

JsGames

Зорница Атанасова Костадинова, 4 курс, КН, фн: 80227, ФМИ,
Искрен Ивов Чернев, 4 курс, КН, фн: 80246, ФМИ

11 февруари 2011 г.

Цел на проекта

Настоящият документ е курсова работа към проекта “JsGames” по предмета “Изкуствен Интелект”. Описали сме задачата която си поставихме с този проект и решението ѝ. Обяснени са алгоритмите по които агентите вземат решения за следващ ход във всяка от игрите. Освен това сме представили програматичната реализация на сайта - база данни, интерфейс, архитектура, използвани технологии. Накрая говорим за възможностите за разширяване на проекта.

Съдържание

1	Описание на проекта	3
2	Използвани технологии	4
2.1	Haml	4
2.2	Sass	4
2.3	Ruby on Rails Framework	4
2.4	NodeJs	5
2.5	jQuery	5
2.6	Yui3 library	5
2.7	Mercurial	5
2.8	RubyGems	6
2.9	Статистика за проекта	6
3	Реализация	6
3.1	Програмен език	6
3.2	Дизайн	8
4	Изкуствен интелект	9
4.1	SG теория	9
5	Инсталация	12
5.1	Инсталиране на зависимости	12
6	Декларация за липса на плагиатство	13

1 Описание на проекта

Проект: сайт с javascript игри с изкуствен интелект.¹

- сайта ще поддържа потребители със следната информация за тях:
 - email
 - парола
 - име
- информация за игра:
 - име
 - кратко описание
- за всяка изиграна игра ще се пази следната информация:
 - коя е играта
 - потребители играли играта (може някой да са компютри, в такъв случай се пази трудността на компютъра)
 - резултат
 - продължителност на играта
- възможност за изкарване на класиране:
 - най-много изиграни игри
 - общо за всички игри или за конкретна игра
 - най-много изкарани точки
 - общо точки или средно аритметично, общо и за конкретна игра
 - най-дълго прекарано време в игри
 - общо време или средно аритметично, общо или за конкретна игра
- ще има възможност за динамично добавяне, редактиране, изтриване на горепосочените данни където това има смисъл
- допълнителни пояснения за игрите:
 - игрите ще бъдат имплементирани на `javascript`
 - всички игри ще имат изкуствен интелект (имплементиран като `javascript` клиент) с поне една степен на трудност
 - сървърът ще поддържа комуникация между различни `javascript` клиенти (браузъри) при игра на няколко души за да се обменят изиграните ходове
 - ще може да се играе и само от един клиент (браузър) ако единия играч е човек, а другият компютър (изкуствен интелект)

¹Проектът се хоства на <http://iskren.info:50005/>.

2 Използвани технологии

2.1 Haml

Haml е [1] маркъп език, с който може чисто и просто да се описва XHTML за всеки онлайн документ, без вграждане на код. Haml е предназначен като алтернатива на други шаблонни езици които вграждат код като PHP, ERB и ASP. Също така с Haml избягвате писането на чист XHTML с използването на семантична идентация.

2.2 Sass

Sass [2] е добавка към CSS3 [3], в която е добавено влагане на правила, променливи, модули, наследяване на селектори и други. Sass се трансформира до добре форматиран CSS с помощта на конзолна програма или плъгин към уеб фреймуърк.

Sass има два синтаксиса. Новия синтаксис (въведен от Sass 3) още известен като SCSS е надмножество на CSS3. Това означава, че всеки валиден CSS3 файл е също валиден SCSS. SCSS файловете имат разширение `.scss`.

Втория, по-стар синтаксис е известен като идентифицирания синтаксис (или просто "Sass"). Вдъхновен от краткостта на Haml, той е предназначен за хора, който предпочитат изразителността пред близостта с CSS. Вместо точка и запетая и скоби се използва семантична идентация, за да се обособяват блокове. Въпреки че вече не е основния синтаксис, ще продължи да бъде поддържан. Файловете с идентифиран синтаксис имат разширение `.sass`.

