Pentru rularea corecta a proeictui este nevoie de plasat fisierul 'bounce' in fisierul 'httdocs' din cadrul xamp-ului, intrucat s-a ales xamp ca server pentru aplicatie.

Scopul proiectului este de a realiza un joc cu tot functionalul necesar

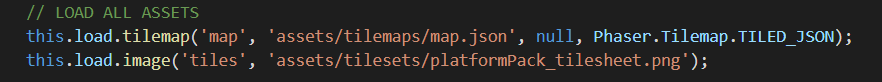
Pentru realizarea scopului s-a ales framework'ul Phaser scris in javascript .

El ne ajuta sa incarcam imaginile necesare petru joc,cat si aplicarea fizicii pentru obiecte.

Deasemnea am imparit jocul in mai multe stari, o 'stare' este modalitatea Phaser care permite impartim proiectul in mai multe fisiere, fiecare cu scopul propriu pentru o modalitate curata si clara de scriere a codului, fiecare stare trebuie sa aiba obligatoriu functia de preload – unde incarcam toate iamginile de care avem nevoie, si functia prima stare prin care trecem cand rulam proiectul este 'Boot.js' in care doar pastram simpla configurare si imedait din interiorul ei se reuleaza :

**'preloader.js'**

In interiorul functiei preload din aceasta stare s-a incarcat incarcat mapa de tip tile realizata in aplicatia tiled, mai jos avem prezentat specificul incarcarii mapei in cadrul proiectului.

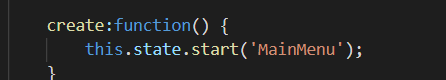


Deasemenea, in mod asemantor se incarca si imaginile penru minge.



Cand se incarca o imagine cu ajutorul functiei ‘loadd’ avem nevoie de doi parametric, primul in cazul nostru ar fi ‘small\_ball’ care e id’ul acestei imagini( ne vom referi la toate imagini ca asset) cand ne vom folosi de e ape parcurs, iar al 2 parametru este drumul cantre imagini din fisierul in care se afla starea actuala. Tooate aseturile se incarca in modul dat si o samai revem aceste detalii pe parcurs.

In functia create din aceasta stare doar rulam starea urmatoare,care ar fi ‘MainMenu’.

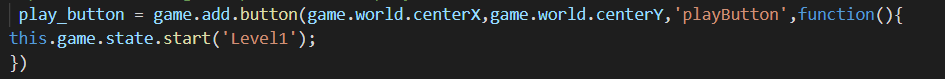


**'MainMenu.js'**



In interiorul functiei ‘preload’ din aceasta stare se incarca asseturile pentru logo([Bounce]) butonul play, cat si imagnii de fundal.

In interiorul functiei ‘create’ au fost pozitionate toate elementele, aveam mai jos prezentat exememplul butonului play, care are rolul de a da start starii urmatoare.

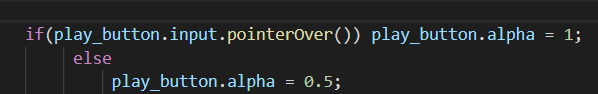


Variabilei globale play\_button i se atribuie un buton, care se arata cu ajutorul imagnii cu id’ul playButton, deasemenea avem alti parametri precum x,y folosite pentru pozitioare, ultimul parametru este o functie unde avem doar un rand de cod care starteaza starea umratoare.

Deasemenea in aceasta stare apare si functuia ‘update’ unde sunt aplasate toate instructiunele care se executa nemarignit pana schimbarea starii, pana acuma nu am avut nevoie de aceasta functie pentru ca taote elementele erau statice, acum insa pentru reda un effect de hoover pe buton play aven nevoie aceasta functie. Insa – si efectul e unul de luminozitate, prezentat in exemplul de mai jos,cand cursorul se afla be buton luminiztatea acestuia e mai mare decat altfel.

