**SZABADKAI MŰSZAKI SZAKFŐISKOLA**

**PROJEKTUM**

**Web programozás tantárgyból**

**JELÖLT MENTOR**

**Molnár Aurél dr Zlatko Čović**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SZABADKA, 2019.**

TARTALOM

[A FELADATKÖR MEGHATÁROZÁSA 3](#_Toc12807352)

[KIDOLGOZÁS 4](#_Toc12807353)

[Az index oldal: 4](#_Toc12807354)

[A játék kódja: 4](#_Toc12807355)

[Az ajax lekérést feldolgozó php oldal 19](#_Toc12807356)

[A connection.php 20](#_Toc12807357)

[Admin oldal 20](#_Toc12807358)

[A csv letöltő oldalak 24](#_Toc12807359)

[AZ ADATBÁZIS STRUKTÚRÁJA 26](#_Toc12807360)

[A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA 27](#_Toc12807361)

[A játék oldal 27](#_Toc12807362)

[Az admin oldal 32](#_Toc12807363)

[FELHASZNÁLT IRODALOM 33](#_Toc12807364)

# A FELADATKÖR MEGHATÁROZÁSA

A project feladata egy szabadon választott webes oldal/alkalmazás elkészítése a következő technológiák kötelező felhasználása: HTML, CSS, JavaScript, AJAX, JSON, PHP és MySQL.

A pojektben emellet még fel lehet használni a következő technológiákat: jQuery, XML, API (Google, Twitter, Facebook…), gameQuery, Node.js, Bootstrap...

A projekt még kötelezően kell hogy tartalmazon egy admin oldalat. Illetve egy VCS megoldást is kell használni, a VCS megoldás fajtája szabadon választható volt.

Az általam választott téma, egy webes játék elkészítése, a phaser 3 JavaScript könyvtár segítségével. A kötelező technológiák mellet még felhasználtam egy NodeJS szervert a játék korai fejlesztése során, illetve a JavaScript könyvtárakat a NodeJS szerver npm csomag menedzsere segítségével telepítettem.

A VCS megoldás amit választottam a GitHub.

# KIDOLGOZÁS

## Az index oldal:

Az index oldal magában tartalmazza a canvast amin a játék fut, illetve a játék kódja és a Phaser 3 könyvtár ide van beágyazva. Az egyedi betűtipus is itt van implementálva.

<!DOCTYPE html>

<html>

<head>

<style>

\*{

margin: auto;

padding: 0;

}

@font-face {

font-family: joystix;

src: url('rsrc/joystixmonospace.ttf');

font-weight:400;

font-weight:normal;

}

</style>

<!--<script src="//cdn.jsdelivr.net/npm/phaser@3.16.2/dist/phaser.js"></script>-->

<script src="node\_modules/phaser/dist/phaser.js"></script>

</head>

<body>

<div style="font-family:joystix; position:absolute; left:-1000px; visibility:hidden;">.</div>

<script src="game.js">

</script>

</body>

</html>

## A játék kódja:

A játék kódja. Először is a kezdő változók, konstansok, és fügvények deklarálása.

window.addEventListener('load',inputname);

window.sessionStorage;

const gameWindow={

width:1000,

height:700

};

var timerEvents=[];

var particleEmitters=[];

const config = {

type: Phaser.AUTO,

width: gameWindow.width,

height: gameWindow.height,

pixelArt:true,

physics: {

default: 'arcade',

arcade: {

gravity: { y: 0 },

debug:false

}

},

scene: {

preload: preload,

create: create,

update: update

}

};

const gameRounds={

gameRound:0,

enemytype1:0,

enemytype2:0,

enemytype3:0,

enemytype4:0

};

var Enemies={};

const game = new Phaser.Game(config);

var enemies\_array=[];

const gameState = {

middleScreen:{

x:(gameWindow.width)/2,

y:(gameWindow.height)/2

},

gameover:true,

altcursors:{},

invincible:false,

score:1000

};

A következő részben a preload fügvény látható, itt töltöm be a játék álltal használt képeket, json fájlt.

function preload ()

{

this.load.image('bullet1','rsrc/bullet1.png');

this.load.image('starfield','rsrc/startfield.png');

this.load.image('spaceship','rsrc/spaceship.png');

this.load.image('enemy1','rsrc/enemy1.png');

this.load.image('bullet2','rsrc/bullet2.png');

this.load.image('enemy2','rsrc/enemy2.png');

this.load.image('enemy3','rsrc/enemy3.png');

this.load.image('enemy4','rsrc/enemy4.png');

this.load.image('playerbullet','rsrc/player\_bullet.png');

this.load.json('Enemies','rsrc/Enemies.json');

this.load.image('logo','rsrc/logofinal.png');

this.load.image('particle1','rsrc/particlered.png');

this.load.image('particle2','rsrc/particlepurp.png');

}

A következő részben a create funkcióban a játék objektumait hozom létre, a felíratokat, az elenfelek csoportját, a lövedékeket stb.

Enemies=this.cache.json.get('Enemies');

gameState.emiter=new Phaser.Events.EventEmitter();

gameState.background=this.add.image(0,0,'starfield').setOrigin(0,0).setDisplaySize(gameWindow.width,gameWindow.height);

gameState.player=this.physics.add.sprite(gameState.middleScreen.x,500,'spaceship');

gameState.logo=this.add.image(gameState.middleScreen.x,gameState.middleScreen.y-180,'logo').setDisplaySize(gameWindow.width/2,gameWindow.height/2).setVisible(false);

gameState.player.setCollideWorldBounds(true);

gameState.player.setBounce(0.2);

gameState.player\_bullet=this.physics.add.group();

gameState.cursors=this.input.keyboard.createCursorKeys();

gameState.altcursors.up=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.W);

gameState.altcursors.down=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.S);

gameState.altcursors.left=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.A);

gameState.altcursors.right=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.D);

gameState.altcursors.enter=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.ENTER);

gameState.altcursors.shift=this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.SHIFT);

gameState.circle=this.add.circle(gameState.player.x,gameState.player.y,5,0xff0000);

gameState.hitbox=this.physics.add.existing(gameState.circle,0);

gameState.bullets=this.physics.add.group();

gameState.enemies=this.physics.add.group();

gameState.test1=this.add.text(0, 0, gameState.bullets.countActive(), {fontFamily:'joystix',fontSize: '15px', fill: '#ffffff'});

gameState.scoreboard=this.add.text(gameState.middleScreen.x, gameState.middleScreen.y, '', {fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

gameState.scoreboard.visible=false;

gameState.wintext=this.add.text(gameState.middleScreen.x, gameState.middleScreen.y-100, 'You Won!!!', {fontFamily:'joystix',fontSize: '40px', fill: '#ffffff'}).setOrigin(0.5);

gameState.wintext.visible=false;

gameState.controltext=this.add.text(gameState.middleScreen.x, gameState.middleScreen.y-100, "Constrols\n Movement - WASD or ArrowKeys\n Fire - Space\n Cloaking - Left Shift\n Cloaking makes you\n impervious for a\n short time", {fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

gameState.controltext.visible=false;

