

**Snake Game Multilevel**

Realizat de:

Danciu Maria Alexandra

Miron Andreea Gentiana

Grupa: 30232

Disciplina: Inginerie software

Indrumator: Luminita Madalina Marghescu

An universitar: 2020-2021

# Cuprins:

## 1.[Introducere](#_1.Introducere)

## 2.[Descriere generala](#_2.Descriere_generala)

## 3.[Cerinte functionale si instructiuni de rulare](#_3._Cerinte_functionale)

## Bibliografie

## 1.Introducere

Proiectul pe care l-am realizat este o abordare nouă pentru originalul și cunoscutul joc numit „Snake”. La fel ca și în jocul original, obiectul principal este reprezentat de un șarpe care se deplasează pe suprafața unei ferestre în încercarea de a prinde mâncarea. Diferență față de clasicul joc pe telefon este faptul că acesta se joacă direct de pe internet, iar jucătorul are posibilitatea de a alege între mai multe moduri sau mai exact nivele. În cele ce urmează se va prezenta modul în care aceste nivele au fost gândite și construite, dar nu numai.

## 2.Descriere generală

## 2.1 Diagrame

Use case diagram

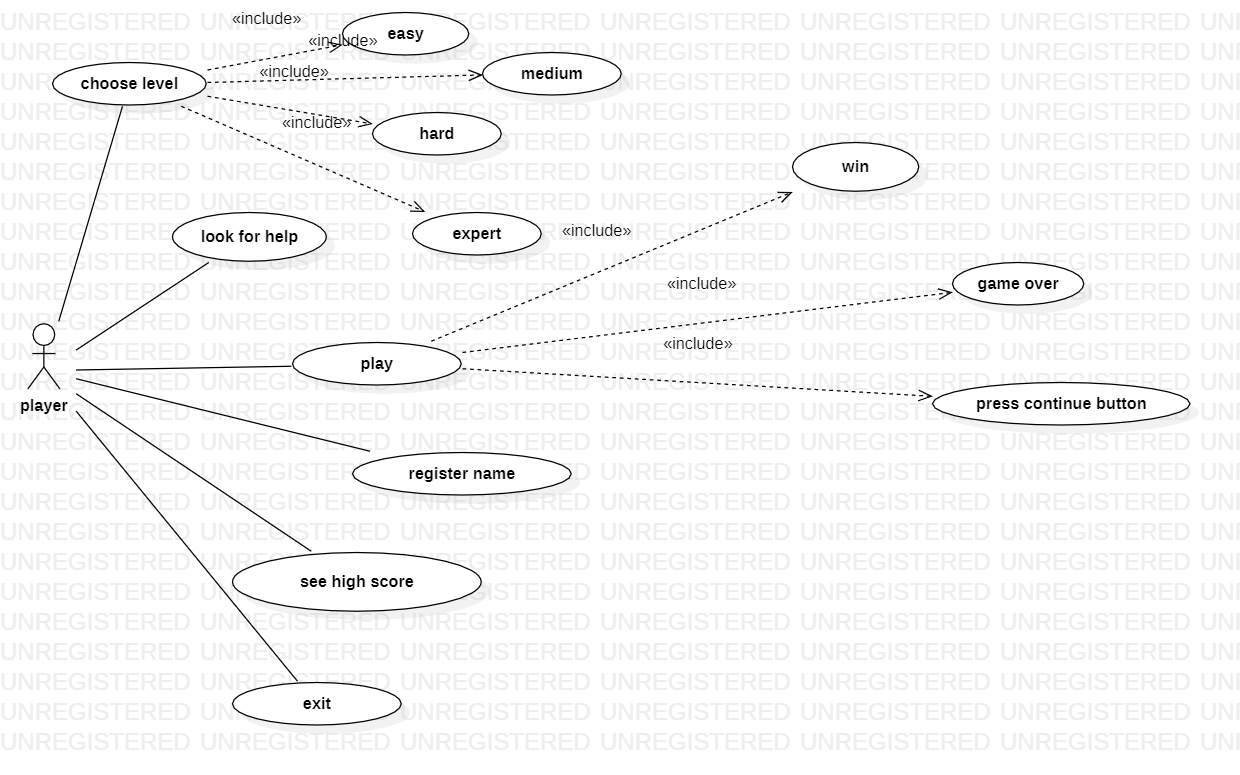


Diagrama de activitati

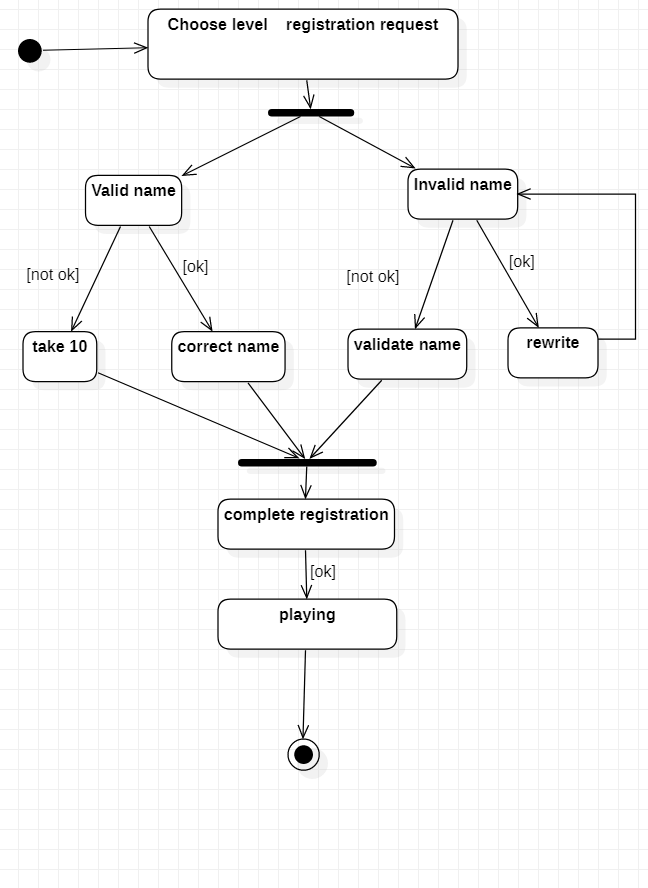
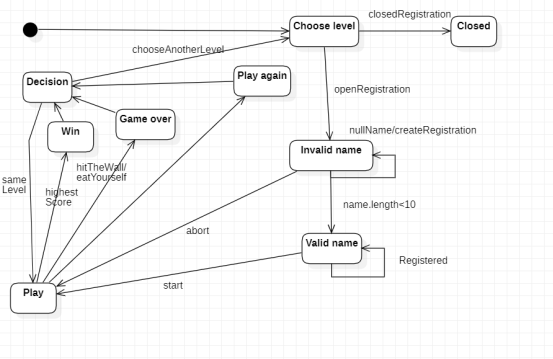


Diagrama de stari cu tranzitii conditionale



## 2.2 Design pattern-uri

## 2.2.1 Constructor

Un constructor este o metodă specială utilizată pentru a inițializa un obiect nou creat odată ce memoria a fost alocată pentru acesta. În JavaScript, deoarece aproape totul este un obiect, cel mai adesea ne interesează constructorii de obiecte.

Constructorii de obiecte sunt utilizați pentru a crea tipuri specifice de obiecte - atât pregătirea obiectului pentru utilizare, cât și acceptarea argumentelor pe care un constructor le poate folosi pentru a seta valorile proprietăților și metodelor membrului atunci când obiectul este creat pentru prima dată.

Acest design pattern se poate observa in intializarile :

Ex. let snakePoint = {'x':'', 'y': ''};

let foodPoint = {'x':'', 'y': ''};

var wall=[];

let wallPoint={'x':'','y':''};

let color = {'snake': '','food': '','wall':''};

etc.

