rând între jucători. Jucătorul trebuie să rotească piesa şi apoi să o lanseze pe direcţia aleasă. Odată lansată piesă, rândul jucătorului s-a încheiat, iar când piesă se va opri din deplasare, rândul celuilalt jucător va începe.

# *Îmbunătăţirea şi abilităţile pieselor:* Piesele sunt îmbunătăţite prin distrugerea pieselor inamice. Abilităţile dobândite prin îmbunătăţirea pieselor sunt:

# -Pionii îşi dublează distanţa de mişcare. Abilitatea poate fi reactivată.

# -Caii nu ricoşează când distrug o piesă.

# -Nebunii crează un cerc cu diametrul de 100 unităţi. Dacă o piesă inamică cu scut intră în cerc, această îşi pierde scutul. Dacă regina intră în cerc, aceasta este teleportată la o poziţie aleatorie.

# -Turnurile crează un cerc cu diametrul de 100 unităţi. Dacă o piesă aliată intră în cerc, aceasta primește imunitate până la ieșirea din cerc când primește un scut.

# -Regina se teleportează la o poziție aleatorie. Abilitatea poate fi reactivată.

# -Regele primește un scut. Abilitatea poate fi reactivată.

# *Distanțele parcurse de piese:*

# -Pionii se deplasează 100 de unități. Distanța se poate schimba cu îmbunătăţirea piesei.

# -Caii se deplasează 300 de unități.

# -Nebunii si Turnurile se deplasează 500 de unități.

# -Regina se deplasează 700 de unități.

# -Regele se deplasează 200 de unități.

# 3. Cerințe de sistem

# Sistem de operare: Windows, Linux sau Android.

# Procesor: 1Ghz sau mai puternic.

# Memorie RAM: 256MB

# Placă video: Integrată pe processor sau mai bună.

# Versiune DirectX: 11.0

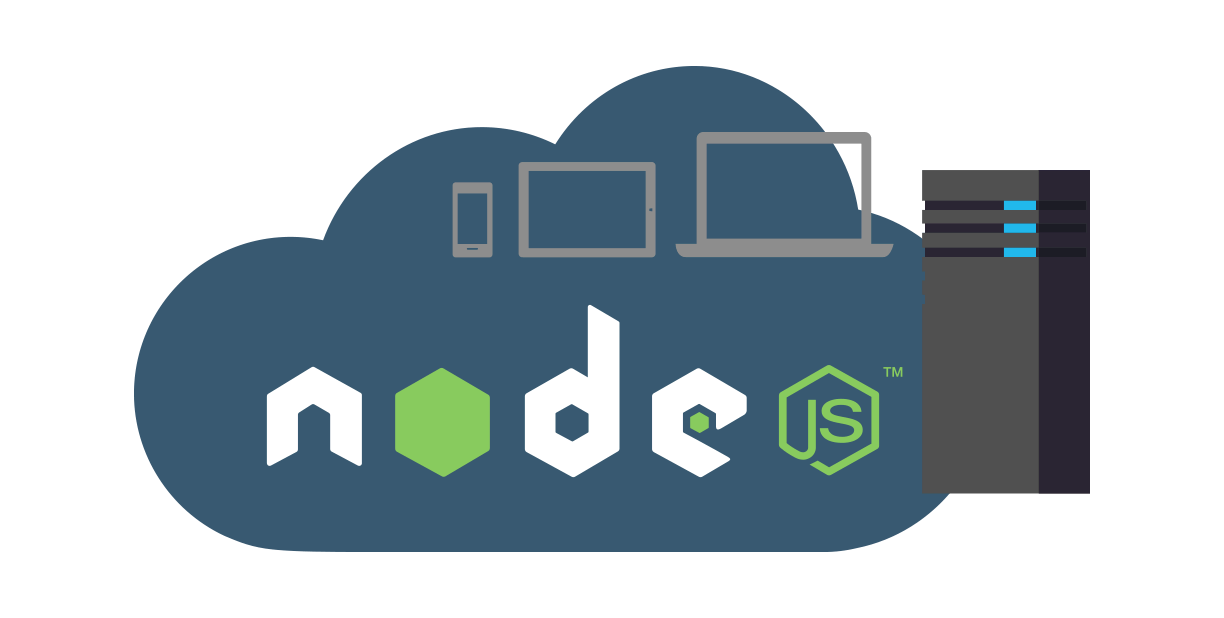
# Conexiune la internet: 256 KBPS pentru serviciile online.