2.3 Ruby on Rails Framework

Ruby on Rails, често съкращаван като Rails или RoR, е уеб фреймуърк с отворен код за програмния език Ruby. Той е предназначен за програмиране по Agile методологията, която се използва от уеб програмисти за ускорено програмиране.

Като много уеб фреймуърци, Ruby on Rails използва Model-View-Controller (MVC) архитектура за организиране на приложението.

Ruby on Rails включва програмки, който правят често срещани задачи в програмирането по-лесни, като например автоматичното генериране на шаблонен код, за по-бързо стартиране на модел или изглед. Също включен е WEBrick - прост уеб сървър писан на Ruby, както и Rake - система за билдване (като make) която се дистрибутира като gem (пакет за Ruby). Заедно с Ruby on Rails тези програмки предоставят базова среда за програмиране.

Ruby on Rails разчита на уеб сървър за да работи. Mongrel е предпочитан пред WEBrick, но също може да се ползва Lighttpd [5], Abyss [6], Apache [7], nginx [8] и много други. От 2008 уеб сървърът Passenger [9] заменя Mongrel като препоръчителна опция. Ruby on Rails е известен също и с използването на javascript библиотеките Prototype [10] и Script.aculo.us за Ajax [11]. В началото Ruby on Rails използва лек SOAP, но по-късно е заменен от REST. От версия 3 Ruby on Rails използва Unobtrusive javascript [12] – техника за разделяне на логиката от изгледа на уеб страницата.

2.4 NodeJs

Целта на Node.js [20] е да предостави лесен начин за програмиране скалируеми, работещи в мрежа програми. Node инструктира операционната система да го уведомява за нови клиенти и после заспива. Ако някой се свърже се изпълнява предварително зададена функция (callback). За всяка връзка се използва съвсем малко памет.

Това е в контраст с по-популярния модел на едновременна работа, в който се използват нишки. Мрежово програмиране имплементирано с нишки е сравнително неефективно и много трудно за използване. Node използва доста по-малко памет при високо натоварване, от колкото системи, които заделят по 2MB за всяка нишка / конекция. Още повече потребителите на Node няма нужда да се притесняват от dead-lock - просто защото няма заключване. Няма функции в Node, която директно изпълнява вход/изход - затова процеса никога не блокира. Поради този факт непрофесионалисти могат да пишат бързи системи.

2.5 jQuery

jQuery [13] е javascript библиотека предвидена да улесни скриптирането на HTML от страна на клиента. Тя е пусната през януари 2006 от Джон Ресиг. Използвана е в 41% от 10000 най-посещавани сайтове. Това е най-популярната javascript библиотека в момента.

jQuery е безплатна, с отворен код, лицензирана под MIT и GNU v2 лицензите. Синтаксиса на jQuery е предвиден да улесни навигацията в документа, селектирането на DOM обекти, създаването на анимации, обработката на събития и разработването на Ajax приложения. jQuery също предоставя възможност за писане на плъгини. С използването на тези възможности програмиста може да създаде абстракции за анимация и взаимодействие от ниско ниво, сложни ефекти както и компоненти от високо ниво поддържащи теми. Това подпомага за създаването на сложни и динамични уеб страници.

2.6 Yui3 library

YUI [14] е javascript библиотека разработвана от Yahoo!. Тя има добре развита модулна система, която позволява да се сваля само кода от библиотеката, който реално се използва. Системата ѝ за събития надгражда тази дефинирана в брауъра и позволява по-лесна обработка и добавяне на специализирани събития.

2.7 Mercurial

Mercurial [15] е дистрибутирана система за управление на сорс код.

Традиционните системи за управление на сорс като Subversion са с типични клиент-сървър архитектури. Те имат централен сървър който пази информацията за различните ревизии на проекта. В контраст Mercurial е напълно дистрибутирана, позволявайки на всеки програмист да има локално копие на цялата история на проекта. Това означава, че програмистите са независими от връзката си с интернет или централния сървър. Добавянето, разклоняването и сливането в проекта са лесни и бързи операции.

