# Apêndice C

A seguir todo o código fonte da aplicação.

(function () {

var MG = {};

const P = 80,

SHIFT = 16,

Z = 90,

RAD = Math.PI / 180;

// capturando canvas e contexto

MG.canvas = document.getElementById('canvas');

MG.ctx = MG.canvas.getContext('2d');

//info sobre arquivos externos

MG.srcLoaded = 0;

MG.srcToLoad = 10;

function GameImage (file) {

var o = new Image();

o.src = file;

o.onload = function () {

MG.srcLoaded++;

MG.loaded = (109 \* MG.srcLoaded / MG.srcToLoad) + 1;

};

return o;

};

MG.bg = new GameImage('core/img/bg.png'); // fundo do jogo

MG.spritesheet = new GameImage('core/img/spritesheet.png'); // sprites do jogo

MG.startscreen = new GameImage('core/img/start.png'); // tela inicial

function GameAudio (file) {

var a = new Audio();

a.src = file;

a.status = setInterval(function () {

if (a.readyState === 4) {

MG.srcLoaded++;

MG.loaded = (109 \* MG.srcLoaded / MG.srcToLoad) + 1;

clearInterval(a.status);

if (MG.srcLoaded === MG.srcToLoad) {

setTimeout(function() {

MG.game.status = 'start';

MG.game.screen.change(2, 'arcade');

MG.sounds[6].started = true;

MG.sounds[6].loop = true;

playSound(6);

}, 1000);

}

}

}, 500);

return a;

}

MG.sounds = []; // audios do jogo

// carregando audios

(function () {

var i = 0,

audios = [

'sfx\_laser', 'sfx\_pause',

'sfx\_toggle', 'sfx\_lose',

'sfx\_meteor\_explosion',

'sfx\_ufo\_explosion',

'alcatraz'

];

for (; i < audios.length; i += 1) {

MG.sounds.push(new GameAudio('core/ogg/' + audios[i] + '.ogg'));

}

}());

// toca audio

function playSound (audio) {

MG.sounds[audio].currentTime = 0;

MG.sounds[audio].volume = 0.4;

MG.sounds[audio].play();

}

// dimensões do jogo

MG.height = MG.canvas.height = 550;

MG.width = MG.canvas.width = 350;

MG.game = {

bg: MG.bg,

ctx: MG.ctx,

status: 'loading',

y: 0, // posição do fundo

// telas do jogo

screen: {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

startscreen: MG.startscreen,

chosen: 'yes',

options: ['yes', 'no'],

tutorial\_page: 0,

height: MG.height,

width: MG.width,

// continuar

continue: function () {

var x = 0, y = 0,

font = '18px Bebas Neue',

half\_width = Math.floor(this.width / 2),

half\_height = Math.floor(this.height / 2);

this.ctx.textAlign = 'left';

this.ctx.fillStyle = 'white';

this.ctx.font = '30px Bebas Neue';

this.ctx.fillText('Do you wish to', half\_width - 72, half\_height);

this.ctx.fillText('Continue?', half\_width - 72, half\_height + 25);

//inverte opção selecionada

if (this.chosen === 'no') {

y = 8;

x = 18;

this.ctx.font = '18px Bebas Neue';

font = '30px Bebas Neue';

}

this.ctx.fillText('Yes', half\_width + 13 + x, half\_height + (80 - y));

this.ctx.font = font;

this.ctx.fillText('/No', half\_width + 50, half\_height + (72 + y));

},

// fim de jogo

gameover: function () {

var x = 0, y = 0, z = 0,

font = '18px Bebas Neue',

half\_width = Math.floor(this.width / 2),

half\_height = Math.floor(this.height / 2);

this.ctx.fillStyle = 'white';

this.ctx.font = '30px Bebas Neue';

this.ctx.textAlign = 'center';

this.ctx.fillText('Game Over', half\_width, half\_height + 15);

//inverte opção selecionada

if (this.chosen === 'menu') {

this.ctx.font = '18px Bebas Neue';

font = '31px Bebas Neue';

y = 8;

x = 14;

}

this.ctx.fillText('Retry', half\_width - 10, half\_height + 80 - y);

this.ctx.font = font;

this.ctx.fillText(

'/Menu', half\_width + 42 - z, half\_height + 72 + (z === 0 ? y : 0)

);