# Spațiu liber pe HDD: 350 MB

# 4. Implementare

# Aplicația a fost scrisă în C++ în Visual Studio Community 2015. Pentru anumite elemente ce țin de definirea grafică a aplicației am folost și sistemul de visual scripting, numit „Blueprints”, prezent în Unreal Engine 4.15, iar baza de date pentru conturi, autentificarea și căutarea online pentru alți jucători este făcută în JavaScript prin platforma GameSparks. Tot în JavaScript a fost scris și server-ul de tip UDP ce funcționează pe o instanță de Windows Server 2012 susținută prin AWS Ec2. Pentru rularea procesului am folosit Node.js, deschis prin linii de command Windows.

# Modelele și efectele 3D au fost realizate cu ajutorul programului 3Ds Max de la Autodesk, utilizând licența de elev. Elementele 2D de tip GUI au fost realizate de Kenney sub licență royalty free pentru proiecte comerciale și non comerciale. Modificările de culoare și iconițele au fost realizate în GIMP 2.



# 5. Justificarea tehnologiilor folosite

# Unreal Engine 4 a fost ales ca fundația aplicației datorită performanțelor grafice mari în raport cu o necesitate hardware cât mai mică, a nenumăratelor instrumente oferite gratuit dezvoltatorilor și a faptului că necesită ca limbaj de programare C++.

# Limbajul de programare C++ este un limbaj de programare preferat de mine datorită faptului că este de tip multi-paradigmă care suportă programarea procedurală și programarea orientată pe obiecte. De asemenea codul este ușor de întreținut și poate comunica ușor cu alte limbaje de programare, în cazul meu cu Blueprints din Unreal Engine 4 și JavaScript prin mesaje JSON (JavaScript Object Notation) cu platforma GameSparks.

# Platforma GameSparks oferă gratuit servicii de gestionare a jucătorilor prin proceduri de stocare în baza de date NoSQL susținută prin MongoDB și cloud computing în limbajul de programare JavaScript care acționează ca intermediar pentru server și codul în C++ al jocului prin trimiterea mesajelor JSON.