2.8 RubyGems

RubyGems [16] е пакетен мениджър за програмния език Ruby, който предоставя стандартен формат за разпространението на Ruby програми и библиотеки във самодостатъчен файлов формат `.gem`. Той предоставя конзолен интерфейс за лесна инсталация и управление на гемове, както и сървър за тяхното разпространение. RubyGems е аналогичен на EasyInstall за програмния език Python. В момента RubyGems е част от стандартната библиотека на Ruby версия 1.9.

2.8.1 База от данни

PostgreSQL [21] е обектно-релационна система за управление на база от данни. Тя е лицензирана под MIT лиценз и следователно е безплатна и с отворен код. Както други системи с отворен код PostgreSQL не се контролира от нито една единствена компания – интернационална група от програмисти и компании разработват продукта.

2.9 Статистика за проекта

- Използвани езици за програмиране
 - javascript
 - Ruby
 - SQL
 - html (генериран чрез haml)
 - css (генериран чрез sass)
- Редове код от началото на проекта - 5000
 - Ruby 1200
 - javascript 3200
- Брой тестове
 - Ruby 21
 - javascript 30
- Брой commits в системата за контрол на версиите - 130

3 Реализация

3.1 Програмен език

Ruby [17] and Javascript [18]. Ruby is a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write.

3.1.1 Уеб сървъри

В проекта използваме два уеб сървъра:

WEBrick

е стандартния web server за малки проекти при Ruby. Използваме го за да сервира страниците и статичното съдържание (картинки, javascript, css).

Node.js

отговаря за комуникацията между различни клиенти посредством `socket.io`. Тази библиотека имплементира 6 различни протокола за двупосочна комуникация на `javascript` клиент и произволен уеб сървър (стандартния `ajax` може да се използва само за поискване на информация - ако сървъра иска да каже нещо на клиента в произволно време трябва да се използва нещо по-сложно). В момента имплементацията поддържа регистриране на потребителите в произволен брой канали, като всяко съобщение изпратено в канал се препраща на всички участници в канала без промяна. В бъдеще може да се имплементира цялата логиката на играта, за да има централизирано място, което да валидира ходовете и резултата преди да бъде изпратен към главния сървър.

3.1.2 База от данни

Използваната база от данни в проекта е PostgreSQL [21]. Схемата използва 4 таблици:

Table "games"

Column	Type	Modifiers
id	integer	not null default
name	character varying(255)	not null
display_name	character varying(255)	
description	text	

Table "users"

Column	Type	Modifiers
id	integer	not null default
game_id	integer	
email	character varying(255)	not null default
encrypted_password	character varying(128)	not null default
password_salt	character varying(255)	not null default
name	character varying(255)	

Table "instances"

Column	Type	Modifiers
id	integer	not null default
game_id	integer	
began	timestamp without time zone	not null
duration	integer	default 0

Table "players"

Column	Type	Modifiers
id	integer	not null default
instance_id	integer	
user_id	integer	
play_order	integer	
score	integer	default 0

3.2 Дизайн

Спазени са основните принципи на Обектно Ориентираното Програмиране [22] - капсулиране, полиморфизъм, наследяване, абстракция, обекти.

3.2.1 Дизайн на игрите

JSG.GameCore.BaseGame Това е базовият клас за всички игри. В него е имплементирана логиката за създаване на играчи. Подготвяне на информацията за реда на игра, спазване на реда на игра, съобщаването на резултата на сървъра. В повечето случаи е достатъчно само да се имплементират методи за валидация на ход и детектор за край на играта. Този клас създава събития за това какъв ход е игран и кой играч е на ход за да може обектите играчи да са наясно с текущото състояние на играта.

JSG.GameCore.LocalUser Представява локално играещ потребител. Той трябва да слуша за събитията свързани с избиране на ход от дъската и да ги предава на обекта игра, който продължава тяхната обработка. Не е предвидено да се наследява.

JSG.GameCore.RemoteGateway Свързващо звено при игри на отдалечени клиенти. Работата е да слуша за събития, касаещи потребители на отдалечени машини (например игра на локални потребители), и да ги препраща към отдалечените потребители посредством уникален канал за тази конкретна играта. Също слуша за действията на отдалечените играчи и съобщава на текущата игра.

JSG.Games.*.Game Всяка игра трябва да имплементира този клас. Той е отговорен за дооформянето на базовия клас като имплементира методи за валидация и детектор за финални състояния.