},

// carregando

loading: function () {

var half\_height = this.height / 2,

half\_width = this.width / 2;

this.ctx.textAlign = 'center';

this.ctx.fillStyle = 'white';

this.ctx.font = '30px Bebas Neue';

this.ctx.fillText(

'Loading', half\_width, half\_height - 9

);

//total

this.ctx.fillStyle = "#949394";

this.ctx.fillRect(

half\_width - 55, half\_height, 110, 5

);

//já carregado

this.ctx.drawImage(

this.spritesheet, 261, 1, MG.loaded, 20,

half\_width - 55, half\_height - 8, MG.loaded, 20

);

},

// pausa

paused: function () {

this.ctx.textAlign = 'center';

this.ctx.fillStyle = 'white';

this.ctx.font = '30px Bebas Neue';

this.ctx.fillText(

'Pause', Math.floor(this.width / 2),

Math.floor(this.height / 2) + 11

);

},

// tela inicial

start: function () {

var y = {

arcade: 363,

tutorial: 409

}[this.chosen];

this.ctx.drawImage(this.startscreen, 0, 0);

this.ctx.drawImage(

this.spritesheet, 335, 119, 29, 28,

160, y, 29, 28

);

},

// tutorial do jogo

tutorial: function () {

var half\_width = Math.floor(this.width / 2),

half\_height = Math.floor(this.height / 2),

y = 68;

this.ctx.textAlign = 'left';

this.ctx.fillStyle = 'white';

this.ctx.font = '30px Bebas Neue';

switch(this.tutorial\_page) {

case 0:

this.ctx.drawImage(

this.spritesheet, 42, 118, 34, 33,

half\_width - 17, half\_height - 91, 34, 33

);

this.ctx.fillText(

'To confirm in menu screens',

half\_width - 139, half\_height - 23

);

this.ctx.fillText(

'use "Enter".',

half\_width - 139, half\_height + 13

);

break;

case 1:

this.ctx.drawImage(

this.spritesheet, 82, 118, 154, 33,

half\_width - 84, half\_height - 91, 154, 33

);

this.ctx.fillText(

'Use the arrows to move the',

half\_width - 139, half\_height - 23

);

this.ctx.fillText(

'ship and to alternate ',

half\_width - 139, half\_height + 13

);

this.ctx.fillText(

'between the menu options. ',

half\_width - 139, half\_height + 49

);

break;

case 2:

this.ctx.drawImage(

this.spritesheet, 242, 118, 50, 33,

half\_width - 26, half\_height - 91, 50, 33

);

this.ctx.fillText(

'Holding he "shift" key',

half\_width - 139, half\_height - 23

);

this.ctx.fillText(

'pressed will move the ship',

half\_width - 139, half\_height + 13

);

this.ctx.fillText(

'slower. ', half\_width - 139,

half\_height + 49

);

break;

case 3:

this.ctx.drawImage(

this.spritesheet, 298, 118, 34, 33,

half\_width - 17, half\_height - 91, 34, 33

);

this.ctx.fillText(

'To shoot in your enemies',

half\_width - 139, half\_height - 23

);

this.ctx.fillText(

'press the key "Z".',

half\_width - 139, half\_height + 13

);

break;

case 4:

this.ctx.drawImage(

this.spritesheet, 3, 52, 34, 33,

half\_width - 129, half\_height - 152, 34, 33

);

this.ctx.drawImage(

this.spritesheet, 41, 52, 34, 33,

half\_width - 129, half\_height - 102, 34, 33

);

this.ctx.drawImage(

this.spritesheet, 111, 52, 34, 33,

half\_width - 129, half\_height - 52, 34, 33

);

this.ctx.drawImage(

this.spritesheet, 147, 52, 34, 33,

half\_width + 7, half\_height - 152, 34, 33

);

this.ctx.drawImage(

this.spritesheet, 76, 52, 34, 33,

half\_width + 7, half\_height - 102, 34, 33

);

this.ctx.fillText('Itens:', half\_width - 129, half\_height - 210);

this.ctx.font = '18px Bebas Neue';

this.ctx.fillText('One live', half\_width - 79, half\_height - 128);

this.ctx.fillText('50 points', half\_width - 79, half\_height - 79);

this.ctx.fillText('300 points', half\_width - 79, half\_height - 28);

this.ctx.fillText('Multiplier', half\_width + 54, half\_height - 128);