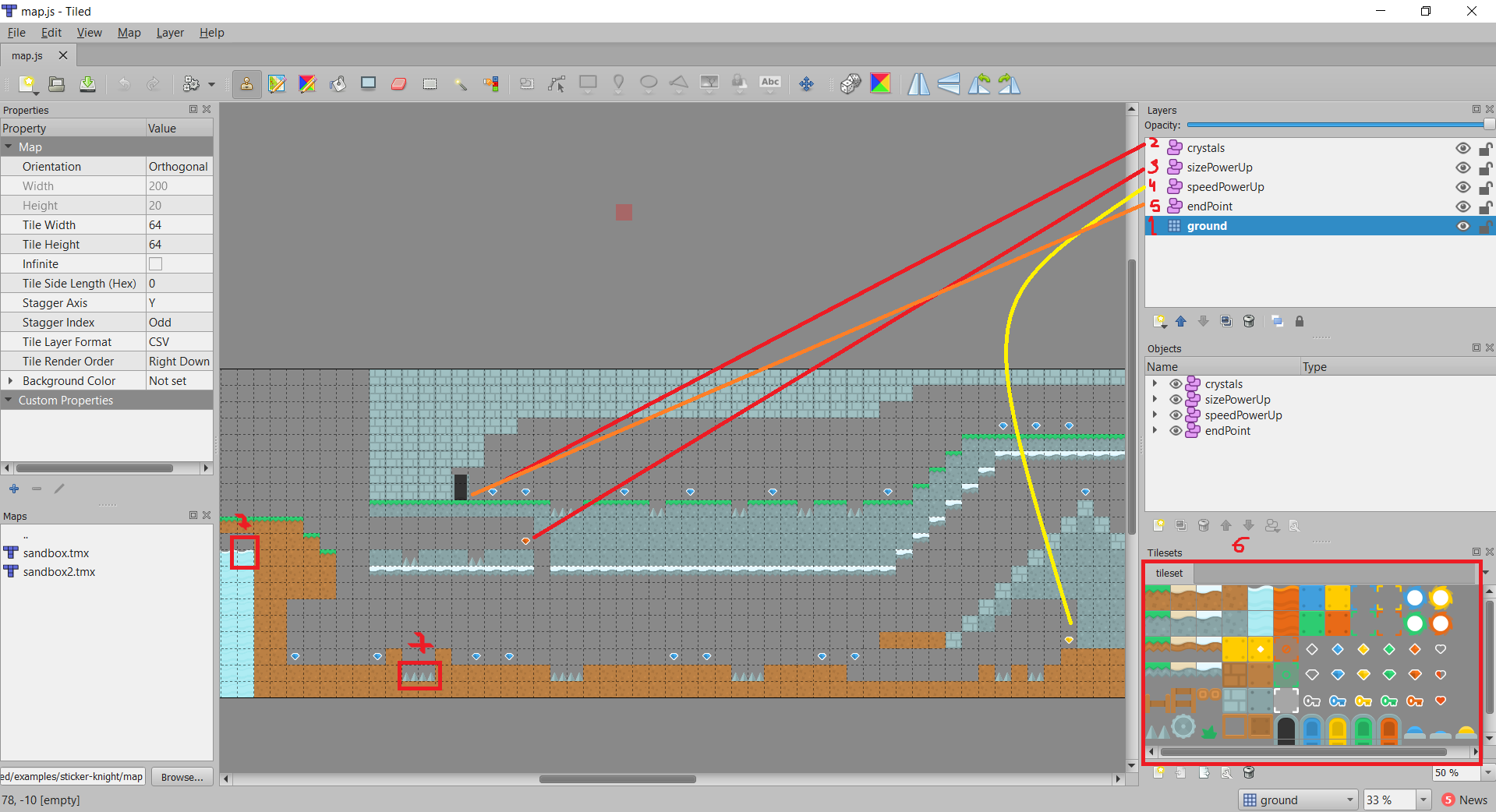
Putem ajunge la acest efetct cu ajutorul proproeitatii alpha in felul urmator.



**'Level1.js'**

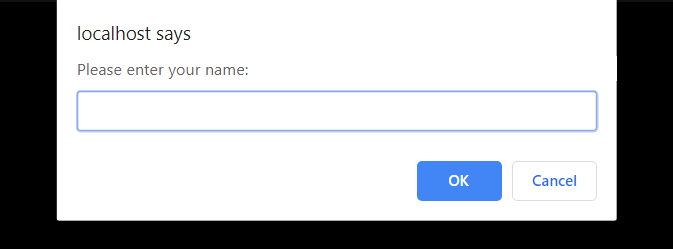
In interiorul acestei stari e setatat taota mecanica jocului.