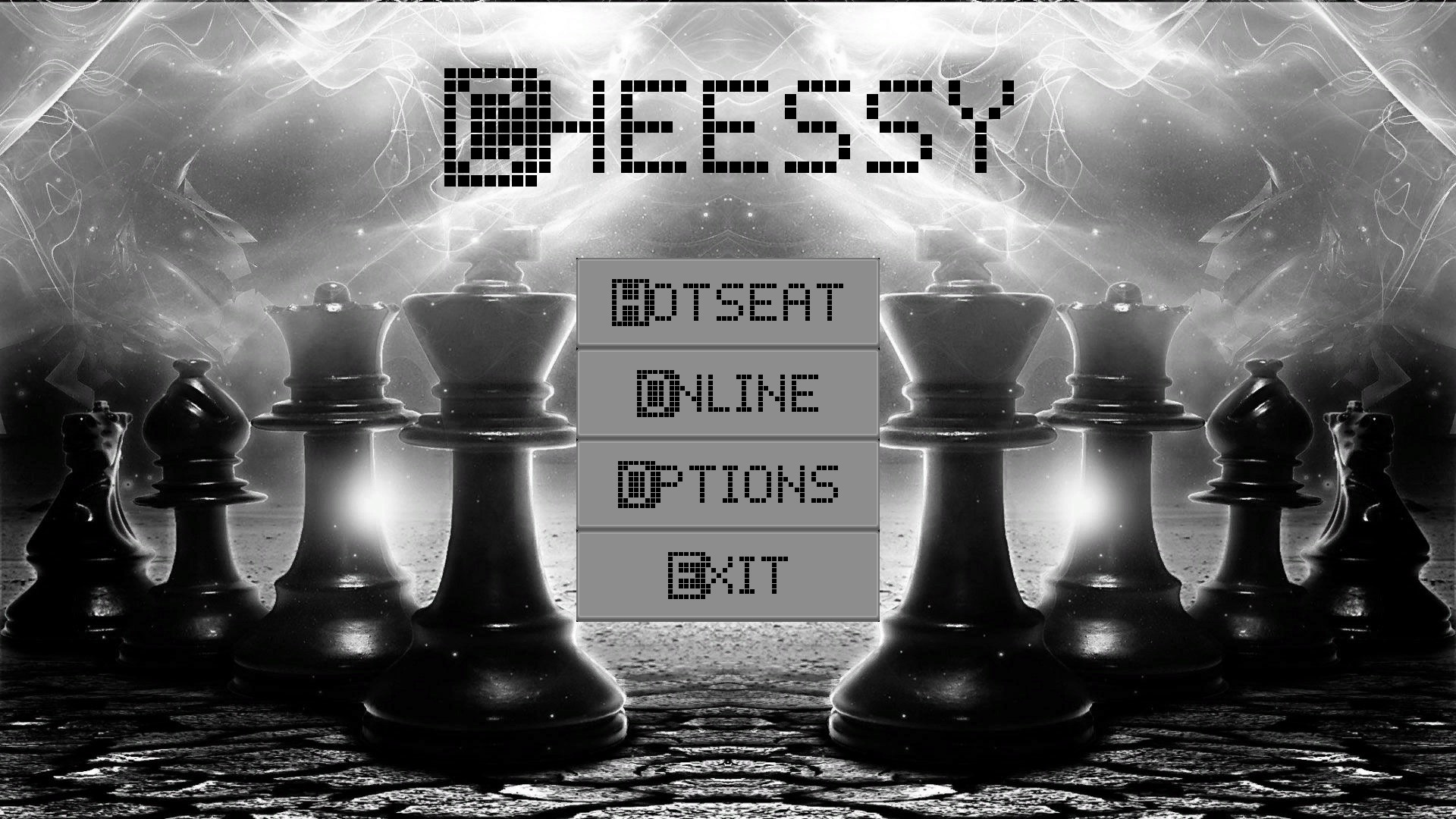
# Limbajul de programare JavaScript a fost ales pentru posibilitatea de a utiliza serviciile de cloud computing oferite de GameSparks și comunicarea ușoară cu C++, dar și caracteristicele ce le împarte cu acesta.

# Amazon Web Services Elastic Compute Cloud 2 îmi permite să rulez o instanță VM timp de 750 ore pe lună complet gratuit și se ocupă în mod automat de mentenanță 24/7.

# Node.js este un runtime pentru JavaScript, creat pentru construirea aplicaților conectate la rețele. Acesta menține o conexiune deschisă pe instanța de AWS EC2 iar la primirea unui mesaj de tip ping de la un player va crea dinamic un proces al unui server dedicat pentru sesiunea a 2 jucători conectați prin GameSparks.

# SCUE4 este o extensie ce oferă o soluție Anti-Cheat pentru securitatea aplicației în Unreal Engine 4 și este open-source. Această criptează valorile variabilelor printr-o cheie generată aleator și închide aplicația automat după DLL injection sau un debugger atașat pe proces.

