



**Факултет по математика и информатика**  
**Софийски университет „Свети Климент**  
**Охридски“**

# **Курсов проект**

**Тема:**

## **TicTacToe**

**Изготвил:**

Димитър Керезов

61700

## Съдържание

I.	Играта.....	3
a.	Общи сведения .....	3
b.	История.....	3
c.	Име.....	3
d.	Правила .....	3
II.	Функционален обзор на проекта .....	4
a.	Цялостен обзор.....	4
b.	Алгоритъмът.....	4
i.	Лесна степен на трудност .....	4
ii.	Средна степен на трудност .....	4
iii.	Трудна степен на трудност.....	4
iv.	Невъзможна степен на трудност .....	5
c.	Стратегиите.....	6
i.	Стратегия тип „ръб“ .....	6
ii.	Контриране на стратегия тип „ръб“ .....	7
iii.	Стратегия тип „ъгл“ .....	8
iv.	Контриране на стратегия тип „ъгл“ .....	9
v.	Стратегия тип „център“ .....	10
vi.	Контриране на стратегия тип „център“ .....	12
d.	Край на играта.....	13
III.	Технологичен обзор на проекта .....	14
a.	Технологии, използвани за проекта.....	14
b.	Мобилни браузъри .....	14
c.	Favicon.....	14
IV.	Функционално проектиране на проекта .....	14
a.	Режими на игра.....	14
b.	Картинка накрая на играта.....	15
c.	Бъдещо развитие на проекта.....	15
V.	Потребителско упътване .....	16
a.	Начало на играта.....	16
i.	Игрови режими .....	16
ii.	Потребителско име.....	16
iii.	Ниво на трудност .....	17
b.	Ход на играта.....	17
c.	Край на играта.....	18

VI.	Преносимост .....	19
a.	Android .....	19
i.	Browser на LG G3 телефон .....	19
ii.	Chrome на LG G3 телефон .....	19
iii.	Chrome beta на LG G Pad 8.3 V500 таблет .....	20
iv.	Safari на iPhone 6 с iOS 8.3 .....	21
v.	Safari на iPhone 4 симулатор .....	21
vi.	Internet Explorer на Windows Phone 8.1 емулятор .....	22
vii.	Адрес на проекта .....	22

## I. Играта

### а. Общи сведения

Морският шах (англ. Tic-tac-toe, noughts and crosses, Xs and Os) е игра за двама, най-често играна на лист хартия. При условие, че и двамата играчи играят с оптимална стратегия, винаги се достига равенство. Ето защо играта се играе най-често от деца.

### б. История

Ранен вариант на морския шах е бил игран още по времето на Римската империя около първи век преди Христа. По онова време играта се е наричала “Terni Lapilli” и вместо да се слагат знаци или фигурки до последна възможност, всеки играч е имал по точно три фигурки и е трябвало да се местят, за да продължи играта. Очертанията на игровата дъска на морски шах са намирани из цял Рим. Според някои учени обаче корените на морския шах биха могли да бъдат проследени чак до древен Египет.

### с. Име

Различните имена на играта са скорошни. Първата препратка към името „кръгчета и нули“ (англ. Noughts and crosses), както наричат играта във Великобритания се появява през 1864 година.

Първата напечатана референция за игра, наречена “tick-tack-toe” се появява 1884 година, но името може и да е свързано с вид табла от 1558 година, наречена „tick-tack”. Преименуването на играта от британското “noughts and crosses” към “tic-tac-toe” се случва чак към 20 век.

### д. Правила

Играта се играе от двама играчи на дъска 3x3. Двамата играчи играят със съответно символите „X” и „O”, като се редуват да поставят собствения си символ на някое свободно поле на дъската. Играчът, който успее да сложи своя символ на вертикален, хоризонтален или диагонален ред печели играта.

## II. Функционален обзор на проекта

### a. Цялостен обзор

Проектът “TicTacToe” представлява играта морски шах, реализирана на програмния език JavaScript. Играта има два режима – срещу скрипта и срещу друг играч. Играта срещу друг играч става на едно и също устройство. За целите на играта срещу скрипта разработих алгоритъм, който скрипта следва, за да играе срещу играча.

### b. Алгоритъмът

По време на замислянето на проекта, идеята беше да има три степени на трудност – лесно, средно и трудно. В процес на разработка степента на трудност „трудно“ се обособи като две отделни степени – „трудно“ и „невъзможно“.

#### i. Лесна степен на трудност

Лесната степен на трудност на игра е реализирана изключително примитивно. Скриптът при тази степен на трудност разчита на шанса. Избира си произволно поле от игровата дъска и го бележи със своя символ, в случай, че полето е свободно. Ако полето не е свободно скриптът избира друго поле на случаен принцип. При игра с оптимална стратегия, играчът може да победи скрипта в тази степен на трудност в почти сто процента от игрите.

#### ii. Средна степен на трудност

Средната степен на трудност бе реализирана след всички останали степени на трудност в играта. При нея скриптът взема решение в началото на играта на произволен принцип дали ще се държи като скрипт с трудност „лесно“, или ще се държи като скрипт с трудност „трудно“. При игра с оптимална стратегия, играчът може да победи скрипта в тази степен на трудност в около петдесет процента от игрите.

#### iii. Трудна степен на трудност

Третата степен на трудност на скрипта беше предвидена за последна. Оригиналната идея беше тази степен на трудност да не може да бъде победена. След първоначална разработка бяха открити грешки в скрипта. След поправяне на грешките бе обособена последната степен на трудност, като тази степен бе

оставена с грешките. Важно е да се спомене, че в тази степен на трудност скрипта следва **стратегии**.

Алгоритъмът за тази степен на трудност, който скрипта следва е следният:

1. Ако скриптът може да сложи на някое поле на дъската и това да му донесе победата, то той играе ходът си на това поле
2. Ако скриптът може да блокира победата на противника си, слагайки знака си на някое поле, то той играе ходът си на това поле (например противникът има два символа на един и същ ред и скриптът слага своя символ на третото поле от този ред)
3. Ако скриптът може да контрира противникова **стратегия**, която разпознава, то тогава той играе хода си на най-неудобното за противника поле
4. Ако скрипта е пръв на ход си избира една от трите **стратегии**
5. Скриптът следва стратегията, която е избрал

Една от целите при разработването на тази степен на трудност беше скриптът да не играе винаги по един и същ начин. Така ако скриптът може с еднакъв успех да сложи както на полето най-горе вдясно, така и на това най-долу вляво, то той ще вземе произволно решение къде да играе, за да се избегне чувство на статичност при игра.