this.ctx.fillText('150 points', half\_width + 54, half\_height - 79)

break;

case 5:

this.ctx.drawImage(

this.spritesheet, 1, 88, 28, 28,

half\_width - 131, half\_height - 158, 28, 28

);

this.ctx.drawImage(

this.spritesheet, 60, 88, 28, 28,

half\_width - 131, half\_height - 126, 28, 28

);

this.ctx.drawImage(

this.spritesheet, 121, 87, 25, 25,

half\_width - 131, half\_height - 41, 25, 25

);

this.ctx.drawImage(

this.spritesheet, 146, 87, 25, 25,

half\_width + 1, half\_height - 41, 25, 25

);

this.ctx.drawImage(

this.spritesheet, 171, 87, 25, 25,

half\_width - 131, half\_height, 25, 25

);

this.ctx.drawImage(

this.spritesheet, 197, 87, 25, 25,

half\_width + 1, half\_height, 25, 25

);

this.ctx.font = '24px Bebas Neue';

this.ctx.fillText('Meteoros:', half\_width - 131, half\_height - 175);

this.ctx.fillText('Ufos:', half\_width - 131, half\_height - 57);

this.ctx.font = '18px Bebas Neue';

this.ctx.fillText('Until 50 points', half\_width - 85, half\_height - 139);

this.ctx.fillText('Until 50 points', half\_width - 85, half\_height - 106);

this.ctx.fillText('50 points', half\_width - 86, half\_height - 22);

this.ctx.fillText('320 points', half\_width - 86, half\_height + 19);

this.ctx.fillText('150 points', half\_width + 43, half\_height - 22);

this.ctx.fillText('470 points', half\_width + 43, half\_height + 19);

break;

case 6:

MG.player.ui();

MG.ship.draw();

this.ctx.font = '18px Bebas Neue';

this.ctx.fillText('1. In the left top corner',

half\_width - 79, half\_height - 128);

this.ctx.fillText('is your live counter, you',

half\_width - 79, half\_height - 107);

this.ctx.fillText('start with three lives.',

half\_width - 79, half\_height - 85);

this.ctx.fillText('2. At right top corner',

half\_width - 79, half\_height - 47);

this.ctx.fillText('is your score.',

half\_width - 79, half\_height - 22);

this.ctx.fillText('3. In the right bottom',

half\_width - 79, half\_height + 12);

this.ctx.fillText('corner, is your multiplier,',

half\_width - 79, half\_height + 34);

this.ctx.fillText('this counter multiply the',

half\_width - 79, half\_height + 56);

this.ctx.fillText('points that you get.',

half\_width - 79, half\_height + 78);

this.ctx.fillText('4. And that ship at the',

half\_width - 79, half\_height + 117);

this.ctx.fillText('bottom is your spaceship.',

half\_width - 79, half\_height + 139);

y = 175;

break;

default:

MG.game.screen.tutorial\_page = 0;

MG.game.status = 'start';

this.change(2, 'arcade');

}

this.ctx.drawImage(

this.spritesheet, 3, 118, 34, 33,

half\_width + 98, half\_height + y, 34, 33

);

},

// alterna entre as opções dos menus

change: function (o, chosen) {

this.options = [

['yes', 'no'], ['retry', 'menu'],

['arcade', 'tutorial']

][o];

console.log(chosen);

this.chosen = chosen;

}

},

// desenha fundo do jogo

draw: function(y) {

this.ctx.drawImage(this.bg, 0, y);

return this;

},

// movimenta o fundo

update: function() {

this.draw(this.y).draw(this.y - this.bg.height);

if (this.y > this.bg.height) {

this.y = 0;

} else {

this.y += 1;

}

return this;

},

// decisão do menu continue

continue: function () {

if (this.screen.chosen === 'yes') {

this.status = 'playing';

} else {

playSound(3);

this.status = 'gameover';

this.screen.change(1, 'retry');

}

},

// decisão do menu gameover

gameover: function () {

if (this.screen.chosen === 'retry') {

this.reset();

}

else if (this.screen.chosen === 'menu') {

this.reset();

this.screen.change(2, 'arcade');

this.status = 'start';

}

},

// decisão do menu inicial

start: function () {

switch(this.screen.chosen) {

case 'arcade':

this.status = 'playing';