JSG.Games.*.Board Отговаря за логическото представяне на визуалната част на играта. Класът създава събития за всеки ход, който локалният потребител направи - например избор на поле в таблица. Също дава удобен интерфейс, за да може играта да отразява действията на отдалечените играчи без да се интересува от точната имплементация на визуалните елементи.

JSG.Games.*.BoardUI Отговаря за визуалното представяне на играта. Този клас е от по-ниско ниво от Board – методите му отговарят за пряка манипулация на DOM обекти, макар да представят по-удобен интерфейс, докато Board класа отговаря за отразяването и създаването на ходове. Например може кликването в няколко полета на дъската

да създава един ход – BoardUI класа изпраща събитие за всеки клик, докато в Board се решава кога група събития е ход.

JSG.Games.*.loader Отговаря за зареждането на цялата игра динамично. Тъй като зареждането на всяка игра става от самата нея това намалява връзката между базовата библиотека и имплементатора на играта, което означава, че даже игри, изискващи по-сложно зареждане ще работи без да се променя базовия код на сайта.

JSG.GameCore.gameManager Отговаря за подготовката преди стартирането на игра. Това включва взимане на детайлни данни относно играта и потребителите които участват. Също кода на играта се сваля динамично, в случай че вече не е наличен. В случая с отдалечени потребители – намира и свързва други потребители, които са изявижи желание да играят същата игра. Когато необходимият брой потребители е налице се избира един активен, който регистрира играта на основния сървър и разпраща уникалния идентификатор на останалите потребители. Посредством този уникален идентификатор се осъществява връзка за обмен на ходове по време на играта.

3.2.2 Дизайн на Потребителския интерфейс

JSG.UI.mainUI Отговаря за построяването на целия графичен интерфейс. Това включва създаването на табулярния интерфейс и използването на останалите UI компоненти за да го попълни.

JSG.UI.GameList Отговаря за показване на списък с игри. Този клас се използва както и в списъка за игра, така и в страницата със статистиките.

JSG.Util Този модул включва голямо число помощни функции. Те надграждат стандартната библиотека на `javascript` и се използват във всички останали класове. Включват функции улесняващи обектно-ориентираното програмиране, работата с масиви и хешове, динамично сваляне на съдържание.

JSG.Util.Event Този модул съдържа елементарна имплементация за създаване и слушане на събития. Използва се от почти всички модули, тъй като подобрява преизползването на код чрез намаляване на връзката между създаващия и слушащия събитието.

JSG.Util.HTML Този модул съдържа група функции за лесно и бързо създаване на DOM обекти чрез `javascript`. Чрез него динамичното създаване на HTML е бързо и лесно. Възможно е и слушането на събития произлизащи от DOM обектите.

4 Изкуствен интелект

4.1 SG теория

SG теорията е мощен механизъм за решаване на игри, състоящи се от съвкупност от под-игри независими по между си.

4.1.1 Изисквания

Изисквания за игра, която искаме да решим чрез SG теория:

- игра G
- двама играчи A и B
- играта G е съставена от подигри g_1, g_2, \dots, g_n ²
- $g_i \quad i \in \overline{1, n}$ е игра за двама играчи
- всеки играч на ход има право да избере една подигра $g_i \quad i \in \overline{1, n}$ и да направи ход в нея. След това другия играч е на ход, без да има каквито и да е ограничения за това в коя игра ще играе
- победител е играча победил в последната останала игра.

4.1.2 Теория

Всяка игра протича като последователност от *състояния*. След хода на всеки играч играта преминава в друго състояние. В стандартната теория на игрите всяко състояние бива определено като:

- *печелившо* – играча на ход има печеливша стратегия
- *губещо* – за всеки ход на текущия играч, другия има печеливша стратегия
- *реми* – съществуват ходове само към губещи състояния и други реми състояния

Реми състоянията се различават от губещите по това, че “следвайки” реми състояния се достига до *финално* реми състояние – т.е. такова състояние, което според правилата на играта е реми – никой не печели, а губещо състояние ще доведе до финално губещо – т.е. другия печели по правилата на играта.

sg стойности При игрите разглеждани от SG теорията, няма реми състояния. При тях всяко състояние получава неотрицателна целочислена *sg стойност*. *sg* стойност 0 означава губещо състояние, *sg* стойност $\neq 0$ означава печелившо състояние.