## 2.2.2 Singleton

Modelul Singleton este cunoscut deoarece restricționează instanțierea unei clase la un singur obiect. In mod clasic, modelul Singleton poate fi implementat prin crearea unei clase cu o metodă care creează o nouă instanță a clasei dacă aceasta nu exista. În cazul unei instanțe deja existente, aceasta returnează pur și simplu o referință la acel obiect. Singleton diferă de clasele (sau obiectele) statice, deoarece putem întârzia inițializarea lor, în general, deoarece necesită unele informații care ar putea să nu fie disponibile în timpul inițializării. Nu oferă o modalitate pentru codul care nu este conștient de o referință anterioară pentru a le recupera cu ușurință. Acest lucru se datorează faptului că nici obiectul sau „clasa” nu este returnată de un Singleton, ci o structură.

În JavaScript, Singleton Pattern serveste ca un spațiu de nume de resurse partajate care izolează codul de implementare ,de spațiul de nume global, astfel încât să ofere un singur punct de acces pentru funcții.

Ceea ce face Singleton este accesul global la instanță (în general prin MySingleton.getInstance ()), deoarece nu se poate adresa direct functia MySingleton().Acest lucru este totuși posibil în JavaScript. Trebuie să existe exact o instanță a unei clase și trebuie să fie accesibilă clienților dintr-un punct de acces bine cunoscut.

Este important de remarcat diferența dintre o instanță statică a unei clase (obiect) și un Singleton: în timp ce un Singleton poate fi implementat ca instanță statică, poate fi construit și leneș, fără a fi nevoie de resurse și nici de memorie până când acesta nu este de fapt necesar.

Implementarea design pattern-ului Singleton se poate observa in cadrul variabilei keycodes.Asadar,se realizeaza o instatiere a tastelor si mai multe instante ale aceluiasi obiect,representant individual fiecare tasta utilizata in cadrul jocului:

•Tasta H (keycode=72) deschide o fereastra care afiseaza toate tastele suport disponibile pentru navigare dar si solicitare de informatii

•Tasta Enter(keycode=13) solicita deschiderea unui nou joc.

•Tasta Esc(keycode=27) stagneaza jocul (pauza).

•Tasta Z(keycode=90) creste viteza de deplasare a sarpelui,se apeleaza ulteriori functia de actualizare a vitezei

•Tasta X(keycode=88) decrementeaza viteza de deplasarea a sarpelui, se apeleaza ulteriori functia de actualizare a vitezei

•Tasta A( keycode=65) efectueaza deplasarea la stanga a sarpelui

•Tasta W(keycode=87) realizeaza miscarea in sus a sarpelui

•Tasta D(keycode=68) realizeaza deplasarea la stanga a sarpelui

•Tasta S(keycode=68) realizeaza miscarea in jos a sarpelui

•Tasta R(keycode=68) deschide o fereastra ce afiseaza scorurile

## 2.3 Limbaje si tehnologii folosite

In vederea realizarii acestui proiect, s-a folosit HTML si CSS pentru front-end, adica pentru interfata jocului, iar pentru partea de logica s-a folosit limbajul Javascript. Ca si baza de date, pentru a putea retine numele si scorul jucatorilor s-a folosit SQLite. Mai jos se prezinta modul in care se lucreaza cu baza de date, urmand apoi sa se prezinte arhitectura de baza a jocului si diferitele concepte implementate.

## 2.3.1 Baza de date

Pentru baza de date, s- a declarat o variabila numita „db”. Aceasta variabila se foloseste de functia openDatabase impusa de sistemul de baze de date WEB pentru a crea sau deschide baza de date daca aceasta este deja exstenta. Aceasta metoda primeste ca si parametrii numele bazei de date, numarul versiunii, o descriere a textului si dimensiunea bazei de date.

In continuare, s-a folosit o metoda care ne ofera abilitatea de a controla tranzactiile. Astfel pentru a executa o interogare, avem nevoie sa folosim functia „transaction”. Aceasta functie primeste ca si argument o alta functie in care se executa de fapt interogarea. Pentru a putea executa interogarile am folosit functia „executeSql”. In acest mod, am format interogari pentru crearea tabelei care retine userii, afisarea numelui si a scorului in ordine descrescatoare, inserarea de noi utilizatori, cautarea acestora dupa nume sau actualizarea scorului.