MG.sounds[6].pause();

break;

case 'tutorial':

this.status = 'tutorial';

break;

}

},

// pausa o jogo

pause: function () {

if (this.status === 'playing') {

this.status = 'paused';

this.screen.paused();

}

else if (this.status === 'paused') {

this.status = 'playing';

}

},

// reiniciar

reset: function () {

MG.enemy.clear();

MG.player.reset();

MG.ship.reset();

MG.item.reset();

MG.explosion.reset();

this.screen.change(0, 'yes');

this.status = 'playing';

}

};

// nave do jogador

MG.ship = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height,

width: MG.width

},

// passo

step: 3,

// posição inicial

x: Math.floor(MG.width / 2) - 11,

y: MG.height - 38,

// dimensões

height: 15,

width: 22,

shoot: {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height

},

height: 13,

width: 5,

recover: 0, // intervalo entre os disparos

velocity: 6,

s: [],

// cria e adiciona um tiro ao vetor de tiros da nave

create: function () {

if (this.recover === 0) {

this.s.push({

height: this.height,

width: this.width,

// posição x da nave mais a metade de sua largura

// menos a metade da largura do tiro

x: this.uber.x + Math.floor(this.uber.width / 2 - this.width / 2),

// posição y da nave menos a altura do tiro e espaço extra

y: this.uber.y - this.height - 3,

});

// intervalo entre os disparos

this.recover = 10;

playSound(0);

}

},

// desenha os tiros da nave do jogador

draw: function() {

var i = 0, s,

max = this.s.length;

for (; i < max; i += 1) {

s = this.s[i];

this.ctx.drawImage(

this.spritesheet, 88, 26, 7, 21,

s.x, s.y, s.width, s.height

);

}

return this;

},

// atualiza a posição dos tiros da nave

update: function () {

var i = 0, s,

shoots = this.s,

max = shoots.length;

// impede que a nave atire antes do recover

// chegar em zero

if (this.recover > 0) {

this.recover--;

}

for (; i < max; i += 1) {

s = shoots[i];

// movimenta os tiros

s.y -= this.velocity;

// remove o tiro quando estiver fora da tela

if (s.y < -s.height) {

shoots.splice(i--, 1);

max--;

}

}

return this;

},

// remove todos disparos da nave

reset: function () {

this.s.splice(0, this.s.length);

}

},

// area colidivel

core: function () {

var half\_height = Math.floor(this.height / 2),

half\_width = Math.floor(this.width / 2);

return [

{

height: half\_height,

width: this.width - 2,

x: this.x + 1,

y: this.y + half\_height - 1,

},

{

height: half\_height,

width: half\_width - 5,

x: this.x + half\_width - 3,

y: this.y,

}

];

},

// reposiciona nave e remove seus disparos

reset: function () {

this.x = Math.floor(this.screen.width / 2) - 11,

this.y = this.screen.height - 38;

this.shoot.reset();

},

draw: function () {

this.ctx.drawImage(

this.spritesheet, 53, 26, 32, 21,

this.x, this.y, this.width, this.height

);

return this;

},

update: function () {

var i = 0;

for (; i < 4; i += 1) {

if (MG.joystick.pressed[i + 37]) {

this[['left', 'up', 'right', 'down'][i]]();

}

};

return this;

},

// movimenta a nave para baixo

down: function () {

this.y += this.step;

if (this.y + this.height > this.screen.height) {

this.y = this.screen.height - this.height;

}

},

// movimenta a nave para a esquerda

left: function() {

this.x -= this.step;

if (this.x < 0) {

this.x = 0;

}

},

// movimenta a nave para a direita

right: function() {

this.x += this.step;

if (this.x + this.width > this.screen.width) {

this.x = this.screen.width - this.width;

}

},

// movimenta a nave para cima

up: function() {

this.y -= this.step;

if (this.y < 0) {

this.y = 0;

}

}

};

// atribuindo herança(da nave) ao disparos da nave

MG.ship.shoot.uber = MG.ship;

// entradas do jogador

MG.joystick = {

// setas shift e tecla Z

listening: [SHIFT, 37, 38, 39, 40, Z],

pressed: [], // teclas pressionadas

keyDown: function (e) {

var k = e.which;

if (typeof this.pressed[k] === 'function') {

return this.pressed[k]();

}

else if (this.listening.indexOf(k) !== -1) {

this.pressed[k] = true;