A következő részben a játék menüjét létrehozó és müködtető kód látható. Ez a kódrészlet felelős a játék ikon láthatóságáért, a gombok müködtetéséért, illetve mindenért ami a játék menüje csinál.

if(gameState.gameover) {

gameState.player.visible=false;

gameState.hitbox.visible=false;

this.physics.pause();

gameState.logo.visible=true;

gameState.startbutton=this.add.rectangle(gameState.middleScreen.x,gameState.middleScreen.y,200,40,0x0506ee).setOrigin(0.5);

gameState.starttext=this.add.text(gameState.startbutton.x,gameState.startbutton.y,'START',{fontFamily: 'joystix',fontSize:'20px',fill:'#ffffff'}).setOrigin(0.5);

gameState.startbutton.strokeColor=0x000000;

gameState.startbutton.isStroked=true;

gameState.startbutton.lineWidth=2;

gameState.startbutton.setInteractive();

gameState.startbutton.on('pointerup',startgame,this);

gameState.startbutton.on('pointerover',function () {

gameState.startbutton.strokeColor = 0xffffff;

});

gameState.startbutton.on('pointerout',function () {

gameState.startbutton.strokeColor= 0x000000;

});

gameState.highscore=this.add.rectangle(gameState.middleScreen.x,gameState.middleScreen.y+100,200,40,0x0506ee).setOrigin(0.5);

gameState.highscoretext=this.add.text(gameState.startbutton.x,gameState.startbutton.y+100,'Highscores',{fontFamily: 'joystix',fontSize:'20px',fill:'#ffffff'}).setOrigin(0.5);

gameState.highscore.strokeColor=0x000000;

gameState.highscore.isStroked=true;

gameState.highscore.lineWidth=2;

gameState.highscore.setInteractive();

gameState.highscore.on('pointerup',showhighscores,this);

gameState.highscore.on('pointerover',function () {

gameState.highscore.strokeColor = 0xffffff;

});

gameState.highscore.on('pointerout',function () {

gameState.highscore.strokeColor= 0x000000;

});

gameState.changename=this.add.rectangle(gameState.middleScreen.x,gameState.middleScreen.y+200,200,40,0x0506ee).setOrigin(0.5);

gameState.changenametext=this.add.text(gameState.startbutton.x,gameState.startbutton.y+200,'Change name',{fontFamily: 'joystix',fontSize:'20px',fill:'#ffffff'}).setOrigin(0.5);

gameState.changename.strokeColor=0x000000;

gameState.changename.isStroked=true;

gameState.changename.lineWidth=2;

gameState.changename.setInteractive();

gameState.changename.on('pointerup',inputname,this);

gameState.changename.on('pointerover',function () {

gameState.changename.strokeColor = 0xffffff;

});

gameState.changename.on('pointerout',function () {

gameState.changename.strokeColor= 0x000000;

});

gameState.controlscheme=this.add.rectangle(gameState.middleScreen.x,gameState.middleScreen.y+300,200,40,0x0506ee).setOrigin(0.5);

gameState.controlschemetext=this.add.text(gameState.startbutton.x,gameState.startbutton.y+300,'Controls',{fontFamily: 'joystix',fontSize:'20px',fill:'#ffffff'}).setOrigin(0.5);

gameState.controlscheme.strokeColor=0x000000;

gameState.controlscheme.isStroked=true;

gameState.controlscheme.lineWidth=2;

gameState.controlscheme.setInteractive();

gameState.controlscheme.on('pointerup',showcontrols,this);

gameState.controlscheme.on('pointerover',function () {

gameState.controlscheme.strokeColor = 0xffffff;

});

gameState.controlscheme.on('pointerout',function () {

gameState.controlscheme.strokeColor= 0x000000;

});

}

A küdrészlet alul a fentihez tartozik, a menu gombjait müködtető funkciók.

function startgame() {

gameState.logo.visible=false;

gameState.startbutton.visible=false;

gameState.starttext.visible=false;

gameState.highscore.visible=false;

gameState.highscoretext.visible=false;

gameState.changename.visible=false;

gameState.changenametext.visible=false;

gameState.controlscheme.visible=false;

gameState.controlschemetext.visible=false;

gameState.gameover=false;

gameState.player.visible=true;

gameState.hitbox.visible=true;

gameState.player.particleEmitter.active=true;

this.scene.restart();

}

function showhighscores() {

gethighscore();

gameState.logo.visible=false;

gameState.changename.visible=false;

gameState.changenametext.visible=false;

gameState.startbutton.visible=false;

gameState.starttext.visible=false;

gameState.highscore.visible=false;

gameState.highscoretext.visible=false;

gameState.controlscheme.visible=false;

gameState.controlschemetext.visible=false;

gameState.scoreboard.visible=true;

this.input.keyboard.on('keydown-ESC',function () {

this.scene.restart();

},this);

}

function showcontrols() {

gameState.logo.visible=false;

gameState.changename.visible=false;

gameState.changenametext.visible=false;

gameState.startbutton.visible=false;

gameState.starttext.visible=false;

gameState.highscore.visible=false;

gameState.highscoretext.visible=false;

gameState.controlscheme.visible=false;

gameState.controlschemetext.visible=false;

gameState.controltext.visible=true;

this.input.keyboard.on('keydown-ESC',function () {

this.scene.restart();

},this);}

Ez a kódrészlet tatralmazza a collidereket a játékos és az elenfél lövedéke között, a játékos lövedéke és az elenfél között, illetve a játékos lövésének a definiálását. Ez mellet a timerek és a particle emitterek is itt vannak definiálva, és ezzel befejeződik a create funkció.

this.physics.add.overlap(gameState.enemies,gameState.player\_bullet,hitEnemy,null,this);

gameState.hitbox\_collider=this.physics.add.collider(gameState.hitbox,gameState.bullets,GameOver.bind(this));

gameState.emiter.on("Win",GameOver.bind(this));

this.input.keyboard.on('keydown-SPACE',function () {

if (gameState.hitbox\_collider.active) {

let bullet = gameState.player\_bullet.createFromConfig({

setXY: {

x: gameState.player.x,

y: gameState.player.y - 20

}, key: 'playerbullet', repeat: 0, max: 1

});

bullet[0].setScale(0.5);

bullet[0].setAccelerationY(-800);