Intr-ucat multe proprietati si detalii sunte setate din interiorul aplicatiei Tiled, mai joc este prezentat mai detaliat mapa: ne vom referi la ea ca figura1.



Aceasta figura va fi explicata in in intregime pe masura ce trecem prin toate nuantele starii ‘Level1’.

Primul lucru care se intampla este un promt care cere un nume(nickname) pentru a fi inserat mai tarziu in baza de date.



Dupa insearea unui nume, pentru aceasta sesiune va fi ‘test’.



Suntem spawnati pe mapa, stelele de culoare galbene trebuie colectate si fiecare din ele adauga cate un punt la scorul final,deasem pana nu se strang un anumit numar de stele, playerul nu poate finisa nivelul. Pasarea e unul din cele 2 tipurin de inamici din joc, el se deplaseaza in sus si jos si are scopul sa impiedice jucatorul sa treaca mai departe,daca cel din urma intra in contact cu aceasta papsare atunci este aruncat la punctual de inceput, este important de notat ca stele nu se reseteaza, odata colectata scorul va ramace acelasi pana la restartarea aplicatiei. Minge se misca cu ajutorul butoanelor A,D,W,S. De fiecare data cand un buton specific a fost apasat, scadem, sau adunam la velocitatea minge pe axa x,y astfel creand efectul de deplasare.

In figura1 dreptunghiul cu nr 6 reprezinta setul de tile’uri cu care s-a lucrat la crearea mapei,fiecare din ele are dimensiunea de 64x64px, stelele sunt reprezentate ca 2 si reprezinta un layer de obiecte care sunt importate cu ajutorul functiei createFromObjects in felul urmator.

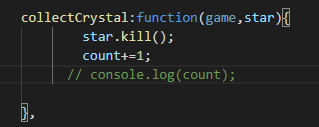


Primul parametru este numele layer’ului de obiecte din aplicatia Tiled numerotat 2 in figura1, la

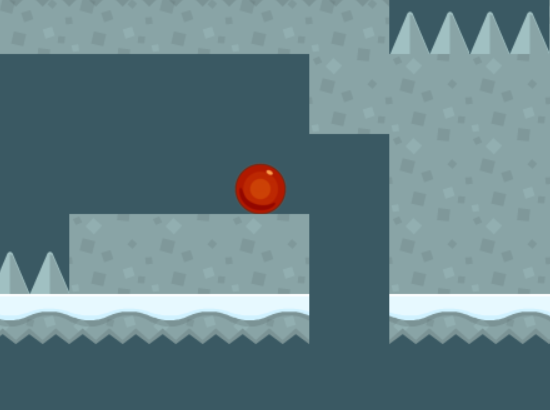
doile reprezinta id’ul global acestor obiecte(50) din configuratia mapei. Mapa este salvata informatul son, iar framework’ul folosit are o intregrare buna cu aplicatia Tiled, din aceasta cauza putem citi aceste informatii direct prin intermediul acestei functii. ‘star’ reprezinta id’ul imagnii ce or reprezenta aceste obiecte on joc. Cand playerul atinca steua galbene ia dispare si un punct se aduna la scorul total.Acest lucru se face printr’ul callback la fiecare overlap intre player si obiect.



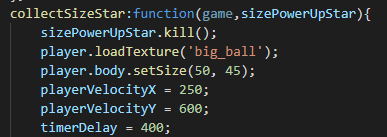
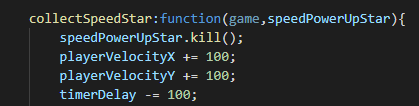
Functia collectCrystal, se apeaza de fiecare data cand functia predefinita overlap se adevereste, primul fapt care se inmpla este distrugerea imaginii care prezinta obiectul cu care avut loc coliziunea, se aduna 1 punct la scor.



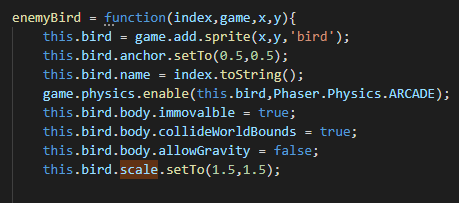
Jocul dispune si de power’upuri sub forma de boost la viteza, si a doua este marirea dimenisunei mingei. In exemplul de mai jos observam ele sunt la fel stele insa distinctive prin culoare, cea rosie pentru marime si cea albastra pentru viteza. E important de notat ca boost’urile sunt in locuri cu risc majorat.