}

},

keyUp: function (e) {

if (this.listening.indexOf(e.which) !== -1) {

this.pressed[e.which] = false;

}

},

keyDownMenu: function (e) {

var s = MG.game.screen,

index\_of = s.options.indexOf(s.chosen);

switch (e.which) {

case 13:

MG.game[MG.game.status].call(MG.game);

return;

case 37:

case 38:

if (index\_of > 0) {

index\_of -= 1;

playSound(2);

}

break;

case 39:

case 40:

if (index\_of < s.options.length - 1) {

index\_of += 1

playSound(2);

}

break;

}

s.chosen = s.options[index\_of];

}

};

// pausa

MG.joystick.pressed[P] = function () {

MG.game.pause();

playSound(1);

};

this.addEventListener('keydown', function (e) {

var e = e || this.event;

switch (MG.game.status) {

case 'paused':

case 'playing':

MG.joystick.keyDown(e);

break;

case 'continue':

case 'gameover':

case 'start':

MG.joystick.keyDownMenu(e);

break;

case 'tutorial':

if( e.which === 13) {

MG.game.screen.tutorial\_page += 1;

}

break;

}

}, false);

this.addEventListener('keyup', function (e) {

var e = e || this.event;

MG.joystick.keyUp(e);

}, false);

// meteoro

MG.meteor = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height,

width: MG.width

},

height: 15,

width: 15,

m: [],

drop: [4, 1, 1, 0], // quantidade de itens dropados

interval: 60,

wait: 0,

score: 25,

types: 4,

create: function () {

var type = randInt(this.types);

if (this.wait === 0) {

this.m.push({

height: this.height,

width: this.width,

// rotaciona os meteoros

rotate: 0,

type: type,

drop: this.drop,

// velocidade randomica

velocity: randInt(2) + 1 + type % 3,

score: this.score \* (type + 1),

// randomiza a posição horizontal inicial do meteor

x: randInt(this.screen.width - this.width),

y: -this.height

});

this.wait = this.interval;

}

},

draw: function () {

var i = 0, m,

meteors = this.m,

max = meteors.length,

half\_width, half\_height;

for (; i < max; i += 1) {

m = meteors[i];

half\_width = Math.floor(m.width / 2);

half\_height = Math.floor(m.height / 2);

// salva o contexto

this.ctx.save();

// transladar o contexto

this.ctx.translate(m.x + half\_width, m.y + half\_height);

// rotaciona o meteoro

this.ctx.rotate(m.rotate);

this.ctx.drawImage(

this.spritesheet, m.type \* 30, 87, 29, 29,

-half\_width, -half\_height, m.width, m.height

);

// restaura o contexto

this.ctx.restore();

}

return this;

},

update: function () {

var i = 0, m,

meteors = this.m;

if (this.wait > 0) {

this.wait -= 1;

} else {

this.create();

}

for (; i < meteors.length; i += 1) {

m = meteors[i];

// verifica colisão da nave com o meteoro

if (hasCollision(MG.ship.core(), m, false)) {

MG.shipCollided();

break;

}

// verifica colisão entre disparo da nave e o meteoro

if (hasCollision(MG.ship.shoot.s, m, true)) {

MG.enemyCollided(meteors, i, m.score);

playSound(4);

continue;

}

m.y += m.velocity;

m.rotate += RAD \* m.velocity;

if (m.y > this.screen.height) {

meteors.splice(i--, 1);

}

}

return this;

}

};

// ufo

MG.ufo = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height,

width: MG.width

},

height: 15,

width: 15,

velocity: 0.5,

interval: 150,

recover: 40,

wait: 0,

drop: [

[5, 3, 2, 1],

[6, 4, 3, 2],

[7, 5, 4, 3],

[3, 3, 5, 3]

],

score: [50, 150, 320, 470],

types: 4,

u: [],

shoot: {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height

},

height: 13,

width: 5,

s: [],

create: function (x, y, velocity, type) {

this.s.push({

height: this.height,

width: this.width,

// 0 - vertical

// 1 - teleguiados (movimentam-se

// horizontalmente na direção da nave)

// 2 - ufo segue o inimigo, disparos rapidos

// 3 - triplo disparo

type: type,

velocity: velocity / 4,

x: x - Math.floor(this.width / 2),

y: y

});