}

});

//Timers for the enemy shooting

timerEvents.push(this.time.addEvent({

delay:Phaser.Math.Between(5000,8000),

loop:true,

paused:true,

callback:function () {

shootEnemy(1);

},

}));

timerEvents.push(this.time.addEvent({

delay:Phaser.Math.Between(3000,4000),

loop:true,

paused:true,

callback:function () {

shootEnemy(2);

},

}));

timerEvents.push(this.time.addEvent({

delay:Phaser.Math.Between(1000,3000),

loop:true,

paused:true,

callback:function () {

shootEnemy(3);

},

}));

timerEvents.push(this.time.addEvent({

delay:Phaser.Math.Between(1000,2000),

loop:true,

paused:true,

callback:function () {

shootEnemy(4);

},

}));

//Particle emitters

particleEmitters.push(

this.add.particles('particle1').createEmitter({

x: 400,

y: 300,

speed: { min: -800, max: 800 },

angle: { min: 0, max: 360 },

scale: { start: 0.5, end: 0 },

blendMode: 'SCREEN',

active: false,

lifespan: 600,

}));

particleEmitters.push(

this.add.particles('particle2').createEmitter({

x: 400,

y: 300,

speed: { min: -800, max: 800 },

angle: { min: 0, max: 360 },

scale: { start: 0.3, end: 0 },

blendMode: 'SCREEN',

active: false,

lifespan: 300,

}));

gameState.player.particleEmitter=this.add.particles('particle2').createEmitter({

x: 100,

y: 100,

speed: { min: 200, max: 500 },

angle: { min: 45, max: 135 },

scale: { start: 0.3, end: 0 },

blendMode: 'ADD',

active: false,

lifespan: 300,

});

}

A kövektező fügvény az update, itt először is a pontszám és a játékör kimutatása van definiálva, majd pedig a játékos mozgására szolgáló feltételek vannak létrehozva.

function update() {

//Scoreboard and Game round counter

gameState.test1.text='Score:'+gameState.score.toString()+" "+'Round:'+gameRounds.gameRound.toString();

// Hitbox and movement controls and emitter

gameState.player.particleEmitter.setPosition(gameState.player.x,gameState.player.y+32);

gameState.hitbox.x=gameState.player.x;

gameState.hitbox.y=gameState.player.y;

if(gameState.cursors.up.isDown || gameState.altcursors.up.isDown){

gameState.player.setVelocityY(-300);

}

else if(gameState.cursors.down.isDown || gameState.altcursors.down.isDown){

gameState.player.setVelocityY(300);

}

else {

gameState.player.setDragY(1000);

//gameState.player.setVelocityY(0);

}

if(gameState.cursors.right.isDown || gameState.altcursors.right.isDown){

gameState.player.setVelocityX(300);

}

else if(gameState.cursors.left.isDown || gameState.altcursors.left.isDown){

gameState.player.setVelocityX(-300);

}

else

{

gameState.player.setDragX(1000);

//gameState.player.setVelocityX(0);

}

Ebben a kódrészletben van az elenfelek spawnolása megoldva, illetve az elenfelek mozgása és a lövedékek megsemisétése abban az esetben ha elhagyják a képernyő határait. Itt van emellet még megoldva a cloaking mehanika. Ezzel be is fejeződik az update fügvény.

if (gameState.enemies.countActive()===0 && gameState.gameover===false)

{

gameRounds.gameRound+=1;

gameState.player.particleEmitter.active=true;

if (gameRounds.gameRound>1){

gameState.score+=1000;

gameState.starttext=this.add.text(gameState.middleScreen.x, gameState.middleScreen.y, 'Stage:'+gameRounds.gameRound.toString(), {fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

}

timerEvents.forEach(function (item) {

item.paused=true;

});

switch(gameRounds.gameRound){

case 1:

gameRounds.enemytype1=0;

gameRounds.enemytype2=0;

gameRounds.enemytype3=3;

gameRounds.enemytype4=0;

break;

case 2:

gameRounds.enemytype1=1;

gameRounds.enemytype2=0;

gameRounds.enemytype3=2;

gameRounds.enemytype4=0;

break;

case 3:

gameRounds.enemytype1=3;

gameRounds.enemytype2=0;

gameRounds.enemytype3=0;

gameRounds.enemytype4=0;

break;

case 4:

gameRounds.enemytype1=3;

gameRounds.enemytype2=1;

gameRounds.enemytype3=0;

gameRounds.enemytype4=0;

break;

case 5:

gameRounds.enemytype1=2;

gameRounds.enemytype2=1;

gameRounds.enemytype3=3;

gameRounds.enemytype4=0;

break;

case 6:

gameRounds.enemytype1=2;

gameRounds.enemytype2=2;

gameRounds.enemytype3=2;

gameRounds.enemytype4=0;

break;

case 7:

gameRounds.enemytype1=2;

gameRounds.enemytype2=0;

gameRounds.enemytype3=2;

gameRounds.enemytype4=1;

break;

case 8:

gameRounds.enemytype1=0;

gameRounds.enemytype2=0;

gameRounds.enemytype3=0;

gameRounds.enemytype4=0;

gameRounds.gameRound=7;

gameState.wintext.visible=true;

gameState.emiter.emit("Win");

break;

}

gameState.gameover=true;

setTimeout(function(){

spawnEnemy();

gameState.starttext.visible=false;

gameState.gameover=false;

timerEvents.forEach(function (item) {

item.paused=false;

})

},2000);

}

//Various calls for the bullet or the enemy reaching the edge of the screen(Needs some tiny fixes)

Phaser.Actions.Call(gameState.bullets.getChildren(),function (bullet) {

if((bullet.x<0 || bullet.x>gameWindow.width)||(bullet.y<0 ||bullet.y>gameWindow.height))

gameState.bullets.remove(bullet,true,true);

});

Phaser.Actions.Call(gameState.enemies.getChildren(),function (enemy) {

if(enemy.x<60 || enemy.x>gameWindow.width-60) {

enemy.body.velocity.x \*= -1;

enemy.body.acceleration.x\*=-1;

enemy.x -= ((enemy.x - 500) \* 0.01);

}});

Phaser.Actions.Call(gameState.player\_bullet.getChildren(),function (bullet) {

if((bullet.x<0 || bullet.x>gameWindow.width)||(bullet.y<0 ||bullet.y>gameWindow.height))

gameState.player\_bullet.remove(bullet,true,true);

});

if (gameState.altcursors.shift.isDown && gameState.invincible===false){

setInvincible();

window.setTimeout(resetInvincible,1500);

}

}

A következőben jönnek a fügvények amik a kód előző részeiben vannak meghívva.