Pentru a crea si introduce un nou utilizator in baza de date am folosit functia „dataBaseNewUser”. In interiorul acestei functii salvam scorul „adevarat” al jocului, scazand cele 3 puncte cu care se incepe initial din cauza lungimii sarpelui, apoi verificam daca jucatorul este adaugat recent in baza de date sau acesta s-a mai jucat pana in acest moment. In primul caz, se adauga in baza de date jucatorul si scorul sau, iar in al doilea doar actualizam scorul acestuia.

In momentul in care se deschide pagina cu nivelul selectat se cere introducerea numelui de utilizator. Acest nume se introduce in fereastra (prompterul) paginii web. In cazul in care dimensiune acestuia este mai mare de 10 caractere, se vor retine doar primele. La fel cum se deschide fereastra pentru introducerea numelui prin „prompt(...)”, folosindu-ne de functia „alert” deschidem o alta fereastra pentru a afisa scorurile.

## 2.3.2 Inceperea jocului

Momentul in care se deschide pagina web este marcat prin evenimentul „onload”. Acesta este folosit pentru a executa un script in momentul in care pagina s-a incarcat. Astfel de fiecare data cand deschidem pagina, vom apela o functie care va crea fereastra corespunzatoare domeniului pe care se poate deplasa sarpele, se va apela functia care cere introducerea numelui, care creeaza si afiseaza fereasta si care apeleaza functia start.

Pentru a desena fereastra jocului, ne folosim de „canvas”. Acest concept este folosit pentru a desena grafica pe o pagina web, dar este nevoie de JavaScript pentru a desena cu adevarat aceasta grafica. Canvasul reprezinta de fapt o suprafata rectangulara de pe pagina web, care este definita unic printr-un id. Ne folosim de functia document.getElementById pentru a recunoaste aceasta suparata, apoi de functia getContext(„2d”) pentru a incepe sa desenam diferite obiecte.

Functia de start este cea care initializeaza sarpele, mancarea, peretii si coordonatele acestora, directia de deplasare si culorile, apeleaza functiile care genereaza mancarea si sarpele in fereastra si seteaza pentru o variabila de tip timer apelarea functiilor de miscare sau generare de mancare o data la o secunda. Avem mai jos un exemplu pentru functia de miscare:

## timer = setInterval(function(){

## move(direction);

## },milliseconds);

## 2.3.3 Logica sarpelui

Functia updateSnake() -initial se verifica daca cel putin un element din variabila sarpe face killedHimself,iar in aceasta situatie se apeleaza dead-sarpele a murit.Daca nu se indeplineste conditia anterioara,se face o actualizare a dimensiunii si apeleaza functiile canvas de desenare a sarpelui

Functia killedHimself(snake)-verifica daca sarpele ajunge sa se suprapuna cu snakePoint

Functia updateSnakeLength()-daca dimensiunea sarpelui este mai mare decat dimensiunea setata initial,se shifteaza corpul si se deseneaza noua dimensiune.Aceasta functie de actualizare a dimensiunii se va apela de fiecare data dup ace sarpele va manca dot-urile plasate pe grid.

Functia hit(snake)-parcurgem cu un for dimensiunea unui perete si verificam fiecare coordonata a peretelui sa coincida cu coordonatele sarpelui.In acest caz,functia hit va returna true,altfel false.

## 2.3.4 Logica mancarii

Pentru nivelele medium, hard si expert mancarea se genereaza intr-o noua locatie o data la 10 secunde, astfel folosindu-ne de functia setInterval din JavaScript, o data la 10 secunde apelam functia food(). Aceasta este de asemenea functia principala, care genereaza in mod aleator un food dot. In cazul in care coordonatele sarpelui ajung in locatia mancarii, adica au aceleasi cordonate, aceasta functie se reapeleaza. In caz contrar, se genereaza aleator o culoare pentru mancare, apoi se deseneaza punctul. Deoarece la fiecare apel al functiei food fereastra este actualizata, trebuie apelata functia pentru desenarea peretilor de fiecare data cand este apelata aceasta functie.