При игра с оптимална стратегия, играчът може да победи скрипта в тази степен на трудност в 0,032 процента от игрите или иначе казано – в една игра от тридесет.

#### iv. Невъзможна степен на трудност

Последната степен на трудност е разработена така, че скриптът да не може да бъде победен. Подобно на степента на трудност „трудно“ и тази степен разчита на **стратегии**. Скриптът на тази степен на трудност играе по същия алгоритъм, както и този при сложност „трудно“ с разликата, че проблемите, намерени при ниво „трудно“ тук са отстранени. При игра с

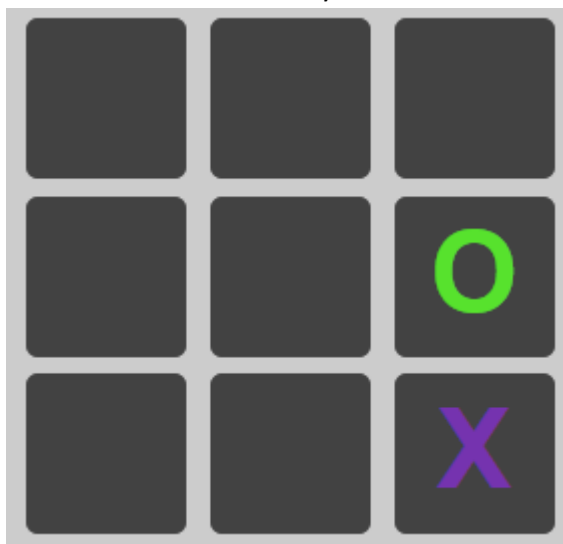
оптимална стратегия, играчът не може да победи скрипта, но може да постигне равенство при сто процента от игрите.

### с. Стратегиите

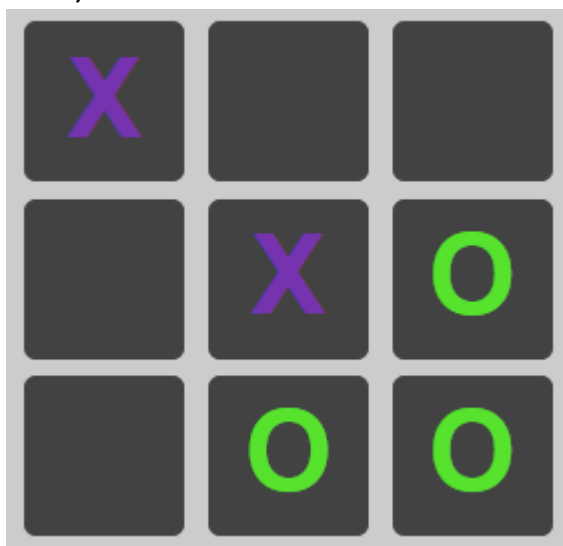
За разработката на алгоритъма, който скрипта следва, за да играе срещу играча, съм използвал собствените си познания за играта морски шах. Тъй като дъската за игра е симетрична (3x3), разделям условно стилът на игра на три стратегии, в зависимост от това къде е игран първият ход – в центъра, в някой ъгъл или на някой ръб. Скриптовите на сложност „трудно“ и „невъзможно“ (както и този на сложност „нормално“ в петдесет процента от случаите) следват някоя от тези стратегии по време на игра. Също така тези скриптове са способни да разпознават тези стратегии в играта на реалните играчи и да ги контрират. Скриптовите избират да следват някоя от стратегиите на произволен принцип с еднакъв шанс за всички стратегии.

#### і. Стратегия тип „ръб“

При нея скриптът е играл на някой ръб. Начална ситуация (скриптът е със зелените O):



Тази стратегия може да бъде избрана от скрипта само ако той играе първи ход с вероятност  $\frac{1}{3}$ . Невъзможно е тази стратегия да бъде избрана в случай, че играчът е направил пръв ход на дъската. Крайната цел на тази стратегия е скриптът да достигне до следното положение (скриптът отново е със зелените O):

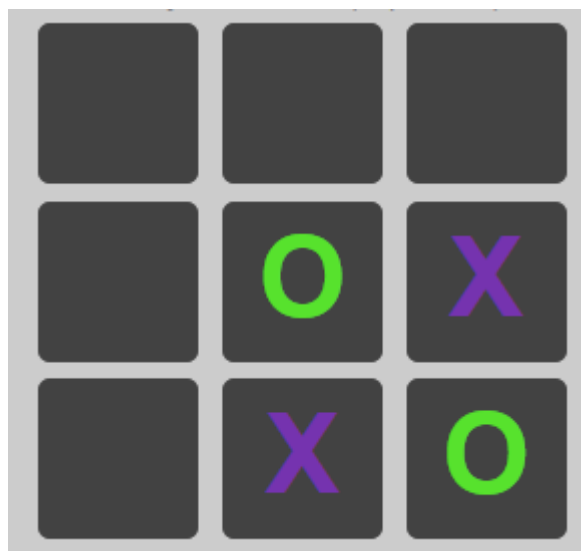


Така скриптът печели независимо къде реши да блокира играчът. При избор на такъв тип стратегия, скриптът слага първия си ход на произволно поле от тип „ръб“ и след това се стреми да постави втори символ на поле от тип „ръб“ на поле, докосващо се до неговото. Накрая завършва с поставяне на символа си в полето в ъгълчето, заградено от двете полета от тип „ръб“.