Ez a füvény halhatatlanná teszi a játékost.

function setInvincible() {

gameState.hitbox\_collider.active=false;

//gameState.cursors.space.enabled=false;

gameState.player.setAlpha(0.5);

gameState.hitbox.setBlendMode(Phaser.BlendModes.SCREEN);

gameState.player.particleEmitter.blendMode=Phaser.BlendModes.SCREEN;

gameState.invincible=true;

gameState.score-=200;

}

Ez a fügvény visszaálitja a halhatatlanságot.

function resetInvincible () {

//gameState.altcursors.space.enabled=true;

gameState.player.setAlpha(1);

gameState.hitbox\_collider.active=true;

//gameState.cursors.space.enabled=true;

gameState.hitbox.setBlendMode(Phaser.BlendModes.NORMAL);

gameState.player.particleEmitter.blendMode=Phaser.BlendModes.ADD;

window.setTimeout(function () {

gameState.invincible=false;

},1000)

}

A következő 3 füvény mind az elenfél lövedékeit helyezik el a képernyőn különböző féleképpen. Az első egy körvonalra helyezi az átadott lövedékeket. A második egy körbe helyezi el a megadott lövedékeket véletlenszerüen. A harmadik pedig egy négyzet éleire teszi az átadott lövedéket, ez a fügvény nincs használva, de meghagytam abban az esetben ha a jövőben használnám.

//Places entitites on a circle around a point

function PlaceGroupAroundCircle(x,y,group,radius=100){

let circle=new Phaser.Geom.Circle(x,y,radius);

Phaser.Actions.PlaceOnCircle(group,circle);

}

//Places entities randomly in a cirlce

function PlaceGroupInCircle(x,y,group,radius=100) {

let circle=new Phaser.Geom.Circle(x,y,radius);

Phaser.Actions.RandomCircle(group,circle);

}

//Places entitites on a rectangle (Not used / useless as of now)

function PlaceGroupAroundRectangle(x,y,group,width=100,height=100)

{

let rectangle = new Phaser.Geom.Rectangle(x,y,width,height);

Phaser.Actions.PlaceOnRectangle(group,rectangle);

}

A következő fügvény akkor hivódik meg ha a játékos lövedéke eltalálja az elenfelet.

function hitEnemy(enemy,player\_bullet) {

player\_bullet.active=false;

gameState.player\_bullet.remove(player\_bullet,true,true);

enemy.setData('health',enemy.getData('health')-1);

enemy.setAlpha(0.5);

enemy.setVelocityY(0);

if (enemy.getData('health')==0) {

gameState.score+=enemy.getData('score');

gameState.enemies.remove(enemy, true, true);

particleEmitters.forEach(function (emitter) {

emitter.setPosition(enemy.x,enemy.y);

emitter.active=true;

emitter.explode(Phaser.Math.Between(15,25));

});

}

else {

particleEmitters.forEach(function (emitter) {

emitter.setPosition(enemy.x,enemy.y);

emitter.active=true;

emitter.explode(Phaser.Math.Between(1,4));

});

}

setTimeout(function () {

enemy.setAlpha(1);

},1000);

}

A következő funkció hozz létre az elenfeleket a képernyőn, a létrehozás egy a képernyő felső felében lévő téglalap ban történik.

function spawnEnemy(){

for(let i=0;i<gameRounds.enemytype1;i++) {

let enemy=gameState.enemies.create(0,0,Enemies.enemy1.sprite\_config.texture);

enemy.setData('number',Enemies.enemy1.number);

enemy.setData('score',Enemies.enemy1.score);

enemy.setData('health',Enemies.enemy1.health);

enemy.setVelocityX(50);

enemy.setAccelerationX(5);

}

for(let i=0;i<gameRounds.enemytype2;i++) {

let enemy=gameState.enemies.create(0,0,Enemies.enemy2.sprite\_config.texture);

enemy.setData('number',Enemies.enemy2.number);

enemy.setData('score',Enemies.enemy2.score);

enemy.setData('health',Enemies.enemy2.health);

enemy.setVelocityX(-50);

enemy.setAccelerationX(-3)

}

for(let i=0;i<gameRounds.enemytype3;i++) {

let enemy=gameState.enemies.create(0,0,Enemies.enemy3.sprite\_config.texture);

enemy.setData('number',Enemies.enemy3.number);

enemy.setData('score',Enemies.enemy3.score);

enemy.setData('health',Enemies.enemy3.health);

enemy.setVelocityX(-80);

enemy.setAccelerationX(-3)

}

for(let i=0;i<gameRounds.enemytype4;i++) {

let enemy=gameState.enemies.create(0,0,Enemies.enemy4.sprite\_config.texture);

enemy.setData('number',Enemies.enemy4.number);

enemy.setData('score',Enemies.enemy4.score);

enemy.setData('health',Enemies.enemy4.health);

enemy.setScale(2);

enemy.setVelocityX(40);

enemy.setAccelerationX(5);

enemy.setAngularVelocity(40);

}

let rect=new Phaser.Geom.Rectangle(60,16,800,200);

Phaser.Actions.RandomRectangle(gameState.enemies.getChildren(),rect);

}

A következő 2 funkció a lövedékek lövését végzi el. Az első egy középpontoh képest körkörösen lövi ki a lövedékeket, a másik pedig a játékos jelenlegi poziciójára lövi a lövedékeket.

function shootCircle(enemy,bullets,mode=1) {

for(let i=0;i<bullets.length;i++){

let bullet=bullets[i];

bullet.collideWorldBounds=true;

bullet.body.onWorldBounds=true;

if(mode==1) {

bullet.setCircle(8);

}

else {

bullet.setCircle(16);

}

bullet.setAcceleration((bullet.x-enemy.x)\*10,(bullet.y-enemy.y)\*10);

}

}

//Function to shoot bullets right at the players current position

function ShootTrackingBullets(enemy,bullets,placebullets=false) {

for(let i=0;i<bullets.length;i++) {

let bullet=bullets[i];

if(placebullets) {

bullet.x = enemy.x;

bullet.y = enemy.y + 10;

}

bullet.setCircle(8);

bullet.setAcceleration((gameState.player.x-bullet.x),(gameState.player.y-bullet.y));

}

}

A következő fügvény létrehozza a lövedékeket az elenfelekhez, és meghivja a lövő fügvényeket.

