Como fazer um jogo de Pong com realidade aumentada usando tracking.js

Nesse tutorial iremos aprender a criar um jogo simples de pong usando uma biblioteca de realidade aumentada tracking.js, portanto, baixe a mesma no link: “http://trackingjs.com/” para que possamos criar o game

Primeiro iremos criar o documento HTML

<!doctype html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>Pong Gesture</title>

<!--SUBSTITUA AQUI O LOCAL ONDE VOCÊ COLOCOU AS BIBLITECAS DO TRACKING.JS-->

<script src="../js/build/tracking-min.js"></script>  
 <script src="../../dat-gui/build/dat.gui.min.js"></script>  
 <script src="../js/assets/stats.min.js"></script>  
 <script src="../js/assets/color\_camera\_gui.js"></script>  
 <!-- Basic styling, centering the canvas -->

<style>  
 canvas {  
 display: block;  
 position: absolute;  
 margin: auto;  
 top: 0;  
 bottom: 0;  
 left: 0;  
 right: 0;  
 }

</style>  
</head>

<body>  
 <div>  
 <div>  
 <video id="video" width="300" height="225" preload autoplay loop muted controls></video>  
 <canvas id="canvas" width="300" height="225"></canvas>  
 </div>  
 </div>

<script>

Mas, vamos agora aprender um pouco de javascript.

Primeiro, vamos definir nosso campo onde irá acontecer as disputas de pong. Para isso, vamos definir os limites da nossa tela e guarda-los em duas variáveis, vamos chama-las respectivamente de Largura e Altura. Para capturar a largura vamos usar o comando window.innerWidth e guardá-lo na variável Largura e o para capturar a altura vamos usaro comando window.innerHeight e guardá-lo na variável ALTURA. Ficando com o código dessa maneira:

var  
 LARGURA = window.innerWidth,  
 ALTURA = window.innerHeight,  
 canvas,  
 ctx

Agora, vamos definir a função principal do nosso projeto que será a nossa main, que será responsável por fazer a chamada de todas as outras funções.

function main() {

}

Agora com nossas variáveis com o tamanho da tela definidas vamos usar o comando createElement("canvas ") para criar imagens por código. E criaremos uma variável ctx (abreviação de contexto) usando o comando canvas.getContext("2D " ) para informar que esse elemento irá ser 2D (sim, o canvas pode criar elementos 3D!) e para não precisarmos ficar digitando toda hora canvas.getContext("2D").alguma\_coisa. Essa variável será responsável por desenhar todos os elementos do jogo, como as paletas, campo e a bola. Por fim iremos adicionar essa variável ao corpo do nosso documento HTML com o comando document.body.appendChild(canvas). Então nosso código ficará dessa forma:

function main(){

canvas = document.createElement("canvas")  
canvas.width = LARGURA  
canvas.height = ALTURA  
ctx = canvas.getContext("2d");  
document.body.appendChild(canvas);

}

Vamos criar uma função responsável por fazer todos os desenhos, vamos chama-la de “desenhar”.

function desenhar() {  
  
 //Desenhando o retângulo na tela a partir da posição x = 0, y = 0 do tamanho dos limites da minha tela (que eu havia capturado anteriormente na variável)  
 ctx.fillRect(0, 0, LARGURA, ALTURA);

//entender melhor como funciona essa bagaça...  
 ctx.save();  
 //definindo a cor dos meus elementos em hexa decimal, sendo fff = branco. E cada caracter é respectivamente responsável pela cor Vermelho, Verde e Azul.  
 ctx.fillStyle = "#fff";  
 }

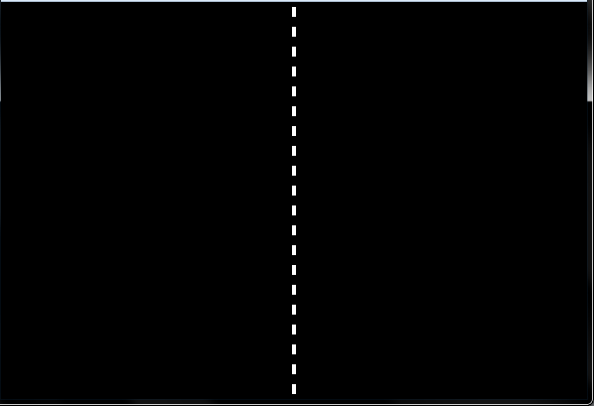
Agora vamos fazer a linha que delimita os lados dos jogadores. Então, dentro da função “desenhar” coloque:

//Definindo uma variável para dizer qual o tamanho da largura da linha  
 var larguraLinha = 4;  
 //Definindo outra variável para dizer qual será a posição X da linha, que no caso, como eu quero colocar no meio do campo, seria a “largura do campo” menos a “largura da linha” divido por 2  
 var x = (LARGURA - larguraLinha) / 2;  
 //Definindo outra variável para dizer a posição Y a partir de onde será desenhado o tracejado  
 var y = 0;  
 //aqui estamos definindo outra variável para dizer qual será a distância entre um traço e outro  
 var passo = ALTURA/20;

//"x" define a posição X do retângulo, y diz a posição Y do retângulo, estamos somando com passo / 4 pois queremos que ele fique no começo da tela com um espaço de 1/4 sem estar pintado. A "larguraLinha" diz a largura do retângulo, e o passo / 2 é que queremos que o tamanho vertical dele sejá igual a metade (1/2) do passo.

ctx.fillRect(x, larguraLinha + passo / 4, larguraLinha, passo / 2);  
 y += passo;  
 }  
 //entender melhor como funciona essa bagaça...  
 ctx.restore();  
}

Se tudo até agora deu certo o resultado esperado deve ser isso:



Agora com nosso campo feito, vamos partir para a criação das paletas, que serão o que cada jogador irá controlar.

Então, vamos criar um objeto, em javascript chamado Jogador. Onde o jogador irá conter, uma posição X e Y inicial, uma pontuação, e irá precisar de uma cor para que possa se mover. Vamos criar também mais duas variáveis para que caso tamanho do monitor seja diferente as paletas (jogadores) consigam se adaptar ao tamanho, e não fique desproporcional, e uma última para queposamos definir a posição Y inicial dos jogadores. Então, mãos na massa!

var  
 LARGURA = window.innerWidth,  
 ALTURA = window.innerHeight,  
 LARGURA\_JOGADOR = LARGURA / 100,  
 ALTURA\_JOGADOR = ALTURA / 6,  
 POS\_INI\_Y = (ALTURA - ALTURA\_JOGADOR) / 2,  
 canvas,  
 ctx,

//Criação do objeto player. Onde respectivamente os parâmetros são referentes a posição X, Y, posição X de onde ficará o valores da pontuação, e a cor para que ele possa se mover na tela.  
 Jogador = function(X, Y, pX, corParaMover){

return {  
 x: X,  
 y: Y,  
 pontos: 0,  
 cor: corParaMover,  
 largura: 20,  
 altura: 100,

//método responsável para atualizar a posição Y da paleta conforme o jogador move a cor.  
 atualizar: function() {  
 //define limites para que o meu jogador não saia dos limites da tela  
 this.y = Math.max(Math.min(this.y, HEIGHT - this.altura), 0);  
 },

//método responsável por atualizar o desenho da paleta conforme o jogador move a cor.  
 desenhar: function() {  
 //desenha a paleta com os parâmetros passados  
 ctx.fillRect(this.x, this.y, this.largura, this.altura);  
 //define a fonte que será usada para escrever a pontuação do jogador  
 ctx.font = "26px Helvetica";  
 //define o que será escrito na tela, e qual será a posição X e depois a Y  
 ctx.fillText(this.pontos, pX, 20);  
 }   
 };  
 };

Com nossa classe criada, coloque logo após ela a criação dos objetos, já que iremos usá-los logo:

jogador1 = new Jogador(0, POS\_INI\_Y, LARGURA / 2 - 35, 'magenta');  
jogador2 = new Jogador(LARGURA - LARGURA\_JOGADOR, POS\_INI\_Y, LARGURA / 2 + 20, 'cyan');

Precisamos colocar também a chamada para desenhar os jogadores no campo na nossa função “desenhar”, então para isso adicione em um local antes do “restore()”:

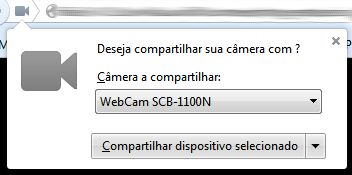
jogador1.desenhar();  
jogador2.desenhar();