Губещите състояния получават *sg* стойност 0. Ако на дадено състояние s са известни *sg* стойностите на всичките му наследници (състоянията в които може да се стигне) s_1, s_2, \dots, s_p – съответно $sg[s_1], sg[s_2], \dots, sg[s_p]$, то *sg* стойността на s се пресмята по формулата

$$sg[s] = \text{mex}(sg[s_1], sg[s_2], \dots, sg[s_p])$$

където *mex* – *minimal excludante* дефинирано като:

$$\text{mex}(a_1, \dots, a_n) = \min\{0, 1, 2, \dots\} \setminus \{a_1, \dots, a_n\}$$

²допустимо е подигрите да се създават и завършват в хода на играта

От дефиницията на sg стойности лесно се вижда, че губещите състояния имат само ходове към печеливши, а печелившите състояния имат поне един ход към губещо. Това, което прави sg стойностите полезни е възможността за *събиранена* игри. От изискванията по-горе: игрите g_1, g_2, \dots, g_n са сумирани за да се получи G . Ще записваме $g_1 \oplus g_2 \oplus \dots \oplus g_n = G$. Приятното свойство е, че

$$sg[G] = sg[g_1] \oplus sg[g_2] \oplus \dots \oplus sg[g_n]$$

където \oplus е операцията побитово XOR (изключващо или).

доказателство на сумирането на игри За да докажем горното твърдение е достатъчно да покажем, че ако дадени n числа имат XOR s – $a_1 \oplus a_2 \oplus \dots \oplus a_n = s$, то за всяко $0 \leq p \leq s$ съществува $i \in [1, n]$, и $0 \leq a'_i \leq a_i$, такива че $a_1 \oplus a_2 \oplus \dots \oplus a'_i \oplus \dots \oplus a_n = p$.

За целта ще изразим $a_1 \oplus a_2 \oplus \dots \oplus a_{i-1} \oplus a_{i+1} \oplus \dots \oplus a_n = s \oplus a_i$. Сега заместваме във второто равенство и получаваме $a'_i \oplus s \oplus a_i = p$. От тук тривиално $a'_i = (p \oplus s) \oplus a_i$. Сега остава да докажем, че съществува i за което $a'_i < a_i$.

Първо ще покажем, че при $p < s$ $p \oplus s$ има най-висок бит, който е 1 в s с други думи най-високия бит на $p \oplus s$ идва от s . Понеже $p < s$ то в двоичен запис гледано от старши към младши битове p и s съвпадат до един момент, след което се различават – като задължително в най-старшия бит в който се различават s е 1 (а p е 0, иначе $p > s$). Така в $p \oplus s$ старшите битове, в които p и s съвпадат ще бъдат 0, а първия бит в който се различават ще бъде от s .

Сега остава да забележим, че за всеки бит 1 в s съществува i за което a_i има този бит 1. Това е така, защото $s = a_1 \oplus a_2 \oplus \dots \oplus a_n$ и за всеки бит 1 в s има нечетен брой индекси, чиито a -та имат 1 в този бит.

оптимална стратегия В случай, че $sg[G] \neq 0$ използвайки идеята от горното доказателство намираме подигра g_i в която може да направим ход, който да намали sg стойността ѝ по такъв начин, че $sg[G] \oplus sg[g_i] \oplus sg[g'_i] = 0$. Т.е. това ни позволява във всяка игра с ненулева sg стойност да изберем подигра и да направим валиден ход в нея, така че новополученото състояние в G' да има sg стойност 0 – губеща за опонента.

4.1.3 Nim

Nim е математическа стратегическа игра, произхождаща от дълбока древност.

Правила Състои се от n купчинки в които има камъчета. На всеки ход играчът избира произволна купчинка и маха произволен брой камъчета от нея (може и всичките). Печели играча взел последния камък.