function shootEnemy(number) {

gameState.score-=10;

if(gameState.enemies.countActive()>0) {

let E = gameState.enemies.getChildren();

for (let i = 0; i < E.length;i++) {

let current\_enemy = E[i];

if(current\_enemy.getData('number')==Enemies.enemy1.number && current\_enemy.getData('number')==number ){

let bullets = gameState.bullets.createMultiple(Enemies.enemy1.bullet\_config);

PlaceGroupInCircle(current\_enemy.x, current\_enemy.y+10, bullets, 20);

shootCircle(current\_enemy, bullets);

}

else if (current\_enemy.getData('number')==Enemies.enemy2.number && current\_enemy.getData('number')==number) {

let bullets = gameState.bullets.createMultiple(Enemies.enemy2.bullet\_config);

PlaceGroupAroundCircle(current\_enemy.x, current\_enemy.y, bullets, 20);

shootCircle(current\_enemy, bullets,2);

}

else if(current\_enemy.getData('number')==Enemies.enemy3.number && current\_enemy.getData('number')==number ){

let bullets = gameState.bullets.createMultiple(Enemies.enemy3.bullet\_config);

ShootTrackingBullets(current\_enemy, bullets,true);

}

else if(current\_enemy.getData('number')==Enemies.enemy4.number && current\_enemy.getData('number')==number ){

let k=Phaser.Math.Between(1,3);

let bullets;

switch (k) {

case 1:

bullets = gameState.bullets.createMultiple(Enemies.enemy4.bullet\_config);

PlaceGroupInCircle(current\_enemy.x, current\_enemy.y+10, bullets, 40);

shootCircle(current\_enemy, bullets, 2);

break;

case 2:

bullets = gameState.bullets.createMultiple(Enemies.enemy4.bullet\_config);

PlaceGroupAroundCircle(current\_enemy.x, current\_enemy.y, bullets, 40);

shootCircle(current\_enemy, bullets,2);

break;

case 3:

bullets = gameState.bullets.createMultiple(Enemies.enemy4.bullet\_config);

PlaceGroupInCircle(current\_enemy.x, current\_enemy.y+10, bullets, 50);

ShootTrackingBullets(current\_enemy, bullets,);

break;

}

}

}

}

}

A következő 3 funkció a név és a pontszámot kezeli. Az első funkció a játékos nevének bevitelét valósítja meg. A második fügvény a pontszámot AJAX módon elküldi. A harmadik pedig AJAX módon lekéri a top 10 pontszámot.

function inputname() {

let defaultname='Unknown';

if(sessionStorage.getItem('playername'))

defaultname=sessionStorage.getItem('playername');

let name;

let person = prompt("Please enter your name:", defaultname.toString());

if (person == null || person == "") {

name = "Unknown";

}

else {

name = person;

}

gameState.player\_name=name;

sessionStorage.setItem('playername',name);

}

//Function that sends the score to the database

function sendhighscore(){

let xhttp= new XMLHttpRequest();

xhttp.open('GET','process.php?name='+gameState.player\_name+'&score='+gameState.score+'&stage='+gameRounds.gameRound,true);

xhttp.send();

}

//Function that gets the top 10 scores from the database

function gethighscore(){

let xhttp=new XMLHttpRequest();

xhttp.open('GET','process.php?t='+Math.random(),true);

xhttp.send();

xhttp.onreadystatechange = function () {

if (xhttp.status==200) {

gameState.scoreboard.setText(this.responseText);

}

}

}

Az utolsó fügvény a játék befejező fügvénye, megállitja a játékot, elküldi a pontszámot, visszaálítja a pontszámot , és kiírja a Game Over szöveget.

function GameOver() {

this.physics.pause();

sendhighscore();

gameRounds.gameRound=0;

timerEvents.forEach(function (item) {

item.destroy();

});

timerEvents=[];

gameState.gameover=true;

gameState.player.particleEmitter.active=false;

this.add.text(gameState.middleScreen.x,gameState.middleScreen.y-30,'Score:'+gameState.score.toString(),{fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

this.add.text(gameState.middleScreen.x, gameState.middleScreen.y, 'Game Over', {fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

this.add.text(gameState.middleScreen.x, gameState.middleScreen.y+30, 'Click or Press Enter to Restart', {fontFamily:'joystix',fontSize: '30px', fill: '#ffffff'}).setOrigin(0.5);

this.input.on('pointerup',restartgame,this);

gameState.altcursors.enter.on('down',restartgame,this);

function restartgame() {

gameState.wintext.visible=false;

gameState.score = 1000;

gameRounds.gameRound=0;

Phaser.Actions.Call(gameState.enemies.getChildren(),function (enemy) {

gameState.enemies.remove(enemy,true,true);

});

//gameState.enemies=this.physics.add.group();

gameState.player.particleEmitter.active=true;

this.scene.restart();

gameState.gameover=false;

}

this.input.keyboard.on('keydown-ESC',backtomenu,this);

function backtomenu() {

gameState.wintext.visible=false;

gameState.test1.hidden=true;

gameState.score=1000;

gameRounds.gameRound=0;

Phaser.Actions.Call(gameState.enemies.getChildren(),function (enemy) {

gameState.enemies.remove(enemy,true,true);

});

gameState.gameover=true;

this.scene.restart();

}

}

## Az ajax lekérést feldolgozó php oldal

Ez az oldal dolgozza fel a játék álltal elküldött AJAX lekérést. Vagy beviszi az elküldött adatokat az adatbázisba, vagy pedig kiírja a 10 legmagasabb pontszámot.

<?php

require\_once "connection.php";

if (isset($\_REQUEST['name'])){

$\_REQUEST['name']=trim($\_REQUEST['name'],'\t\n\r');

$sql="INSERT INTO highscores (username,score,stage) VALUES ('".$\_REQUEST['name']."','".$\_REQUEST['score']."','".$\_REQUEST['stage']."')";

if($mysqli->query($sql)==false)

die('Procedure was unsuccessfull!:'.$mysqli->error);

}

else{

$text="Top 10 highscores \n";

$i=1;

$sql='SELECT \* FROM top10';

$result=$mysqli->query($sql);

if($result->num\_rows>0){

while($row=$result->fetch\_assoc()){

$text=$text.$i++.'.'.$row['username'].' '.$row['score']." \n";

}

}

else

{

$text='Empty';

}

echo ($text);

}

?>

## A connection.php

Tartalmazza az adatbázist összekötő fügvény létrehozását, és a szüséges adatokat a kapcsolat létrehozásához

<?php

define('USERNAME','root');

define('PASSWORD','');

define('HOST','localhost');

define('DATABASE','game');

$mysqli=new mysqli(HOST,USERNAME,PASSWORD,DATABASE);

if ($mysqli->connect\_error)

die('Problem with the database connection:'.$mysqli->connect\_error);

?>

## Admin oldal

Először is az admin oldal stilus beállításai és a szüksége chart.js könyvtár beágyazása.