ii. Контриране на стратегия тип „ръб“

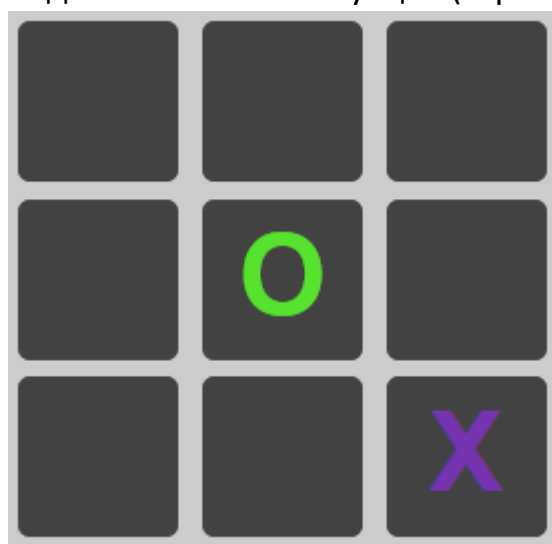
Скриптът е способен да разпознава кога играчът се опитва да използва стратегия тип „ръб“ и да я контрира. В случай, че играчът успее да изиграе два свои хода на две докосващи се полета от тип „ръб“, скриптът блокира, като играе на ъгловото поле между тях. Така скриптът гарантира, че няма да попадне в горната ситуация, а ще достигне равенство. По мои лични наблюдения тази стратегия е доста непопулярна сред реалните играчи. (на картинката долу скриптът отново е със зелените O)



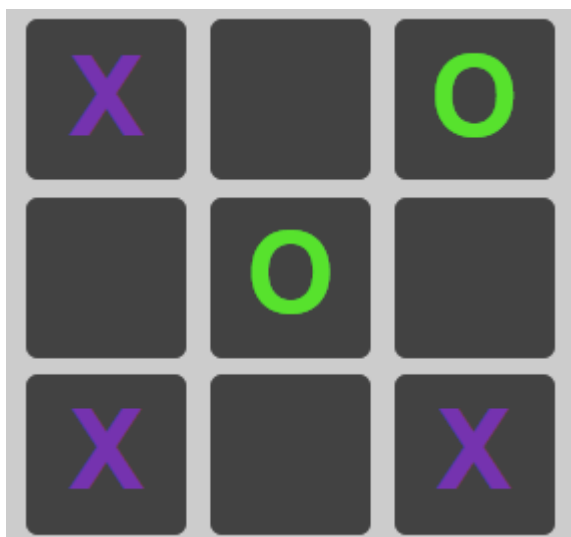


### iii. Стратегия тип „ъгъл“

При тази стратегия скриптът е играл в някой от ъглите на игровата дъска. Начална ситуация (скриптът е с лилавите X):



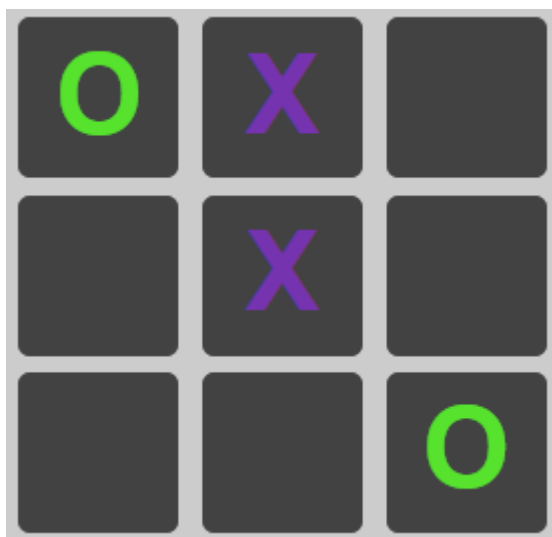
За разлика от стратегията от тип „ръб“, тази може да бъде избрана дори и когато играчът е направил пръв ход. Нещо повече, тази стратегия се избира **винаги** когато играчът е направил своя пръв ход на полето в центъра на дъската. Стратегията може да бъде избрана с вероятност  $\frac{1}{3}$  при начален ход на скрипта и с вероятност  $\frac{1}{2}$  в случай, че играчът е играл пръв на поле от тип „ръб“. Крайната цел на тази стратегия е скриптът да достигне до следното положение (скриптът отново е с лилавите X):



Така отново скриптът постига ситуация, в която няма значение къде играчът блокира, - скриптът печели. При избор на такъв тип стратегия, скриптът слага първия си ход на произволно поле в някой ъгъл на игровата дъска и след това се стреми да постави втори символ в противоположния ъгъл на дъската. В случай, че играчът сгреси своя ход и позволи на скрипта да постави своя знак в трети ъгъл на игровата дъска, се достига показаната горе ситуация.

#### iv. Контриране на стратегия тип „ъгъл“

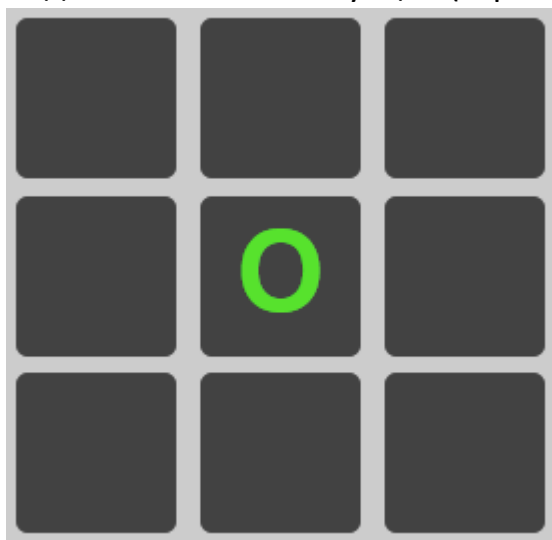
Както и с предната стратегия, скриптът разпознава кога играчът се опитва да използва стратегия от тип „ъгъл“, за да спечели. В такъв случай скрипта не позволява на играча да постави третия си знак в някой от ъглите, като за целта поставя собствения си знак в някое поле от тип „ръб“, принуждавайки играча или да блокира, или да загуби (на картинката долу скриптът отново е с лилавите X)



Решението, за това на кое поле от тип „ръб“ да играе скриптът, се взема на произволен принцип, за да не са еднообразни игрите.