Mancare trebuie de asemenea verificata, pentru a nu fi generata pe o pozitie pe care avem un perete. Acest lucru se verifica in functia foodOverWall, in care pentru fiecare coordonata x si y din perete se verifica egalitatea cu coordonatele mancarii.

## 2.3.5 Logica de miscare

Functia move(direction)-aceasta functie verifica daca directia este fie stanga, dreapta, sus, jos. Totodata, dupa ce se identifica ce tip de directie este, se verifica daca nu ne lovim de peretele grid-ului din directia in care mergem(sa nu iesim din cadru). Daca nu se mai poate deplasa caci ajunge la marginea cadrului de joc, se va apela functia dead(), iar sarpele moare. Daca se mai poate deplasa, se va executa directia solicitata.

Functia position(direction)-verifica ce tip de directie dorim,daca este stanga, la noua scadem din coordonata x de la snakePoint, dimensiunea standard cu care se deplaseaza iar la dreapta adaugam .Daca directia este in sus, scadem din coordonata y a pozitiei sarpelui dimensiunea de deplasare iar daca ne deplasam in jos adaugam dimensiunea. Se va returna noua pozitie.

Functia executeDirection(direction,axis,value) - primeste ca parametru directia(up,down,left,right),axa(x sau y) si valoarea de deplasare.

## 2.3.6 Logica peretilor

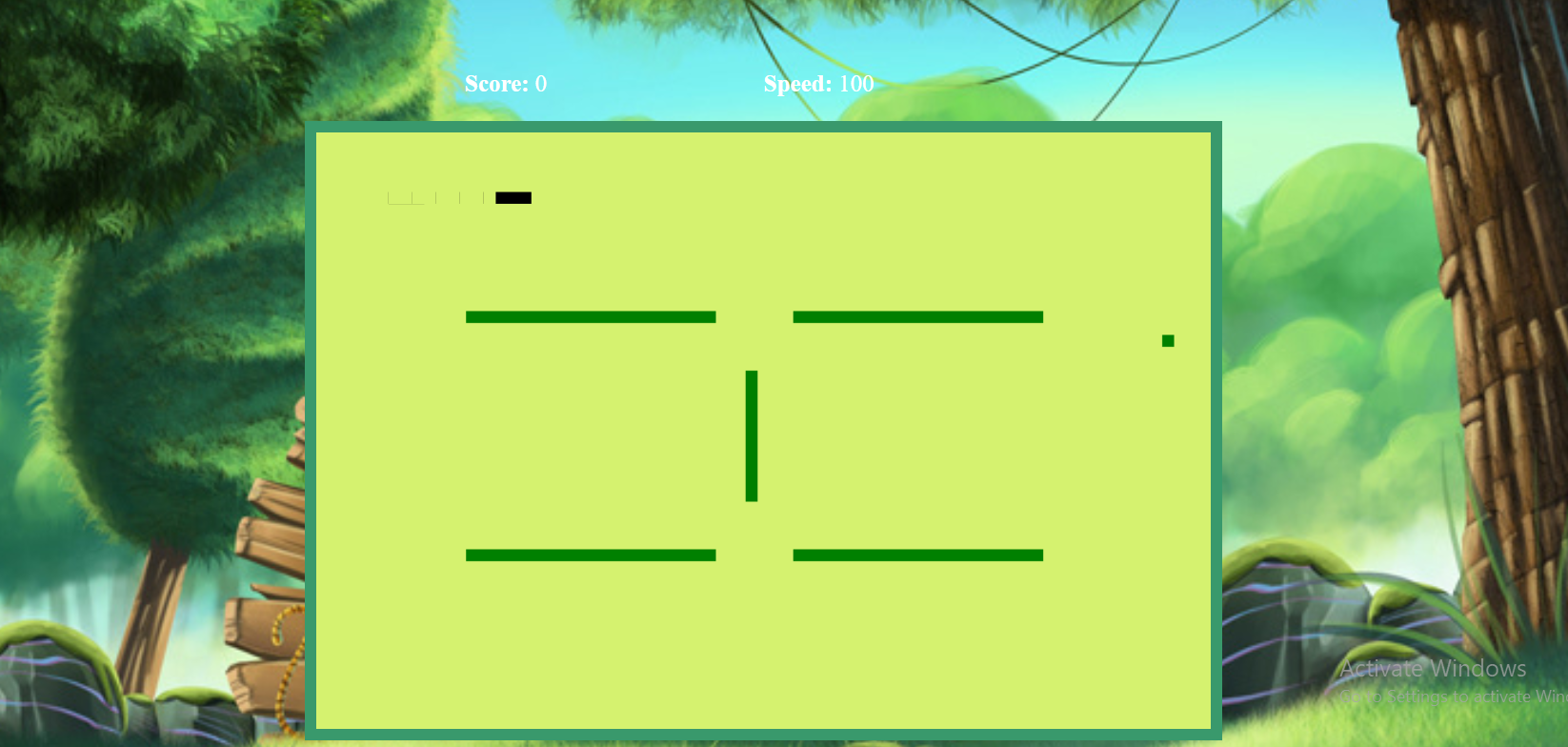
Pentru nivele, in functie de modul in care s-a dorit implementarea acestora s-au declarat peretii si pozitiile de start ale acestora.