Agora, vamos fazer os nossos jogadores se moverem, para isso, vamos começar a usar a biblioteca jasvascript “trackin.js” e colocaremos logo após nosso jogador player a seguinte chamada:

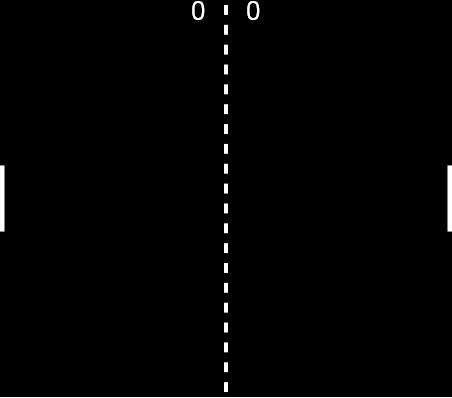
//pegando o elemento pelo "id"  
 var video = document.getElementById('video');

//definindo a variavel para trabalhar com a realidade aumentada, e definindo as cores que a câmera irá procurar  
 var tracker = new tracking.ColorTracker([player.cor, player2.cor]);   
 //pegando a tag "video" que possui o ID = video, e liga a câmera  
 tracking.track('#video', tracker, {  
 camera: true  
 });

//pegando a var. "track" e falando ligado a ela, o programa fará...  
 tracker.on('track', function(event) {  
 //fazendo um for para cada retângulo referente as cores que a biblioteca encontrar  
 event.data.forEach(function(rect) {  
 //se a cor for igual a cor do jogador 1 ele irá posicionar a paleta (jogador) conforme o jogador(humano) mover a cor.  
 if (rect.color === jogador1.cor)  
 jogador1.y = rect.y \* 2.7;  
 if (rect.color === jogador2.cor){  
 jogador2.y = rect.y \* 2.7;  
 }  
 });  
 });

Note que ele irá pedir autorização para capturar a sua câmera::

E se tudo deu certo, o seu jogo deve estar dessa maneira:

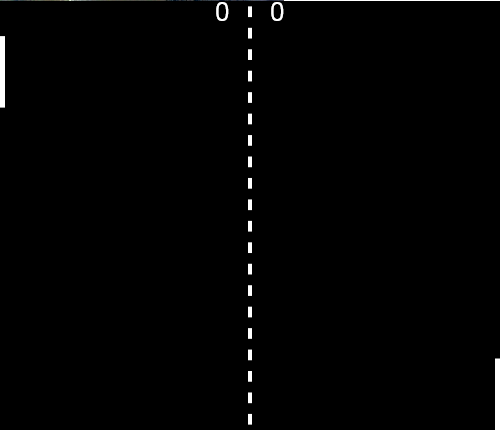


Porém, note que mesmo que você faça os movimentos com a cor nenhuma paleta se move. Isso se deve pois, a tela precisa ser “repintada, para isso, precisamos implementar mais um trecho de código na nossa função principal (main):

var repintar = function() {  
 atualizar();  
 desenhar();  
 window.requestAnimationFrame(repintar, canvas);  
 };

window.requestAnimationFrame(repintar, canvas);

Agora sim, as paletas devem estar se movendo conforme o esperado:



Mas, ainda falta algo essencial, a bola! Vamos então, criar uma bola que possua o atributo diâmetro. Vamos por enquanto apenas criar uma variável informando o diâmetro da bola, e também um método “desenhar” para que possamos enxergar se nossa bola está sendo desenhada na tela ou não, para isso digite logo após a criação dos jogadores:

//esse if ternário, irá servir para que se a tela do usuário for muito pequena, a bola não fique tão pequena também, facilitando a jogabilidade e a diversão.  
 DIAMETRO\_BOLA = window.innerWidth / (window.innerWidth < 600 ? 30 : 70),