#### v. Стратегия тип „център“

При тази стратегия скриптът е играл в центъра на игровата дъска. Начална ситуация (скриптът е със зелените O):



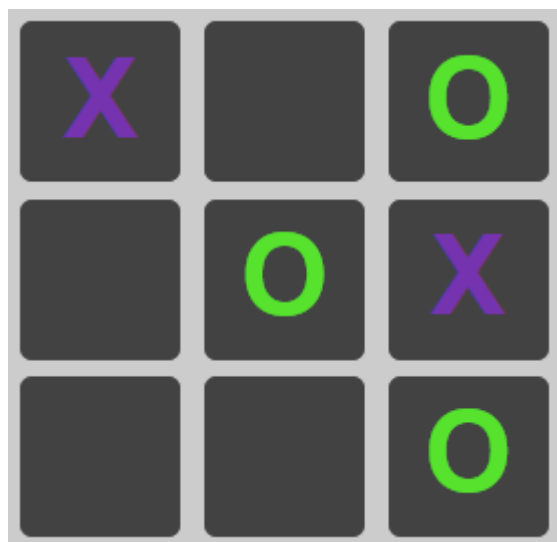
Тази стратегия е безспорно най-често избирана както от скрипта, така и от реални играчи. Скрипта избира тази стратегия с вероятност  $\frac{1}{3}$  при условие, че е пръв на ход, вероятност  $\frac{1}{2}$ , ако играчът е играл пръв на поле от тип „ръб“ и **винаги**, ако играчът е играл на поле в ъгъла на игровата дъска. Крайната цел на скрипта е следната ситуация (скриптът отново е със зелените O):



При избор на такъв тип стратегия първият ход на скрипта е винаги в центъра на дъската, разбира се. След това той се стреми да постави знака си в някои от ъглите (в случай, че са възможни няколко, взема решение на произволен принцип), след което цели поставянето на знака си в друг ъгъл, което да му донесе победата. Важно е да се отбележи, че тази стратегия е може би най-лесната за победа, защото в случай, че играчът реши да играе на някое поле от тип „ръб“ като първи свой ход след като скриптът вече е играл в центъра, то победата на скрипта е гарантирана (скриптът е със зелените O):



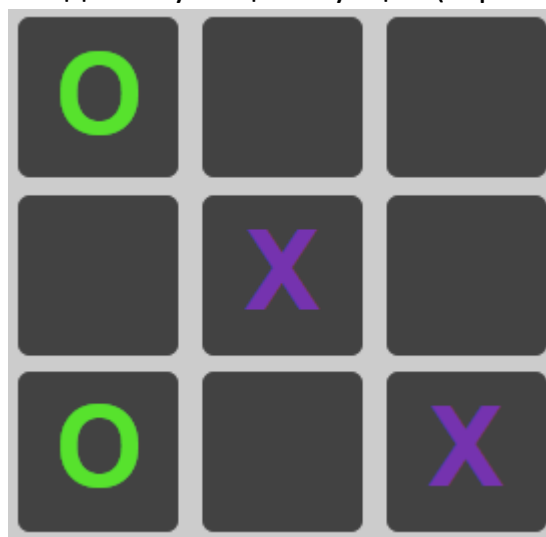
В случай, че играчът сгреша така своя ход, скриптът продължава играта си в някой от ъглите, при което играчът е принуден да блокира и скриптът завършва гамбита, слагайки в другия ъгъл. Крайната ситуация изглежда така (скриптът е със зелените O):



Отново за да не е скована и еднообразна играта, скриптът взема решение на произволен принцип в кой ъгъл на игровата дъска да завърши своя гамбит.

vi. Контриране на стратегия тип „център“

Изключително лесно е разпознаването на стратегия от тип център – ако играчът е играл в центъра, значи използва нея. В такъв случай скриптът реагира подобаващо – избира на произволен принцип един от четирите ъгъла на игровата дъска и прави своя ход там. При разпознаване на стратегия от тип „център“ при играча, скриптът никога не слага като първи свой ход знака си на поле от тип „ръб“, защото това би му гарантирало загуба при оптимална стратегия на играча. В случай, че играчът играе в ъгъл, противоположен на този на скрипта, то тогава скрипта избира някои от останалите ъгли, за да не изпадне в губеща ситуация (скриптът е със зелените O):



Така скрипта кара играчът да блокира, с което разваля стратегията му изцяло. По този начин се гарантира победа или равенство

d. Край на играта

След край на играта започва да свири музика, обозначаваща края на играта, както и се визуализира картинка, която показва статуса на играта. При игра срещу скрипта, тези картинки са:



За съответно загуба, победа и равенство, докато при игра срещу реален играч, картинките са:



В зависимост от това кой играч е спечелил, се визуализира картинка за победа, оцветена с цвета на знака на победилия играч.

Музиката, която звучи след края на играта е свободна за разпространение и е свалена от аудио подразделението на сайта <http://www.newgrounds.com/>.

### III. Технологичен обзор на проекта

#### a. Технологии, използвани за проекта

За реализирането на проекта са използвани технологиите HTML и CSS за визуализацията и JavaScript за логиката и динамичното скриване/показване на елементите. Като помощни библиотеки са взети библиотеката bootstrap за стилизиране и постигане на отзивчив дизайн, наред с библиотеката jQuery за по-лесно манипулиране на HTML DOM елементите.

#### b. Мобилни браузъри

Изиграването на ход от играча се извършва, чрез тъч на някое поле на дъската. След край на играта се визуализира картинка, която пада на дъното на екрана, в случай, че има достъп до обекта "DeviceOrientationEvent" на обекта "window". За улесняване на мобилните браузъри .css и .js файловете са минифицирани. Оригиначните .css и .js файлове също са приложени, с цел по-лесна разработка.

#### c. Favicon

В проекта са заложили 26 различни размера на оригиналната favicon-a. Така се гарантира използването на една от тях, без значение от браузъра и версията, която потребителят използва.

### IV. Функционално проектиране на проекта

#### a. Режими на игра

Първоначалната идея за играта бе само игра срещу скрипта. Стремелът на този проект бе да успее да разработи скрипт, който не може да бъде победен на играта морски шах. Това е и причината нивото на трудност „средно“ да се появи последно – първо бе написана функционалността за скрипта на ниво „невъзможно“, след което алгоритъмът, залагащ на произвола, за ниво на трудност „лесно“ и накрая – ниво на трудност „средно“, като смесица от двете.

След пълното разработване на играта срещу скрипта се роди идеята за игра между двама реални играчи. Тази идея бе реализирана доста лесно, при условие че и двамата играчи играят на едно и също устройство.