Pentru acestia s-a folosit o functie de draw, una pentru miscarea acestora sau pentru generarea random de obstacole.

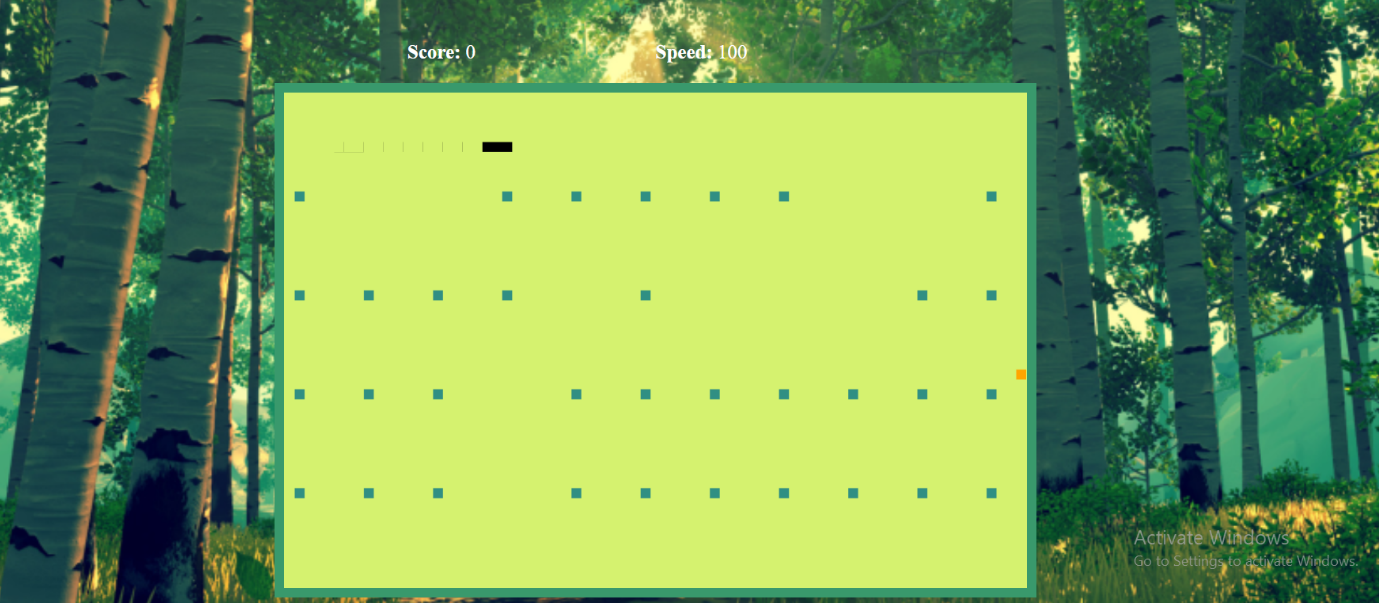
La fel cum este desenat un punct pentru a reprezenta o parte din sarpe, sau un punct care reprezinta mancarea, in acelasi mod este desenat si peretele, prin mai multe puncte simultan. Pentru a misca peretii se foloseste o functie javascript pentru shiftare, apoi se deseneaza punctul in locatia noua si se sterge din cea veche. Desenarea si stergerea se face cu ajutorul functiilor fillRect, fillStyle si clearRect. Fiecare coordonata este salvata in vectorul peretelui, astfel se face wall.push([wallPoint['x'],wallPoint['y']]);

