Para establecer el control del acelerómetro mediante Cordova, cuando el dispositivo esté listo. Aunque Pharse controla que esto ocurre.

|  |
| --- |
| if ('addEventListener' in document) { |
|  | document.addEventListener('deviceready', function() { |
|  | app.inicio(); |
|  | }, false); |
|  | } |

Si bien, según la documentación de Phaser, éste se encarga de comprobar que el dispositivo esté listo. Así que de momento no incluimos este evento, en su lugar, creamos la función dentro del objeto GameState y lo invocamos desde el estado Preload. Es necesario también establecer las variables velecidadX, velocidadY.

Las primeras instrucciones del objeto GamaState:

velocidadX = 0;

velocidadY = 0;

En Preload:

app.vigilaSensores();

Y al final del objeto GameState:

vigilaSensores: function(){

function onError() {

console.log('onError!');

}

function onSuccess(datosAceleracion){

app.registraDireccion(datosAceleracion);

}

navigator.accelerometer.watchAcceleration(onSuccess, onError,{ frequency: 10 });

},

registraDireccion: function(datosAceleracion){

velocidadX = datosAceleracion.x ;

velocidadY = datosAceleracion.y ;

}

En Create, para asignar el control el motor phisica del sprite pel y el choque con las bandas:

//para la aceleración

this.game.physics.arcade.enable(pet);

this.pet.body.collideWorldBounds = true;

this.pet.body.onWorldBounds = new Phaser.Signal();

En Update:

// cuando hay aceleración

this.pet.body.velocity.y = velocidadY;

this.pet.body.velocity.x = velocidadX \* -1;

No funciona:

El js hasta aquí es:

//this game will have only 1 state

var GameState = {

// variables para acelerómetro

inicio: function(){

dificultad = 0;

velocidadX = 0;

velocidadY = 0;

puntuacion = 0;

this.vigilaSensores();

var game = new Phaser.Game(360, 640, Phaser.AUTO);

game.state.add('GameState', GameState);

game.state.start('GameState');

},

//load the game assets before the game starts

preload: function() {

this.game.load.image('backyard', 'assets/images/fondo.png');

this.game.load.image('apple', 'assets/images/manzana.png');

this.game.load.image('candy', 'assets/images/caramelo.png');

this.game.load.image('rotate', 'assets/images/rotate.png');

this.game.load.image('toy', 'assets/images/pato.png');

this.game.load.image('arrow', 'assets/images/arrow.png');

this.load.spritesheet('pet', 'assets/images/pet.png', 97, 83, 5, 1, 1);

//para añadir control de acelerómetro

},

//executed after everything is loaded

create: function() {

//scaling options

this.scale.scaleMode = Phaser.ScaleManager.SHOW\_ALL;

//have the game centered horizontally

this.scale.pageAlignHorizontally = true;

this.scale.pageAlignVertically = true;

//screen size will be set automatically

this.scale.setScreenSize(true);

this.background = this.game.add.sprite(0,0, 'backyard');

this.background.inputEnabled = true;

this.background.events.onInputDown.add(this.placeItem, this);

this.pet = this.game.add.sprite(100, 400, 'pet',0);

this.pet.animations.add('funnyfaces', [0, 1, 2, 3, 2, 1, 0], 7, false);

this.pet.anchor.setTo(0.5);

//custom properties of the pet

this.pet.customParams = {health: 100, fun: 100};

//draggable pet

this.pet.inputEnabled = true;

this.pet.input.enableDrag();

//para la aceleración

this.game.physics.arcade.enable(pet);

this.pet.body.collideWorldBounds = true;

this.pet.body.onWorldBounds = new Phaser.Signal();

//buttons

this.apple = this.game.add.sprite(72, 570, 'apple');

this.apple.anchor.setTo(0.5);

this.apple.customParams = {health: 20};

this.apple.inputEnabled = true;

this.apple.events.onInputDown.add(this.pickItem, this);

this.candy = this.game.add.sprite(144, 570, 'candy');

this.candy.anchor.setTo(0.5);

this.candy.customParams = {health: -10, fun: 10};

this.candy.inputEnabled = true;

this.candy.events.onInputDown.add(this.pickItem, this);

this.toy = this.game.add.sprite(216, 570, 'toy');

this.toy.anchor.setTo(0.5);

this.toy.customParams = {fun: 30};

this.toy.inputEnabled = true;

this.toy.events.onInputDown.add(this.pickItem, this);

this.rotate = this.game.add.sprite(288, 570, 'rotate');

this.rotate.anchor.setTo(0.5);

this.rotate.inputEnabled = true;

this.rotate.events.onInputDown.add(this.rotatePet, this);

this.buttons = [this.apple, this.candy, this.toy, this.rotate];

//nothing selected

this.selectedItem = null;

//stats

var style = { font: "20px Arial", fill: "#fff"};

this.game.add.text(10, 20, "Comida:", style);

this.game.add.text(140, 20, "Risa:", style);

this.healthText = this.game.add.text(100, 20, "", style);

this.funText = this.game.add.text(200, 20, "", style);

this.refreshStats();

//decrease health and fun every 10 seconds

this.statsDecreaser = this.game.time.events.loop(Phaser.Timer.SECOND \* 5, this.reduceProperties, this);

this.statsDecreaser.timer.start();

this.uiBlocked = false;

},

//rotate the pet

rotatePet: function(sprite, event) {

if(!this.uiBlocked) {

this.uiBlocked = true;

//alpha to indicate selection

this.clearSelection();

sprite.alpha = 0.4;