#### б. Картинка накрая на играта

След тестване на различни мобилни устройства забелязах, че на някои устройства с по-малки екрани след край на играта картинката, която се визуализира, се намира върху някой от бутоните, като така спира достъпът на потребителя до функционалността, предлагана от съответния бутон. Този проблем бе решен, с помощта на javascript обекта за мобилни браузъри "DeviceOrientationEvent". В случай, че javascript има достъп до такъв обект, то той бива използван, за да се мести картинката на дъното на екрана, в зависимост от ориентацията му. Подобно на приложението [Queens](#), разработено от компанията „MGD“ – GMW Sp. z o. o. Sp. k. Така се гарантира пълен достъп до всички бутони след край на играта, - ако някой бутон е закрит от картинката, потребителят може да я премести, чрез накланяне на устройството си.

#### с. Бъдещо развитие на проекта

Бъдещото развитие на проекта е описано във файла TODO.md, който може да бъде видян както в GitHub, така и в директорията на проекта. Той включва

- Указване по някакъв начин кой е на ход при игра между двама реални играчи. Това може да бъде реализирано чрез подчертаване на името на играча, който е на ход, оцветяването на името му в друг цвят и пр.
- Реализация на сървър на node.js, който да се грижи за съхранението на най-високите резултати на играчи. Може да има различни такива списъци за играчи, победили скрипта на различни нива на трудност



## V. Потребителско упътване

### а. Начало на играта

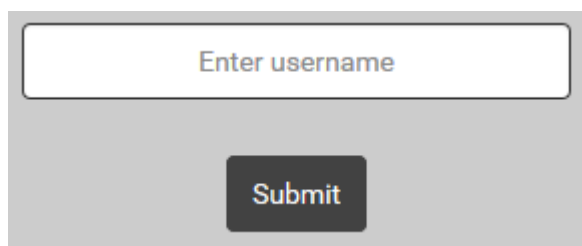
#### i. Игрови режими

В началото на играта на потребителя се дава възможност да избере един от двата режима на игра – игра срещу скрипта (вляво на картинката) и игра срещу друг играч (вдясно на картинката)

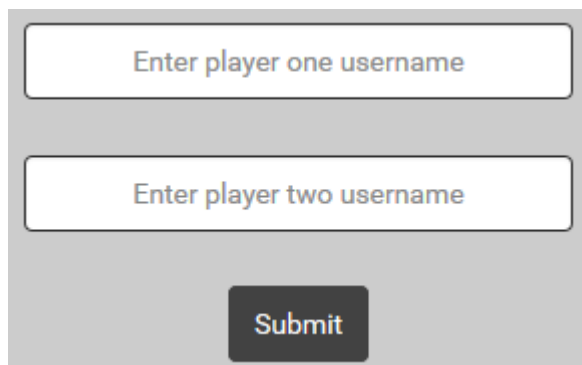


#### ii. Потребителско име

След избиране на режима на игра се появява форма за въвеждане на потребителското име на потребителя. Формата е с едно поле за игра срещу скрипта:

A screenshot of a form for single-player game. It features a single text input field with the placeholder text 'Enter username' in blue. Below the field is a dark grey button with the word 'Submit' in white text.

И с две полета при игра между двама играчи:

A screenshot of a form for two-player game. It features two text input fields stacked vertically. The top field has the placeholder text 'Enter player one username' and the bottom field has 'Enter player two username', both in blue. Below the fields is a dark grey button with the word 'Submit' in white text.

Попълването на тази форма е **незадължително**, в случай, че потребителя не въведе потребителско име му се назначава служебно – *Anonymous*, при игра срещу скрипта и *Player One*, *Player Two*, респективно за първия и втория играч, при игра между двама реални играчи.

Важно е да се отбележи, че към момента тези потребителски имена не се записват никъде, а тяхната функция

е чисто визуална – след въвеждане на потребителско име, по време на игра над игровата дъска се изписва името, което е въведено. Така играта става малко по-персонална.

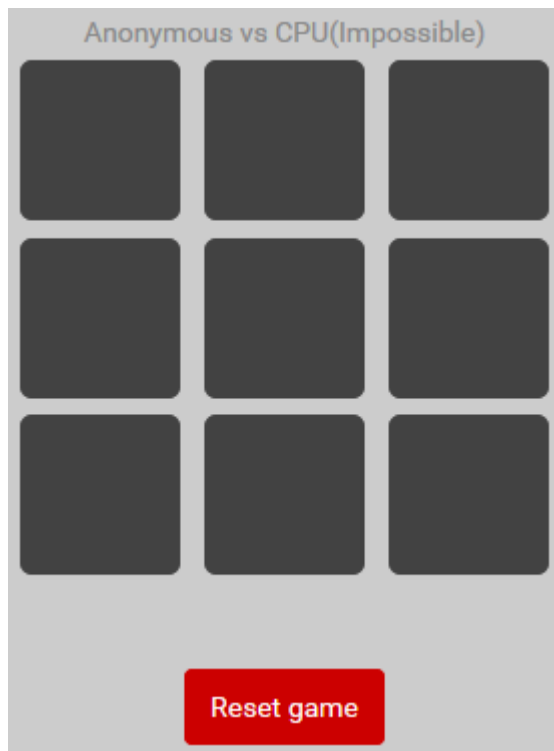
### iii. Ниво на трудност

В случай, че потребителят е избрал игра срещу скрипта, след въвеждане на потребителско име му се дава възможност за избор на ниво на трудност:



### b. Ход на играта

След начало на играта се визуализират игровата дъска, имената на играчите и бутон за рестартиране на играта:



Името на скрипта се формира както следва: CPU(<нивото на трудност>). Бутонът "Reset game" служи за рестартиране на

целия процес на игра – при натискането му играта се връща в начална позиция – преди избор на режим на игра.

Играчът може да направи своя ход чрез докосване на някое поле за игра. При докосване на поле за игра, на което вече е играно не се случва нищо, разбира се.

с. **Край на играта**

При достигане на край на една игра – независимо дали победа, загуба, или равенство, - се визуализира картинка, отразяваща статуса на играта и се появява един допълнителен бутон – “Play again?”