# 6. Arhitectură și funcționalitate



# *Meniul principal*

# Interfața jocului este un exemplu perfect pentru comunicarea și colaborarea dintre limbajul de programare C++ și sistemul de visual scripting Blueprints. Tranzițiile au fost făcut cu ajutorul sistemului de visual scripting, iar în cazul necesității executării funcțiilor mai avansate cum ar fi schimbarea setărilor, sunt apelate funcții definite în C++ și expuse către Blueprints.

# C:\Users\silly\OneDrive\Pictures\Capturi de ecran\2017-05-30 (4).png C:\Users\silly\OneDrive\Pictures\Capturi de ecran\2017-05-30 (7).png

# Codul funcțiilor GetVideoQualitySettings și SetVideoQualitySettings scrise în C++ (stânga) și apelate în Blueprints (dreapta).

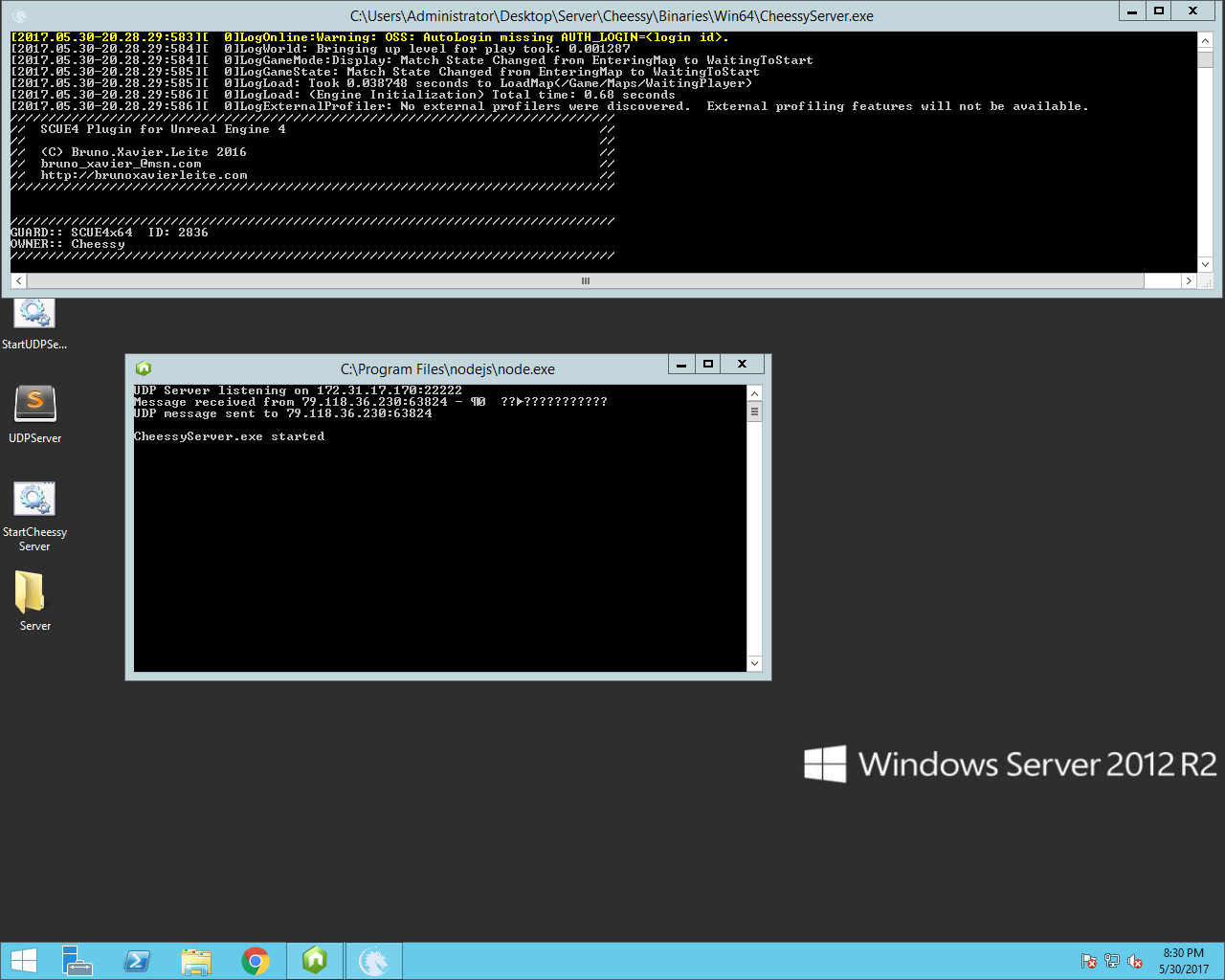


# *Modelul client-server utilizat de aplicație*

# Rolul de server poate fi însușit de un jucător, iar apoi un alt jucător să se conecteza la adresa de IP pe care a fost creat server-ul sau în cazul unei conexiuni LAN existente, se poate realiza o conectare directă în LAN.

# În cazul în care jucătorul alege să își caute un oponent prin platforma GameSparks, în momentul găsirii acestuia, clientul va primi un mesaj în urmă căruia va execută FUDPPing::UDPEcho(Adress, 5.0f, PingResult).

# La primirea unui mesaj de tip ping, serverul UDP de pe AWS va deschide o sesiune dedicată pentru cei doi jucatori. În imaginea alăturată putem observa server-ul UDP prin Node.js (jos) alături de server-ul dedicat sesiunii de joc (sus), rulate pe instanța Windows Server 2012 R2 prin AWS EC2.