Решение Всяка купчинка е независима от останалите. Освен това може лесно да бъдат пресметнати sg стойностите на купчинка с p камъка – $sg[h_p] = p$. Доказва се индуктивно. Сега по формулата може да пресметнем и sg стойността на цялата игра – просто XOR-ваме броя камъни във всичките купчинки – нека резултата е t . Намираме най-високия вдигнат бит в t

и купчинка h_r , чиито брой камъни r има този бит вдигнат. От тук лесно $r \oplus t$ е новия брой камъни които трябва да останат в купчината – т.е. вадим $r - r \oplus t$ камъка.

5 Инсталация

5.1 Инсталиране на зависимости

За да разработим JsGamesci инсталирахме:

- mercurial** Изтегля се от официалния сайт или през пакетен мениджър. Чрез него се изтегля сорса на проекта от <http://iskren.info:3001>.
- PostgreSQL** Изтегля се от официалния сайт или през пакетен мениджър. След инсталацията трябва да се създаде потребител с пълни права както и 3 бази, които му принадлежат.
- jsl** Изтегля се от официалния сайт и се настройва текстовия редактор да го използва при запазване на `javascript` файлове. Показва възможни грешки за да помогне на по-ранното им отстраняване.
- jquery** Изтегля се от официалния сайт.
- jquery.log** Плъгин за jquery за печатане на съобщения в дебъг конзолите.
- qunit** Плъгин за jquery за писане на unit test-ове.
- ruby** Програмният език - от официалния сайт или през пакетен мениджър.
- ruby:minitar** Плъгин за поддръжка на `tar` архиви от Ruby. Инсталира се с `gem` след като е свален от официалния сайт.
- ruby:rails** Може да се инсталира с `gem`.
- rails:jquery** Адаптер на rails.js използващ jQuery. Това е необходимо, тъй като Prototype и jQuery не могат да бъдат използвани едновременно. Инсталира се от официалния сайт ³.
- rails:devise** Плъгин за потребители в rails. Инсталира се с `gem`.
- rails:haml** Плъгин за haml и sass. Инсталира се с `gem`.
- rails:foreigner** Плъгин за добавяне на foreign key в базата данни чрез миграция. Инсталира се с `gem`.
- node** От официални сайт се сваля и компилира. Препоръчително е да се инсталира в директория, в която user-а има права.
- node:npm** От официалния сайт се сваля и инсталира. Чрез него се инсталират всички плъгини за Node.
- node:socket.io** Плъгин за Node. Инсталира се през npm.
- node:json** Плъгин за Node. Инсталира се през npm.
- node:yui3** Плъгин за Node. Инсталира се през npm.

³<https://github.com/rails/jquery-ujs>

6 Декларация за липса на плагиатство

Тази курсова работа е наше дело, като всички изречения, илюстрации и програми от други хора са изрично цитирани. Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция. Разбираме, че ако се установи плагиатство в работата ни ще получим оценка "Слаб".

Зорница Атанасова Костадинова

Искрен Ивов Чернев

Литература

- [1] <http://haml-lang.com/>
- [2] <http://sass-lang.com/>
- [3] <http://en.wikipedia.org/wiki/Css>
- [4] http://en.wikipedia.org/wiki/Web_server
- [5] <http://www.lighttpd.net/>
- [6] <http://www.aprelum.com/>
- [7] <http://www.apache.org/>
- [8] <http://nginx.org/>
- [9] <http://www.modrails.com/>
- [10] <http://www.prototypejs.org/>
- [11] [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))
- [12] http://en.wikipedia.org/wiki/Unobtrusive_JavaScript
- [13] <http://jquery.com>
- [14] <http://developer.yahoo.com/yui/>
- [15] <http://mercurial.selenic.com/>
- [16] <http://rubygems.org/>
- [17] <http://www.ruby-lang.org/en/>
- [18] <http://en.wikipedia.org/wiki/ECMAScript>
- [19] <http://en.wikipedia.org/wiki/WEBrick>
- [20] <http://nodejs.org/>
- [21] <http://www.postgresql.org/>
- [22] http://en.wikipedia.org/wiki/Object-oriented_programming