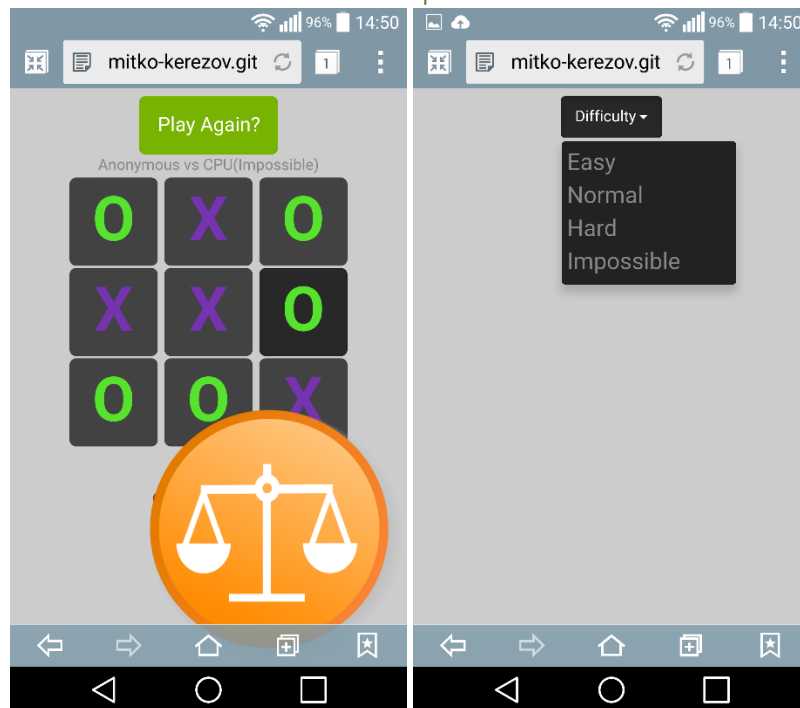
При натискане на бутона “Play again?” започва нова игра със същите настройки – същия режим и същото ниво на трудност. Бутонът “Reset game” остава видим, в случай, че потребителят иска да промени настройките на играта си.