},

draw: function () {

var i = 0, s,

shoots = this.s,

max = shoots.length;

for (; i < max; i += 1) {

s = shoots[i];

this.ctx.drawImage(

this.spritesheet, 3 + (s.type \* 12),

27, 7, 19, s.x, s.y, s.width, s.height

);

}

return this;

},

update: function () {

var i = 0, s,

shoots = this.s,

guided, // guia do laser

shcx, scx; // centro no eixo x

for (; i < shoots.length; i += 1) {

s = shoots[i];

if (hasCollision(MG.ship.core(), s, false)) {

MG.shipCollided();

break;

}

if (s.y < this.screen.height) {

switch (s.type) {

case 0:

s.y += 1;

guided = 1;

break;

// teleguiado

case 1:

shcx = MG.ship.x + Math.floor(MG.ship.width / 2);

scx = s.x + Math.floor(s.width / 2);

guided = 1;

s.y += 1;

s.velocity = Math.abs(s.velocity);

if (s.y > MG.ship.y || shcx === scx) {

guided = 0;

break;

}

else if (shcx < scx) {

guided = -1;

}

break;

// ufo suicida

case 2:

s.y += 3;

guided = 0;

break;

// disparo triplo

case 3:

s.y += 2;

guided = 1;

break;

}

s.x += s.velocity \* guided;

} else {

shoots.splice(i--, 1);

}

}

return this;

}

},

create: function () {

var type = randInt(this.types),

i = (type === 2 ? type : randInt(2));

if (this.wait === 0) {

this.u.push({

height: this.height,

width: this.width,

level: type, // tipo do ufo

score: this.score[type],

drop: this.drop[type],

// direção

velocity: [this.velocity, -this.velocity, 1][i],

recover: 0, // intervalo entre os disparos

x: [

-this.width,

this.screen.width,

randInt(this.screen.width - this.width)

][i],

y: [

randInt(this.screen.height - 250),

-this.height,

][type !== 2 ? 0 : 1],

});

if (type === 3) {

this.u[this.u.length - 1].status = 'moving';

this.u[this.u.length - 1].wait = randInt(350) + 100;

}

this.wait = this.interval;

}

},

draw: function () {

var i = 0, u,

ufos = this.u,

max = ufos.length;

for (; i < max; i += 1) {

u = ufos[i];

this.ctx.drawImage(

this.spritesheet, 121 + (u.level \* 25), 87,

26, 26, u.x, u.y, u.width, u.height

);

}

return this;

},

update: function () {

var i = 0, j, vel,

u, ucx, scx,

max, direction,

ufos = this.u;

if (this.wait > 0) {

this.wait -= 1;

} else {

this.create();

}

for (; i < ufos.length; i += 1) {

u = ufos[i];

if (u.recover > 0) {

u.recover--;

} else {

max = 1;

if (u.level === 3) {

max = u.level;

}

vel = u.velocity;

for (j = 0; j < max; j += 1) {

if (u.level === 3) {

vel = [

Math.abs(this.velocity),

0,

Math.abs(this.velocity) \* -1,

][j];

}

this.shoot.create(

u.x + Math.floor(u.width / 2),

u.y + u.height, vel, u.level

);

}

u.recover = this.recover;

}

if (hasCollision(MG.ship.core(), u, false)) {

MG.shipCollided();

break;

}

if (hasCollision(MG.ship.shoot.s, u, true)) {

MG.enemyCollided(ufos, i, u.score);

playSound(5);

continue;

}

switch (u.level) {

case 0:

case 1: // laser teleguiado

direction = 1;

break;

// ufo seguidor

case 2:

ucx = u.x + Math.floor(u.width / 2);

scx = MG.ship.x + Math.floor(MG.ship.width / 2);

u.y += u.velocity;

direction = 0;

if (u.y > MG.ship.y) {

break;

}

else if (scx < ucx) {

direction = -1;

}

else if (scx > ucx) {

direction = 1;

}

break;

case 3:

switch (u.status) {

case 'moving':

direction = 1;

if (u.wait === 0) {

u.status = 'stoped';

u.wait = 200;

}

break;

case 'stoped':

direction = 0;

if (u.wait === 0) {

u.status = 'moving';

u.wait = randInt(350) + 100;

}

break;

}

u.wait -= 1;