# *Serviciile GameSparks*

# Comunicarea cu serviciile GameSparks este realizată printr-o serie de funcții de tip static incluse in extensia oficială pentru Unreal Engine. În stânga avem declararea acestor funcții din fișierul MainMenuGameMode.h

La pornirea jocului, acesta se va conecta la platforma GameSparks, conexiune care va fi mereu reușită în prezența conexiunii la internet. Odată conectați la platformă, este posibil să realizăm o comunicare cu codul JavaScrip de pe aceasta. În acest fel putem apela funcții fundamentale deja prezente sau putem trimite un LogEvent\_Request în C++ sau în Blueprints care execută funcții definite de dezvoltatori în codul din cloud.

# 7. Direcții viitoare de dezvoltare

# -Mutarea codului jocului în întregime pe codul din cloud. Acest lucru îmi va oferi protecție împotrivă anumitor modificări locale.

# -Modificarea sistemului de conectare la rețea astfel încât să folosească un model de tip P2P în loc de unul UDP/TCP. P2P este mult mai stabil pentru platformele mobile decât UDP/TCP și nu o să mai fiu nevoit să depind de AWS EC2.

# -Utilizarea unor imagini 2D în loc de 3D. În conceptul original, acest joc era 2D deoarece a fost gândit special pentru Android. Pentru acest concurs am optat pentru o reprezentare 3D. De asemenea, utilizarea graficii 2D va reduce cerințele aplicației.

# -Împrumutarea unor elemente din jocuri tip TCG (exemplu Hearthstone) cum ar fi posibilitatea de a crea seturi de joc modulare pentru a oferi libertate de alegere jucătorilor. Stocarea acestor piese în baza de date NoSQL se va face după următorul JSON:

* cardName: "The King"
* cardType: "Hero"
* set: "starterCards"
* attack: 1
* health: 10
* distance: 200
* bounce: 1
* spawnCost: 1
* ability: "striker"

# 8. Referințe și resurse externe

# <https://docs.gamesparks.com/>

# <https://www.gamesparks.com/>

# <https://docs.unrealengine.com/latest/INT/>

# <https://github.com/BrUnOXaVIeRLeiTE/SCUE4-Plugin>

# <https://nodejs.org/en/docs/>

# <https://aws.amazon.com/ec2/>

# <http://kenney.nl/>

# <http://www.dafont.com/chess.font>