<?php

require\_once "connection.php";

?>

<html>

<head>

<title>

Admin

</title>

<script src="https://www.jsdelivr.com/package/npm/chart.js?path=dist"></script>

<script src="https://cdnjs.com/libraries/Chart.bundle.js"></script>

<script src="node\_modules\chart.js\dist\Chart.bundle.js"></script>

<style>

table{

margin: auto;

display: block;

width: 90%;

border: 1px solid #eee;

max-height: 620px;

overflow: auto;

}

caption{

display: block;

line-height: 3em;

width: 100%;

-webkit-align-items: stretch;

border: 1px solid #eee

}

tbody{

display: -webkit-flex;

height: 400px;

overflow: auto;

-webkit-flex-flow: row wrap;

}

tbody tr{

display: -webkit-flex;

width: 100%;

}

thead{

display: -webkit-flex;

-webkit-flex-flow: row;

}

thead tr{

padding-right: 15px;

display: -webkit-flex;

width: 100%;

-webkit-align-items: stretch;

}

td,th{

text-align: left;

padding: 8px;

width: 15%;

}

tr:nth-child(even) {background-color: #f2f2f2;}

canvas{

margin: auto;

height: 600px;

width: 1200px;

}

button{

-webkit-align-items: stretch;

line-height: 3em;

float: right;

width: 200px;

height: inherit;

background-color:rgb(255, 99, 132);

color: whitesmoke;

margin: auto;

}

</style>

</head>

A az első dinamikusan létrehozott tábla és a canvas ami majd a gráfot tartalmazza, ez az adatbázis összes adatát, összes pontszámot tartalmazza.

<body>

<table id="all">

<caption>All time scores

<button onclick="getfull()">Get the full table as csv</button></caption>

<thead>

<tr>

<th>ID</th>

<th>Username</th>

<th>Score</th>

<th>Stage</th>

<th>Time</th>

</tr>

</thead>

<tbody>

<?php

$counter=1;

$score=array();

$stage=array();

$time=array();

$sql="select \* from highscores order by time asc ";

$results=$mysqli->query($sql);

while ($row = $results->fetch\_assoc())

{

array\_push($score,$row['score']);

array\_push($stage,$row['stage']);

array\_push($time,$row['time']);

echo "<tr>";

echo "<td>".$counter++."</td><td>".$row['username']."</td><td>".$row['score']."</td><td>".$row['stage']."</td><td>".$row['time']."</td>";

echo "</tr>";

}

?></tbody>

</table>

<canvas id="allchart"></canvas>

A második tábla és gráf ami az ebben a hónapban játszódott játékok pontszámait tartalmazza.

<table id="thismonth">

<caption>This months scores

<button onclick="getmonth()">Get this months table as a csv</button></caption>

<thead>

<tr>

<th>ID</th>

<th>Username</th>

<th>Score</th>

<th>Stage</th>

<th>Time</th>

</tr>

</thead>

<tbody>

<?php

$counter=1;

$mscore=array();

$mstage=array();

$mtime=array();

$sql="select \* from highscores where year(time)=year(current\_date) and month(time)=month(current\_date) order by time asc";

$results=$mysqli->query($sql);

while ($row = $results->fetch\_assoc())

{

array\_push($mscore,$row['score']);

array\_push($mstage,$row['stage']);

array\_push($mtime,$row['time']);

echo "<tr>";

echo "<td>".$counter++."</td><td>".$row['username']."</td><td>".$row['score']."</td><td>".$row['stage']."</td><td>".$row['time']."</td>";

echo "</tr>";

}

?>

</tbody>

</table>

<canvas id="thismonthchart"></canvas>

A két gráfot létrehozó script, és a gombok scriptje ami a táblák csv-ben való formában letölthetőségét teszi lehetővé.

<script>

var ctx = document.getElementById('allchart').getContext('2d');

var chart = new Chart(ctx, {

// The type of chart we want to create

type: 'line',

// The data for our dataset

data: {

labels:<?php echo json\_encode($time);?>,

datasets: [{

label: 'All time scores',

borderColor: 'rgb(255, 99, 132)',

lineTension:0.0,

data: <?php echo json\_encode($score);?>

}]

},

// Configuration options go here

options: {scales: {

xAxes: [{

type: 'time',

time: {

unit: 'month'

}

}]

},responsive:false}

});

var ctx2 = document.getElementById('thismonthchart').getContext('2d');

var chart2 = new Chart(ctx2, {

// The type of chart we want to create

type: 'line',

// The data for our dataset

data: {

labels:<?php echo json\_encode($mtime);?>,

datasets: [{

label: 'Monthly scores',

borderColor: 'rgb(255, 99, 132)',

lineTension:0.0,

data:<?php echo json\_encode($mscore);?>

}]

},

// Configuration options go here

options: {scales: {

xAxes: [{

type: 'time',

time: {

unit: 'day'

}

}]

},

responsive:false}

});

</script>

<br>

<script>

function getfull() {

window.location.replace('full\_csv.php');

}

function getmonth() {

window.location.replace('month\_csv.php');

}

</script>

</body>

</html>

## A csv letöltő oldalak

Az első kód, a teljes adatbázist letöltését teszi lehetővé csv formátumban.

<?php

require\_once "connection.php";

$sql="select \* from highscores order by time desc ";

$results=$mysqli->query($sql);

if (!$results)

die('Couldn\'t fetch records');

$num\_fields =mysqli\_num\_fields($results);

$headers = array();

for ($i = 0; $i < $num\_fields; $i++) {

$temp=mysqli\_fetch\_field\_direct($results,$i);

$headers[] = $temp->name;

}

$fp = fopen('php://output', 'w');

if ($fp && $results) {

header('Content-Type: text/csv');

header('Content-Disposition: attachment; filename="full\_list.csv"');

header('Pragma: no-cache');

header('Expires: 0');

fputcsv($fp, $headers);

while ($row = $results->fetch\_array(MYSQLI\_NUM)) {

fputcsv($fp, array\_values($row));

}

die;

}

?>

A második kódrészlet ugyan az mint az előző azzal, hogy cask az ebben a hónapban történt játékok adatjait konvertálja át csv-be.