}

u.x += u.velocity \* direction;

if (u.x < -this.width || u.x > this.screen.width ||

u.y > this.screen.height) {

ufos.splice(i--, 1);

}

}

return this;

}

};

// veifica colisão entre objetos

function hasCollision (array, target, should\_remove) {

var i = 0, a,

max = array.length;

for (; i < max; i += 1) {

a = array[i];

o = target;

if (a.width > o.width) {

a = target;

o = array[i];

}

if (a.y > o.y && a.y < o.y + o.height ||

a.y + a.height > o.y && a.y + a.height < o.y + o.height) {

if (a.x > o.x && a.x < o.x + o.width ||

a.x + a.width > o.x && a.x + a.width < o.x + o.width) {

if (should\_remove) {

array.splice(i, 1);

}

return true;

}

}

}

}

// inimigos

MG.enemy = {

e: [],

// remover inimigos

clear: function () {

var enemy;

for (enemy in this.e) {

this.e[enemy].splice(0, this.e[enemy].length);

}

}

};

MG.enemy.e.push(MG.meteor.m, MG.ufo.u, MG.ufo.shoot.s);

// nave colidiu

MG.shipCollided = function () {

this.player.credits -= 1;

this.player.mult = 1;

// apaga todos inimigos

this.enemy.clear();

if (this.player.credits > 0) {

this.game.status = 'continue';

this.game.screen.change(0, 'yes');

} else {

this.game.screen.change(1, 'retry');

this.game.status = 'gameover';

}

playSound(3);

};

// inimigo colidiu

MG.enemyCollided = function (array, i, score) {

var e = array[i], max;

e.y += Math.floor(e.height / 2);

e.x += Math.floor(e.width / 2);

// atribuindo pontuação

this.player.score += this.player.mult \* score;

// cria pontos

MG.item.drop(e, 0, 2);

// cria multiplicadores

max = randInt(e.drop[3]);

if (max > 0) {

MG.item.launch(e, 'mult', 3, 1, max + 1);

}

if (this.player.score > this.player.new\_life) {

if (this.player.credits < 6) {

MG.item.launch(e, 'live', -1, 1, 1);

}

this.player.new\_life \*= 2;

}

// explosão da colisão

this.explosion.create(e.x, e.y);

// removendo inimigo

array.splice(i, 1);

};

// explosões

MG.explosion = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

height: 7,

width: 8,

velocity: 1,

e: [],

create: function (x, y) {

this.e.push({

height: this.height,

width: this.width,

// estado inicial da animação

status: 'exploding',

wait: 40,

rotate: 0,

x: x,

y: y

});

},

draw: function () {

var i = 0, e,

max = this.e.length,

half\_width, half\_height;

for (; i < max; i += 1) {

e = this.e[i];

half\_width = Math.floor(e.width / 2);

half\_height = Math.floor(e.height / 2);

this.ctx.save();

this.ctx.translate(e.x , e.y);

this.ctx.rotate(e.rotate);

this.ctx.drawImage(

this.spritesheet, 99, 24, 25, 24,

-half\_width, -half\_height, e.width, e.height

);

this.ctx.restore();

}

return this;

},

update: function () {

var i = 0,

max = this.e.length,

//redimensionando explosão

transforming = function (e, increase) {

e.height += increase;

e.width += increase;

};

for (; i < max; i += 1) {

e = this.e[i];

e.rotate += RAD \* 3;

switch (e.status) {

//explosão crescendo

case 'exploding':

if (e.height > 18) {

e.status = 'idle';

} else {

transforming(e, this.velocity);

}

break;

//explosão no apice

case 'idle':

if (e.wait == 0) {

e.status = 'imploding';

} else {

e.wait -= 1;

}

break;

//explosão reduzindo

case 'imploding':

if (e.height < 2) {

this.e.splice(i--, 1);

max--;

} else {

transforming(e, -this.velocity);

}

break;

}

}

return this;

},

reset: function () {

this.e.splice(0, this.e.length);

}

};

// itens dropados

MG.item = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height,

},

height: 10,

width: 10,

i: [],

create: function (x, y, effect, type) {

this.i.push({

height: this.height,

width: this.width,

// indica o comportamento do item

effect: effect,

score: [50, 150, 300, 0][type],

status: 'dropping', //estado do item

type: type,

velocity: -1,

wait: 50,

x: x - Math.floor(this.width / 2),

y: y - Math.floor(this.height / 2)