След край на играта полетата на игровата дъска остават неактивни

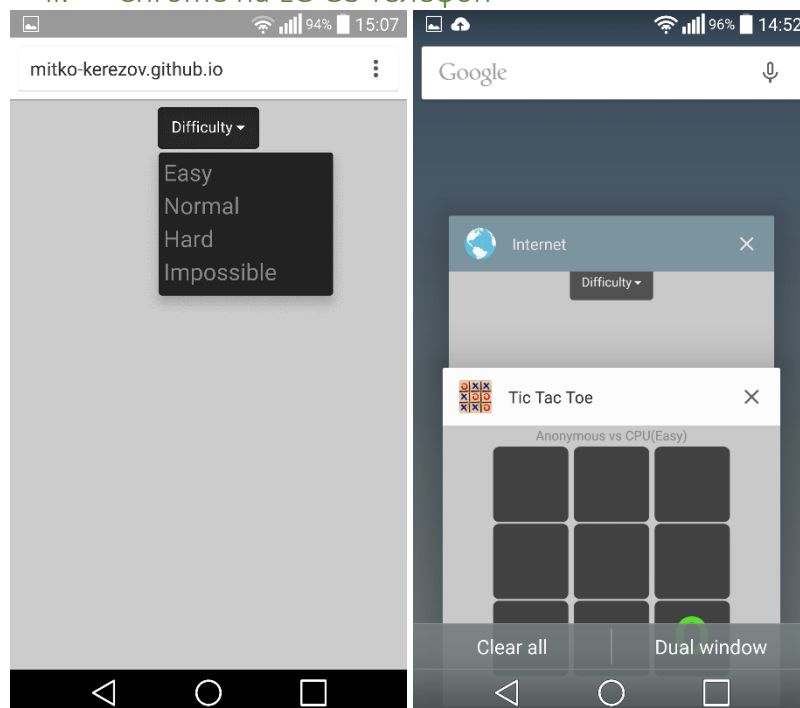
## VI. Преносимост

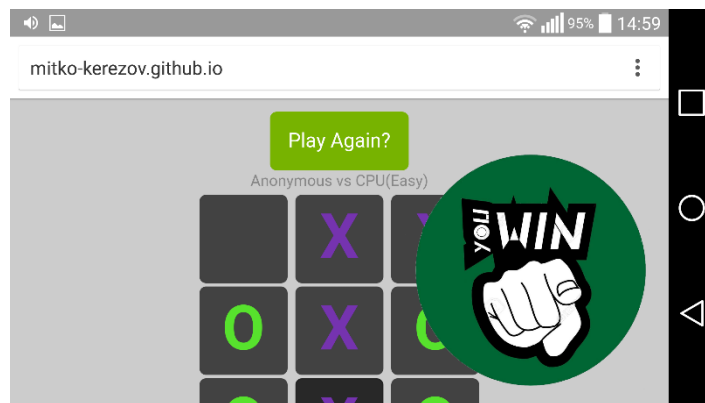
### а. Android

#### i. Browser на LG G3 телефон

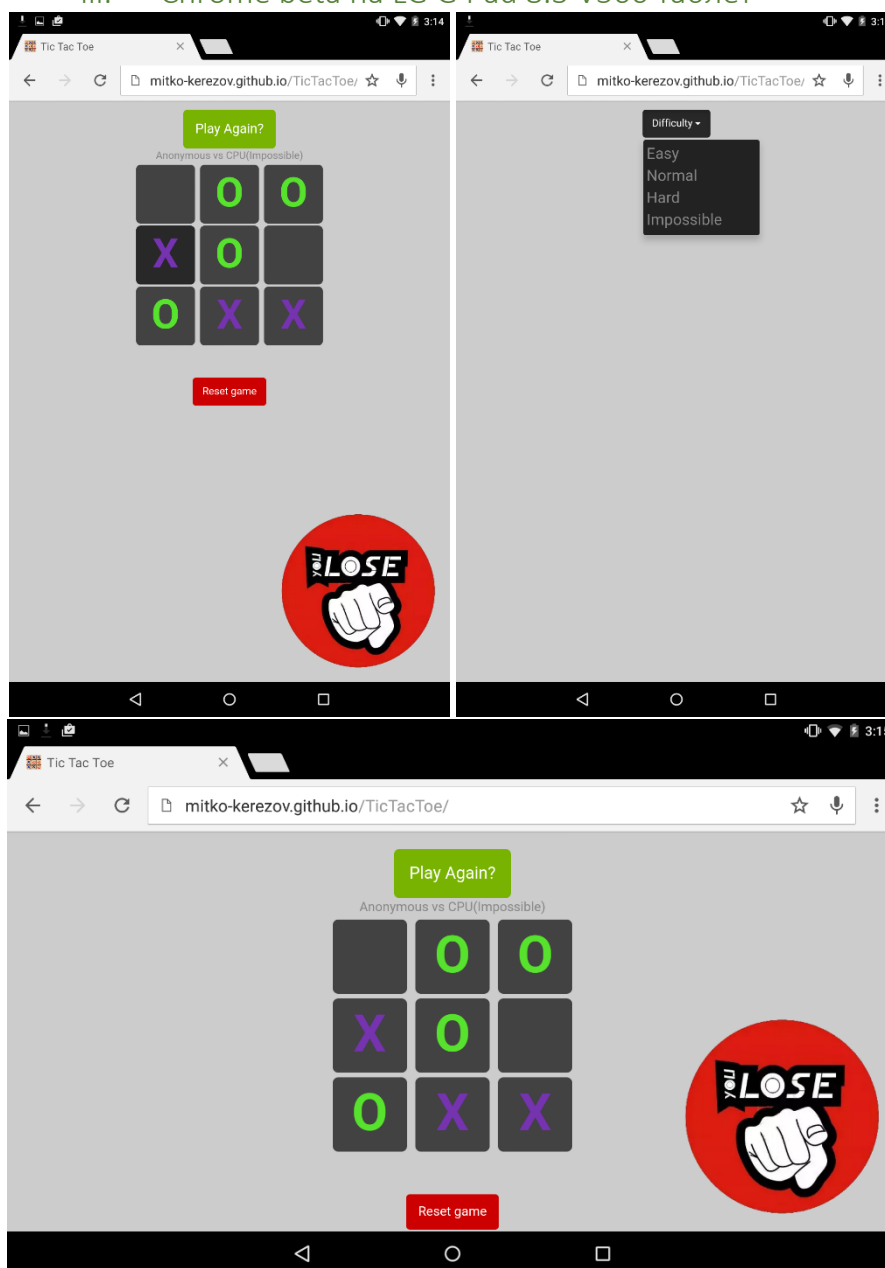


#### ii. Chrome на LG G3 телефон

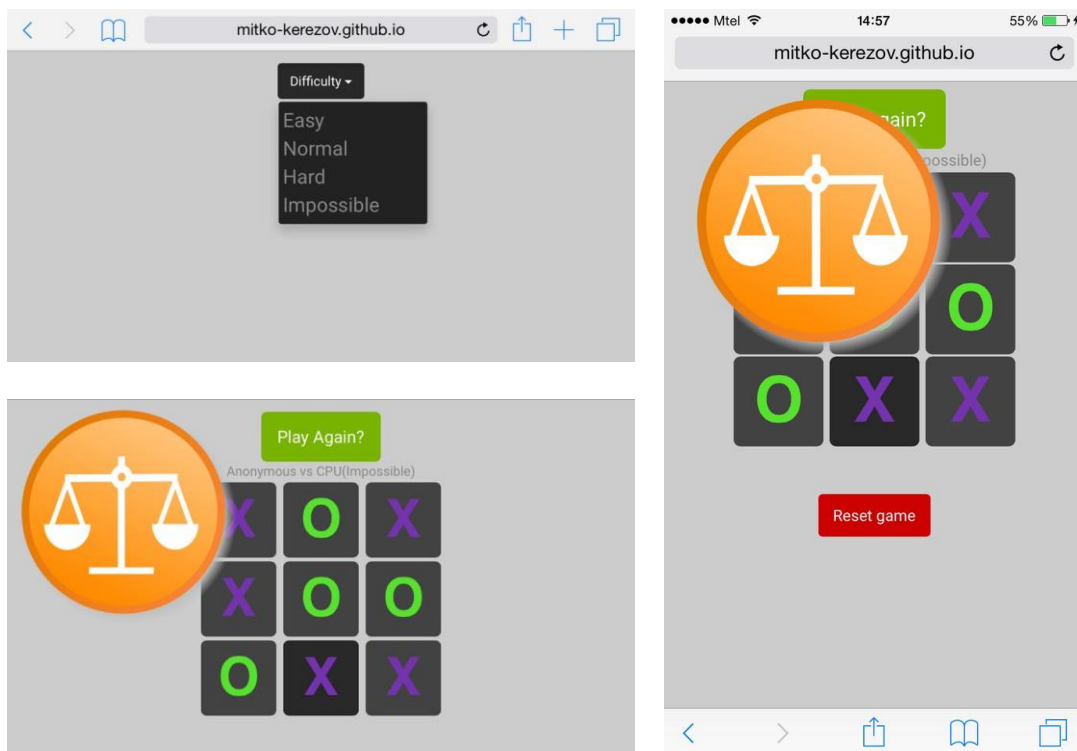