<?php

require\_once "connection.php";

$sql="select \* from highscores where year(time)=year(current\_date) and month(time)=month(current\_date) order by time desc ";

$results=$mysqli->query($sql);

if (!$results)

die('Couldn\'t fetch records');

$num\_fields =mysqli\_num\_fields($results);

$headers = array();

for ($i = 0; $i < $num\_fields; $i++) {

$temp=mysqli\_fetch\_field\_direct($results,$i);

$headers[] = $temp->name;

}

$fp = fopen('php://output', 'w');

if ($fp && $results) {

header('Content-Type: text/csv');

header('Content-Disposition: attachment; filename="monthly\_list.csv"');

header('Pragma: no-cache');

header('Expires: 0');

fputcsv($fp, $headers);

while ($row = $results->fetch\_array(MYSQLI\_NUM)) {

fputcsv($fp, array\_values($row));

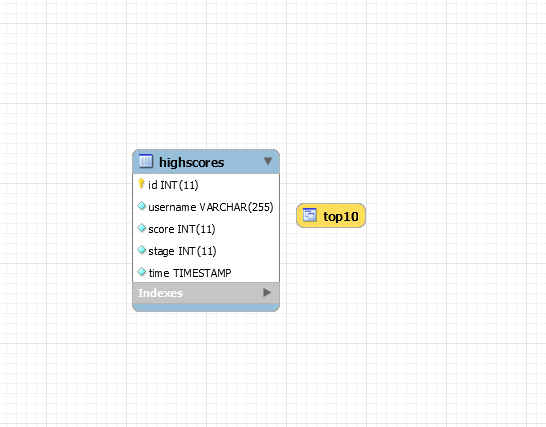
}

die;

}

?>

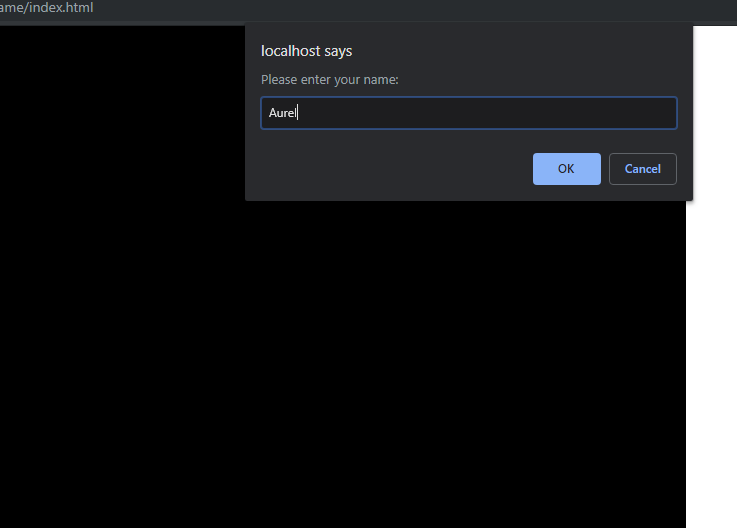
# AZ ADATBÁZIS STRUKTÚRÁJA



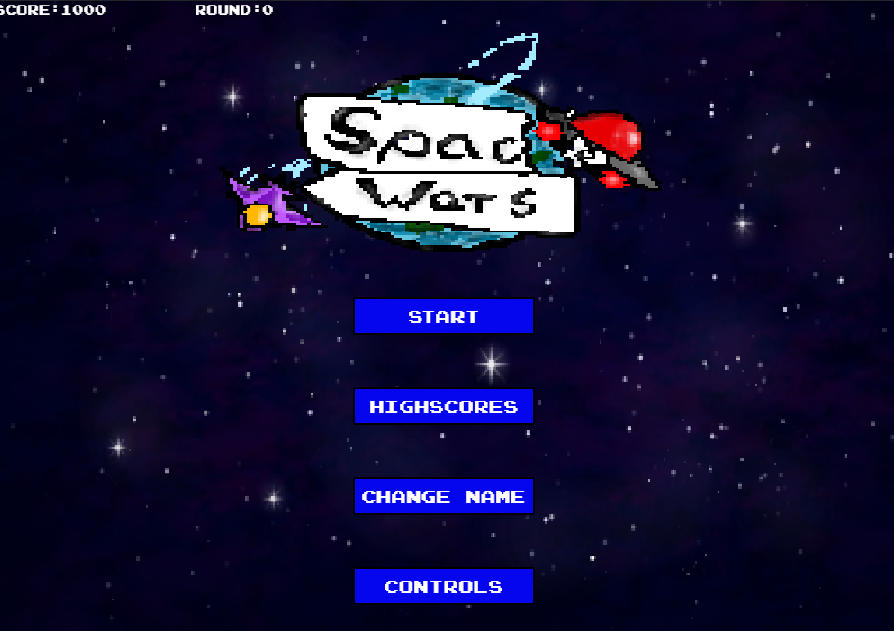
# A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA

## A játék oldal

Az oldal betöltésekor be kell vinnünk a nevünket amivel szeretnénk játszani.



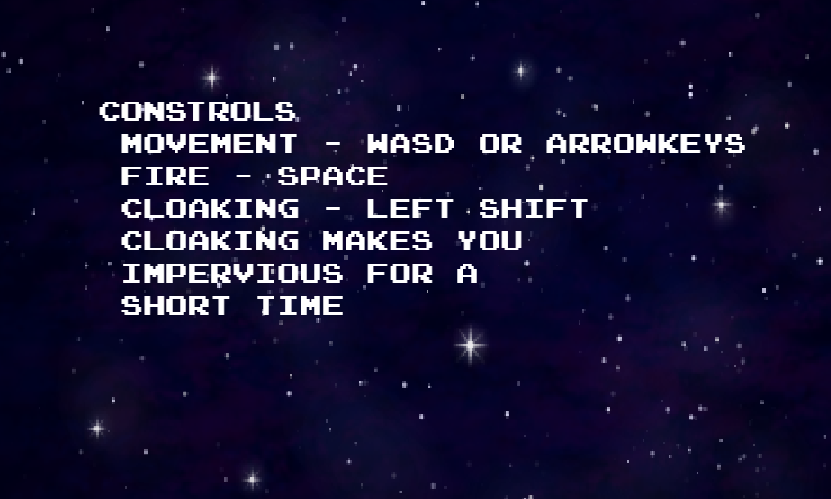
Ezután a képen látható főoldalon találjuk magunkat. Az oldalon 4 gomb van, az első elindítja a játékot. A második a highscore gomb, ami kiírja a top 10 pontszámot elérő játékost. A harmadik gombot megnyomva az előző képen látható képernyőt kapunk, és megváltoztathatjuk a nevünket. A negyedik gomb egy rövid leírást ad a játék kontroljaírol.