});

},

draw: function () {

var i = 0, item,

itens = this.i,

max = itens.length;

for (; i < max; i += 1) {

item = itens[i];

this.ctx.drawImage(

this.spritesheet, 42 + (item.type \* 34), 52,

34, 33, item.x, item.y, item.width, item.height

);

}

return this;

},

update: function () {

var i = 0, item,

itens = this.i;

for (; i < itens.length; i += 1) {

item = itens[i];

// verifica se um item foi coletado

if (hasCollision(MG.ship.core(), item)) {

this[item.effect].call(item);

itens.splice(i--, 1);

continue;

}

switch (item.status) {

case 'dropping':

if (item.wait > 0) {

item.y += item.velocity;

item.wait -= 1;

}

else {

item.status = 'idle';

}

break;

case 'idle':

if (item.y < this.screen.height) {

item.y += item.velocity;

} else {

itens.splice(i--, 1);

}

break;

}

item.velocity += 0.01;

}

return this;

},

//lança um item na tela

launch: function (e, effect, type, i, max) {

var x = randInt(2) > 1 ? randInt(11) : (-randInt(7));

this.create(e.x + x, e.y, effect, type);

if (i < max) {

setTimeout(function () {

MG.item.launch(e, effect, type, i + 1, max);

}, 200);

}

},

// dropa os varios tipos de item

drop: function (e, type, limit) {

var max = randInt(e.drop[type]);

this.launch(e, 'score', type, 0, max);

if (type < limit) {

setTimeout(function () {

MG.item.drop(e, type + 1, limit);

}, 200);

}

},

// pontuação

score: function () {

MG.player.score += this.score \* MG.player.mult;

},

// multiplicador

mult: function () {

MG.player.mult += 1;

},

// incrementa creditos

live: function () {

MG.player.credits += 1;

},

// apaga todos os itens

reset: function () {

this.i.splice(0, this.i.length);

}

};

// dados do jogador

MG.player = {

ctx: MG.ctx,

spritesheet: MG.spritesheet,

screen: {

height: MG.height,

width: MG.width

},

credits: 3, // vidas

mult: 1, // multiplicador

score: 0, // pontuação

new\_life: 1000000, // pontuação minima para liberar um credito

// restaura pontuação, multiplicador e vidas

reset: function () {

this.credits = 3;

this.mult = 1;

this.score = 0;

},

numbers: function (value, y) {

var v = value.toString(),

i = 0, max = v.length;

for (; i < max; i += 1) {

this.ctx.drawImage(

this.spritesheet, (v[i] \* 22), 1, 20, 20,

this.screen.width - ((max - 1 - i) \* 13 + 20),

y, 13, 13

);

}

},

// informações do jogador:

ui: function() {

var i = 0,

m = this.mult;

// vidas restantes

for (; i < this.credits; i += 1) {

this.ctx.drawImage(

this.spritesheet, 241, 0, 13, 21,

(i \* 12 + 8), 10, 9, 14

);

}

// pontuação

this.numbers(this.score, 10);

// multiplicador

this.numbers(m, this.screen.height - 22);

this.ctx.drawImage(

this.spritesheet, 221, 1, 17, 17,

this.screen.width - (m.toString().length \* 13 + 17),

(this.screen.height - 20.5), 10, 10

);

}

};

// inteiro randomico

function randInt (max) {

return Math.floor(Math.random() \* max);

}

MG.main = function () {

var status = MG.game.status;

requestAnimationFrame(MG.main);

if (status === 'paused') {

return;

}

MG.game.draw(0).update();

switch (status) {

case 'playing':

if (MG.joystick.pressed[Z]) {

MG.ship.shoot.create();

}

if (MG.joystick.pressed[SHIFT] && MG.ship.step !== 1) {

MG.ship.step = 1;

}

else if (!(MG.joystick.pressed[SHIFT]) && MG.ship.step === 1) {

MG.ship.step = 3;

}

MG.ship.update().draw();

MG.ship.shoot.update().draw();

MG.meteor.update().draw();

MG.ufo.update().draw();

MG.explosion.update().draw();

MG.item.update().draw();

MG.player.ui();

break;

case 'loading':

case 'gameover':

case 'continue':

case 'start':

case 'tutorial':

console.log(status);

MG.game.screen[status]();

}

};

MG.main();

}());