Astfel, harta arata in felul urmator:

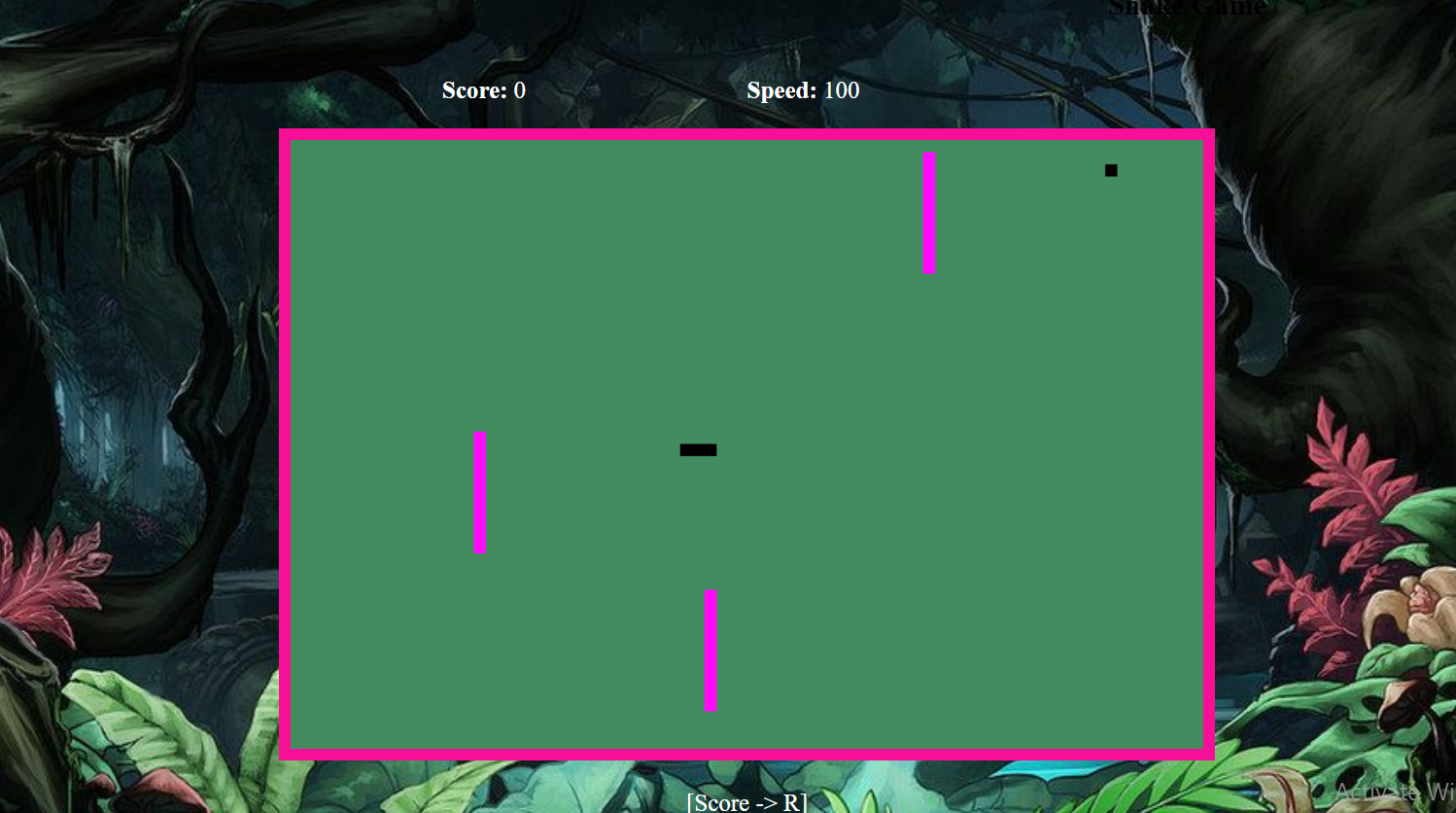
Nivelul mediu



Nivel hard



Nivel expert



De mentionat este faptul ca obstacolele de la nivelul hard se schimba o data la 10 secunde pe pozitii aleatoare, iar peretii miscatori isi cresc viteza relativ la viteza pe care o are sarpele.

## 2.3.7 Alte functii ale jocului

Functia setScore()-functie ce afiseaza scorurile .

Functia setSpeed()-foloseste clearInterval(timer) pentru a opri timer-ul iar mai apoi il reinitializeaza folosind setInterval,la fiecare secunda .Se atribuie noua viteza.

Functia continueGame()-seteaza variabila isPaused(este pauza)-la fals,si se reinitializeaza timer-ul,se reia miscarea prin move(direction) si se deseneaza din nou obstacolele.

Functia dead()-daca sarpele a murit,verificam ulterior daca exista sau nu jucatorul ce a pierdut si facem update la tabela de scor.Se sterge timer-ul jocului.In final se afiseaza o noua pagina ce contine un gif-se semnaleaza ca utilizatorul a pierdut.Pe acest gif,exista 2 optiuni-play again sau alegere alt nivel.

Functia pause()-opreste timer-ul

## 3. Cerinte functionale si instructiuni de rulare

Actorul principal al jocului este utilizatorul. Acesta incepe de pe pagina de start (index.html) de unde isi alege nivelul pe care doreste sa il joace. Dupa ce se deschide fereastra noua este nevoit sa isi treaca numele, altfel nu poate incepe jocul. Dupa ce acesta este introdus, se incepe