Így néz ki a highscores képernyő amit akkor kapunk ha a menüben a highscores gombot nyomjuk meg.



Ez a rövid control leírás amit akkor kapunk ha a menüben az utolsó gombot nyomjuk meg.

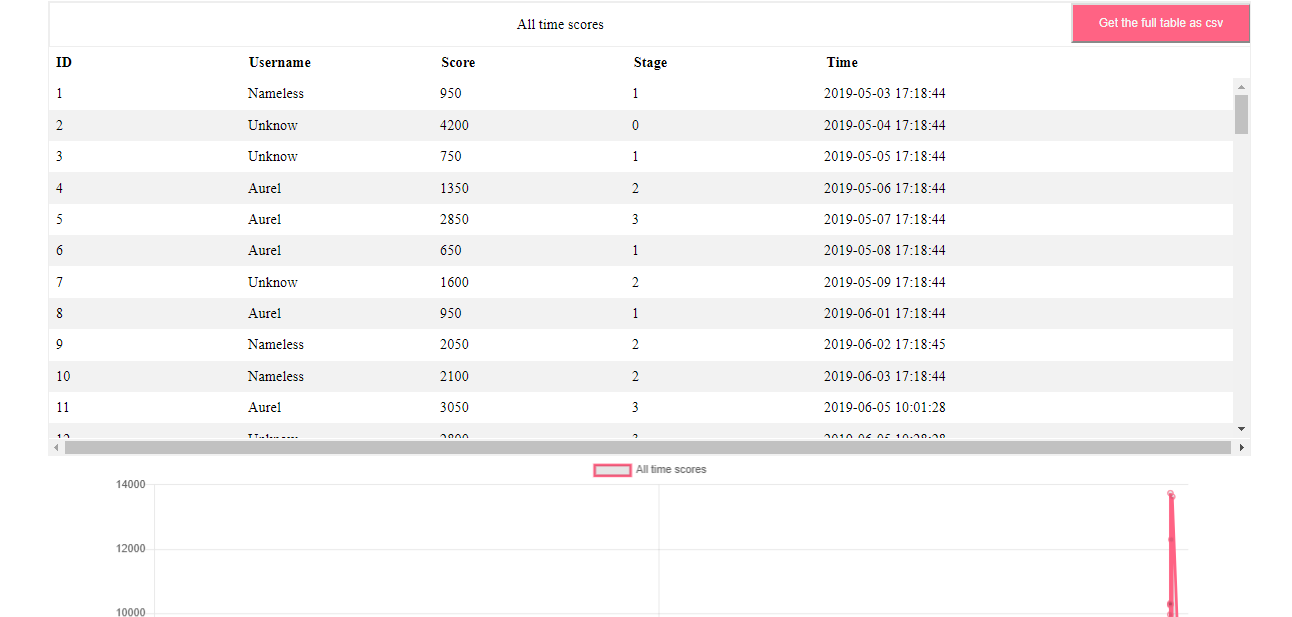


Ha elindítjuk a játékot, akkor a képernyőröl eltünik a menu és elindul a játék, ahol az alul láltható ürhajót irányítjuk. A bal felső sarokban látjuk a jelenlegi pontszámunkat, és a jelenlegi szintet. Halálunk esetén az alul látható felírat jelenik meg a nevünkel, a játék leál, és újra kezdhetjük a játékot vagy esc billentyüvel vissza léphetünk a menübe.

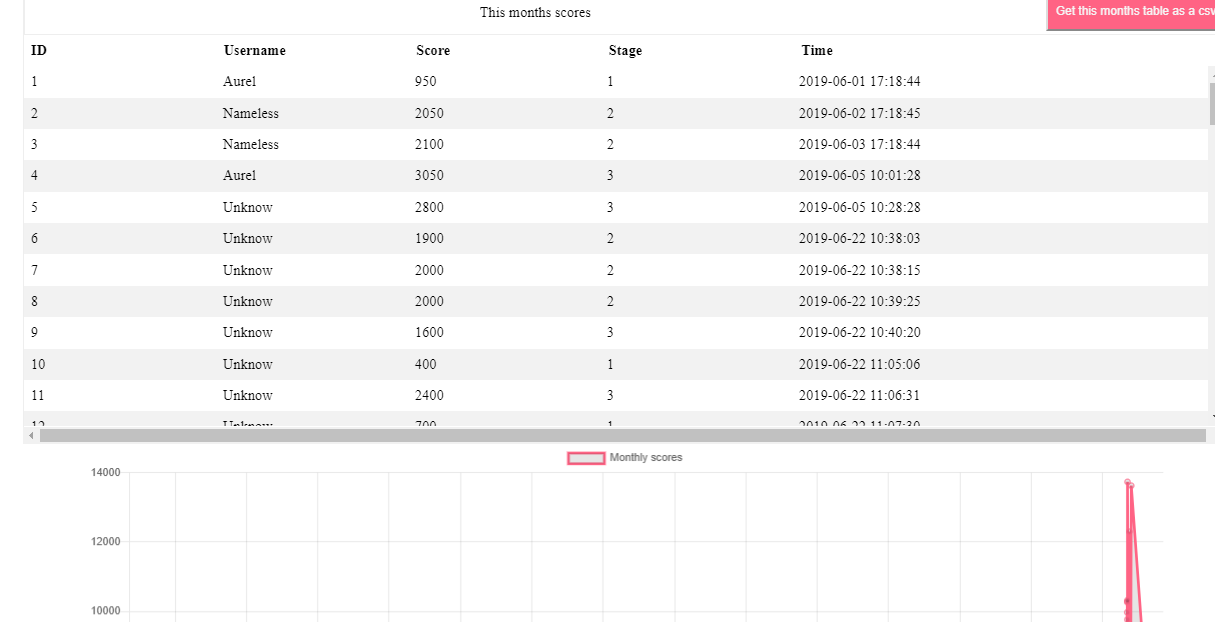


## Az admin oldal

Az admin oldalon láthatjuk az első táblában a teljes adatbázis adatait időszerinti sorrendben, a gráfon pedig az eddig elért pontszámok feltüntetve. A táblázat sarkában lévő gombal a táblzatot mint csv letölthetjük.



A második táblában és gráfon cask az ehónapban játszot játékok vannak ábrázolva. Ezt a táblázatot ugyanúgy letölthetjük csv formátumban.



# FELHASZNÁLT IRODALOM

W1. <https://photonstorm.github.io/phaser3-docs/>

W2. <https://www.chartjs.org/docs/latest/>

W3. <https://www.w3schools.com/>

W4. <https://stackoverflow.com/>