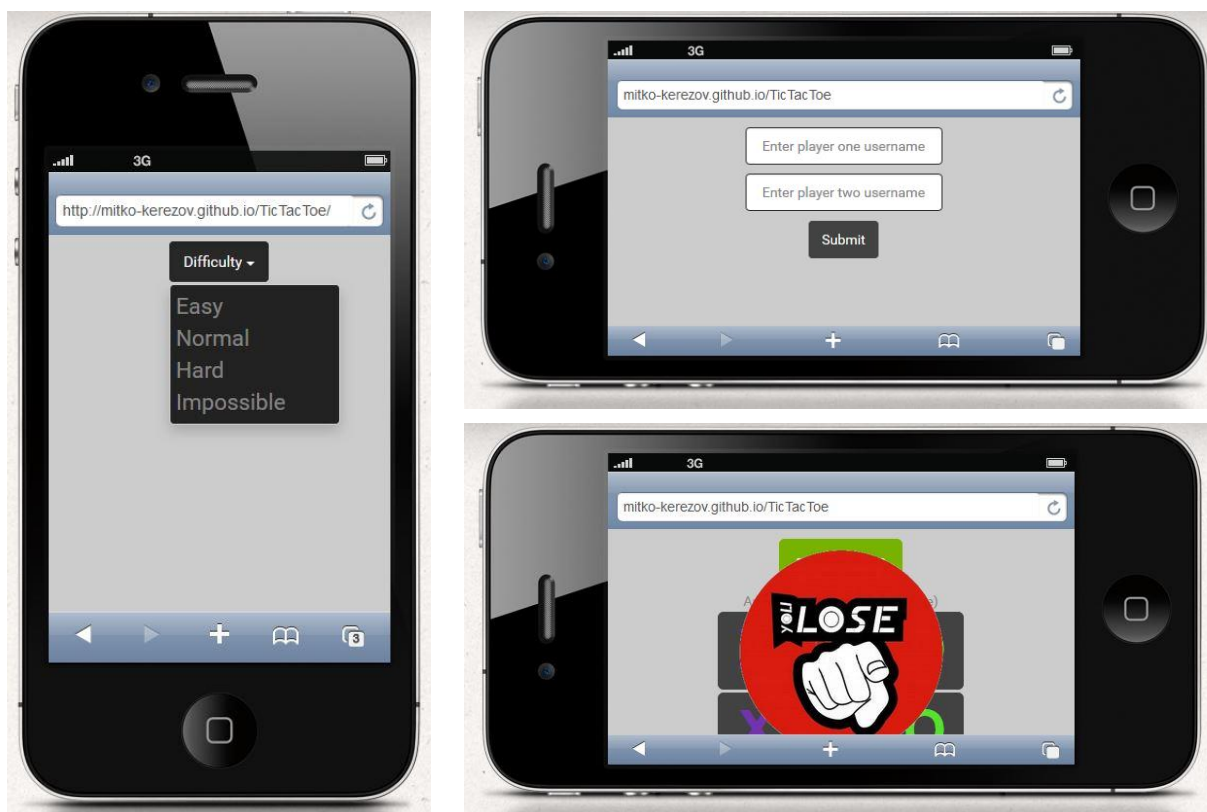
### iii. Chrome beta на LG G Pad 8.3 V500 таблет



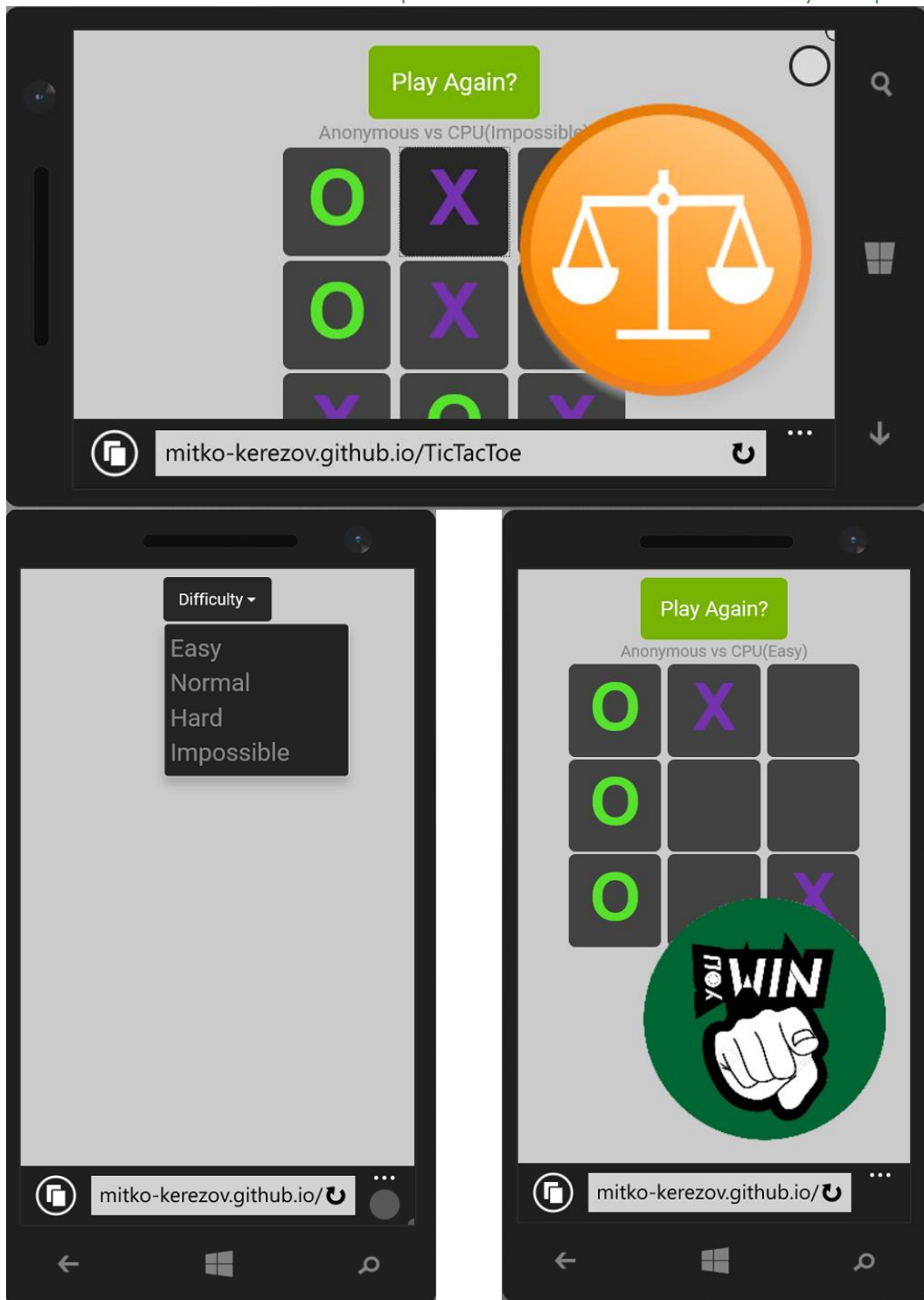
iv. Safari на iPhone 6 с iOS 8.3



v. Safari на iPhone 4 симулятор



vi. Internet Explorer на Windows Phone 8.1 емулятор



vii. Адрес на проекта

Проектът може да бъде достъпен на адрес <http://mitko-kerezov.github.io/TicTacToe/>