jocul propriu zis. In cazul in care utilizatorul se loveste de un perete sau se loveste de el insusi, inseamna ca acesta a murit si se deschide o noua pagina care anunta acest lucru. Pe gif-ul respectiv s-a facut o mapare, astfel apasarea pe „continue” o sa reincarce pagina cu acelasi nivel, iar apasarea pe „exit” o sa ne redirectioneze pe pagina principala de unde putem sa schimabam nivelul. Pe parcursul jocului, putem sa vizualizam scorul folosind tasta R sau tastele pe care dorim sa le folosim apasand pe H, jocul fiind pus automat pe pauza sau folosind tasta Esc, sau putem creste si scadea viteza sarpelui folosind tastele Z si X.

O implementare viitoare poate fi controlarea sarpelui prin cursorul mouse-ului, jocul in mod multiplayer, crearea unui mod in care utilizatorii pot interactiona intre ei sau customizarea ferestrei, a mancarii si a sarpelui.

## Bibliografie

<https://stackoverflow.com/questions/1479319/simplest-cleanest-way-to-implement-a-singleton-in-javascript>

<https://www.telerik.com/blogs/design-patterns-in-javascript>

<https://keycode.info/>

<https://www.w3schools.com/html/html5_canvas.asp>

<https://www.tutorialspoint.com/html5/html5_web_sql.htm>

<https://developer.mozilla.org/en-US/docs/Web/API/Document/keypress_event>

<https://www.w3schools.com/jsref/met_win_prompt.asp>

<https://www.w3schools.com/js/js_htmldom_document.asp>

<https://www.youtube.com/watch?v=E6J2fosujWQ>

<https://pngtree.com/so/snake>