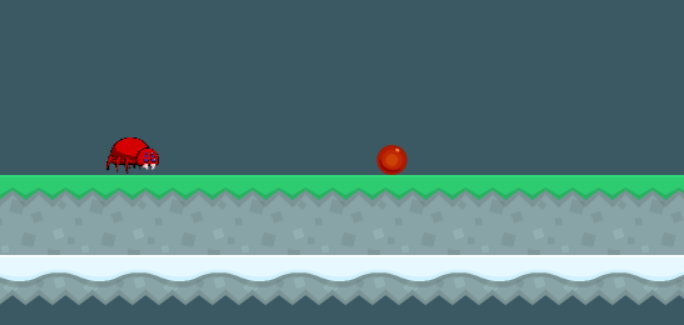
  

Procedura de integrare in joc cat se detectarea colexiunii cu player a boost’urilor e aceeasi ca si stelel de punctaj, se deosebeste insa functiile care se apeala la aceste coliziuni.



Pentru cea de viteza doar adaugam la velicitatea vertical si orizontala, pe cand la acea de dimensiune ii schimbat textura pe o minge mai mare, si deaseamea corpul obiectul se mareste la dimesiunea exacta a texturii. Pentru inamici am creat o clasa, unde s-a specificat toate proprietatilor acestuia, cum ar fi assetul care il v-a reprezenta, cat si atributele fizicii care influenteaza obiectul dat.





Al 2 tip de inamici care reprezinta un paianjen care se misca orizontal la nivelul podelei. La fiecare coliziune cu orice inamic se apeaza o functie resetPlayer care readuce playerul la inceput de nivel, stelel colectate ramam a fi colectate pana la resetare, de asemea playerul e resetat daca intra in contact cu apa sau spinii de pe mapa.

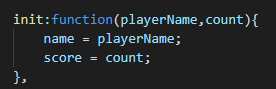


Daca playerul aduna un total de 50 de stele la sfarsit de nivel apare o poarte prin care daca trecem official batem joaca si mergel la starea finala. Daca insa playerul nu a adunat 50 de puncte, poarte nu o sa apara si va fi nevoit sa adune nr de stele necesar.

**‘EndScreen.js’**

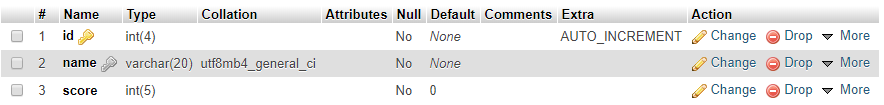


In aceasta stare se afiseaza la ecran numele tastat la iceput si scorul jucatorului. Informatia intre stari se transmite ca parametric ai funtiei init din aceasta stare.



**Baza de date:**

Pentru baza de date noi folosim MySQL. Ea are următoarea structură:



Pentru partea server-side noi folosim doar PHP. Avem în total 2 script-uri: **connect.php** și **persistScore.php**.

**connect.php** răspunde de conexiunea la baza de date. Acolo sunt include configurațiile de bază (username, password, denumirea bazei). Dacă o bază de date cu denumirea **bounce**\_db nu există, atunci ea automat se creează. Totodată, se verifică și dacă există tabelul **user**. Dacă el nu există, se creează automat.

Salvarea scorului în bază de inițiază de **Level1.js**, doar atunci când user-ul termină nivelul cu succes. În cazul dat, se apelează o funcție Javascript: **persistScore**, care construiește și trimite un AJAX request, cu numele și scorul user-ului.

**persistScore.php** răspunde de logica de salvare în baza de date. După ce a primit request-ul, în primul rând se verifică dacă numele user-ului din request deja există în bază. Dacă nu, atunci se execută un insert cu numele și scorul din request. Dacă da, atunci se verifică scorul. Dacă scorul din bază este mai mare decât scorul din request, atunci se ignoră request-ul și nu se face nimic în bază. Dacă scorul din bază este mai mic decât scorul din request, atunci se execută un update, unde se modifică scorul pentru user-ul dat.