//vibrate device if present

if(navigator.vibrate) {

navigator.vibrate(1000);

}

var petRotation = game.add.tween(this.pet);

petRotation.to({ angle: '+720' }, 1000);

petRotation.onComplete.add(function(){

this.uiBlocked = false;

sprite.alpha = 1;

this.pet.customParams.fun += 10;

//show updated stats

this.refreshStats();

}, this);

petRotation.start();

}

},

//pick an item so that you can place it on the background

pickItem: function(sprite, event) {

if(!this.uiBlocked) {

//clear other buttons

this.clearSelection();

//alpha to indicate selection

sprite.alpha = 0.4;

//save selection so we can place an item

this.selectedItem = sprite;

}

},

//place selected item on the background

placeItem: function(sprite, event) {

if(this.selectedItem && !this.uiBlocked) {

//position of the user input

var x = event.position.x;

var y = event.position.y;

//create element in this place

var newItem = this.game.add.sprite(x, y, this.selectedItem.key);

newItem.anchor.setTo(0.5);

newItem.customParams = this.selectedItem.customParams;

//the pet will move to grab the item

this.uiBlocked = true;

var petMovement = game.add.tween(this.pet);

petMovement.to({x: x, y: y}, 700);

petMovement.onComplete.add(function(){

this.uiBlocked = false;

//destroy item

newItem.destroy();

//animate pet

this.pet.animations.play('funnyfaces');

//update pet stats

var stat;

for(stat in newItem.customParams) {

//make sure the property belongs to the object and not the prototype

if(newItem.customParams.hasOwnProperty(stat)) {

this.pet.customParams[stat] += newItem.customParams[stat];

}

}

//show updated stats

this.refreshStats();

//clear selection

this.clearSelection();

}, this);

petMovement.start();

}

},

//clear all buttons from selection

clearSelection: function() {

//set alpha to 1

this.buttons.forEach(function(element){element.alpha = 1});

//clear selection

this.selectedItem = null;

},

//show updated stats values

refreshStats: function() {

this.healthText.text = this.pet.customParams.health;

this.funText.text = this.pet.customParams.fun;

},

//the pet slowly becomes less health and bored

reduceProperties: function() {

this.pet.customParams.health = Math.max(0, this.pet.customParams.health - 20);

this.pet.customParams.fun = Math.max(0, this.pet.customParams.fun - 30);

this.refreshStats();

},

//game loop, executed many times per second

update: function() {

if(this.pet.customParams.health <= 0 || this.pet.customParams.fun <= 0) {

this.pet.customParams.health = 0;

this.pet.customParams.fun = 0;

this.pet.frame = 4;

this.uiBlocked = true;

this.game.time.events.add(2000, this.gameOver, this);

}

// cuando hay aceleración

this.pet.body.velocity.y = velocidadY;

this.pet.body.velocity.x = velocidadX \* -1;

},

gameOver: function() {

this.game.state.restart();

},

vigilaSensores: function(){

function onError() {

console.log('onError!');

}

function onSuccess(datosAceleracion){

app.registraDireccion(datosAceleracion);

}

navigator.accelerometer.watchAcceleration(onSuccess, onError,{ frequency: 10 });

},

registraDireccion: function(datosAceleracion){

velocidadX = datosAceleracion.x ;

velocidadY = datosAceleracion.y ;

}

};

if ('addEventListener' in document) {

document.addEventListener('deviceready', function() {

GameState.inicio();

}, false);

}

//initiate the Phaser framework

CREO OTRA RAMA: debugAcelerometro

Y voy probando:

* Quito todo pero dejo la llamada inicial.

if ('addEventListener' in document) {

document.addEventListener('deviceready', function() {

GameState.inicio();

}, false);

}

inicio: function(){

//dificultad = 0;

//velocidadX = 0;

//velocidadY = 0;

//puntuacion = 0;

//this.vigilaSensores();

var game = new Phaser.Game(360, 640, Phaser.AUTO);

game.state.add('GameState', GameState);

game.state.start('GameState');

},

He asteriscado la llamada a vigilaSensores y las variables.

Y en Create:

//para la aceleración

// this.game.physics.arcade.enable(pet);

//this.pet.body.collideWorldBounds = true;

//this.pet.body.onWorldBounds = new Phaser.Signal();

En update:

// cuando hay aceleración

// this.pet.body.velocity.y = velocidadY;

// this.pet.body.velocity.x = velocidadX \* -1;

La aplicación carga, pero el juego no funciona. ¿Raro, no?

Otra prueba:

if ('addEventListener' in document) {

document.addEventListener('deviceready', function() {

GameState.inicio();

}, false);

var game = new Phaser.Game(360, 640, Phaser.AUTO);

game.state.add('GameState', GameState);

game.state.start('GameState');

¡Vale, hasta aquí funciona, pero sin acelerómetro!

Mierda, acabo de ver que en VigilaSensores se invocaba a una función de app, en lugar de GameState (el copy-paste)

De momento llamo solo a VigilaSensores

¡Vale, hasta aquí funciona, con acelerómetro! Pero sin que se aplique a la aplicación.

Quito el resto de asteriscos.

Y deja de funcionar, así que poco a poco.

La apk carga pero no con el acelerómetro.

Vale, carga, pero no se muevo nada.

Pruebo ahora a variar el valor de la puntuación con el acelerómetro y sí funciona, lo único que falta es hacer que se mueva el pet.

Funciona con la manzana, pero es que además he cambiado en la velocidad que la multiplico por 300

Quito que sea draggable , porque no vá.