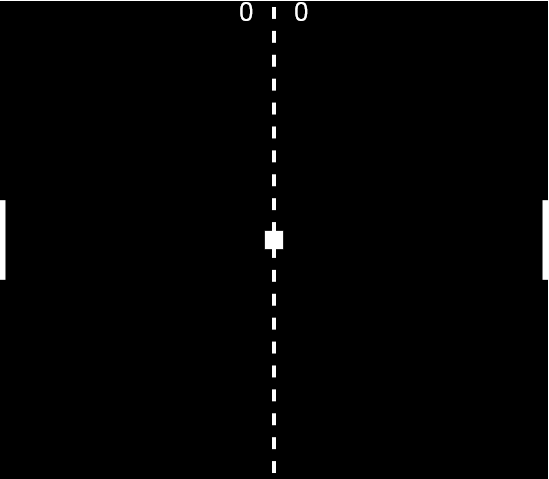
Bola = function(Diametro){  
 return {  
 x: null,  
 y: null,  
 diametro: Diametro,  
 velocidade: null,  
 desenhar: function() {  
 ctx.fillRect(this.x, this.y, this.diametro, this.diametro);  
 }  
 };  
 }

bola = new Bola(DIAMETRO\_BOLA);

E na função principal “desenhar” chame a função desenhar do objeto “bola”:

bola.desenhar();

Se tudo deu certo o nosso jogo deve estar assim:



Vamos criar um método “configuracoes” onde irá informar a posição Y que a bola irá começar, direção para que ela irá, ângulo e afins, para isso crie uma variável dentro da classe “Bola” chamada de “lado”, onde ela vai ser 1 para ir para a direita e -1 para ir para a esquerda (imagine isso como um plano cartesiano).

lado: 1,

configurar: function(lado) {  
 // sorteando uma posição aleatória para criar a nossa bola para colocar na posição Y  
 var r = Math.random();  
 this.x = lado === 1 ? jogador1.x + jogador1.largura : jogador2.x - this.diametro;  
 this.y = (ALTURA - this.diametro) \* r;  
 // calcular o ângulo, maior ângulo / inferior no eixo-y mais íngreme  
 var phi = 0.1 \* Math.PI \* (1 - 2 \* r);  
 // calculando a velocidade que a bola irá se mover  
 this.vel = {  
 x: lado \* this.velocidade \* Math.cos(phi),  
 y: this.velocidade \* Math.sin(phi)  
 }  
 },

E na função principal (main) chame o método “configurar(1)”:

bola.configurar(1);

Agora, precisamos fazer a nossa bola se movimentar, então vamos criar mais um método “atualizar” que será responsável por isso e detectar as colisões:

atualizar: function() {  
 // atualiza a posição com a velocidade atual  
 this.x += this.vel.x;  
 this.y += this.vel.y;  
 //checando se sai dos limites da tela na posição Y  
 if (0 > this.y || this.y + this.diametro > ALTURA) {  
 // calculate and add the right offset, i.e. how far  
 // inside of the canvas the ball is  
 var offset = this.vel.y < 0 ? 0 - this.y : ALTURA - (this.y + this.diametro);  
 this.y += 2 \* offset;  
 // mirror the y velocity  
 this.vel.y \*= -1;  
 }

// helper function to check intesectiont between two  
 // axis aligned bounding boxex (AABB)  
 var AABBIntersect = function(ax, ay, aw, ah, bx, by, bw, bh) {  
 return ax < bx+bw && ay < by+bh && bx < ax+aw && by < ay+ah;  
 };

// detectando colisão

var pdle = this.vel.x < 0 ? jogador1 : jogador2;  
 if (AABBIntersect(pdle.x, pdle.y, pdle.largura, pdle.altura,  
 this.x, this.y, this.diametro, this.diametro)  
 ) {   
 // set the x position and calculate reflection angle  
 this.x = pdle === jogador1 ? jogador1.x + jogador1.largura : jogador2.x - this.diametro;

var n = (this.y + this.diametro - pdle.y) / (pdle.altura + this.diametro);  
 var phi = 0.25 \* Math.PI \* (2 \* n - 1); // pi/4 = 45  
 // calculate smash value and update velocity  
 var batida = Math.abs(phi) > 0.2 \* Math.PI ? 1.5 : 1;  
 this.vel.x = batida \* (pdle === jogador1 ? 1 : -1) \* this.velocidade \* Math.cos(phi);  
 this.vel.y = batida \* this.velocidade \* Math.sin(phi);  
 }

// coloca a bola novamente no campo na posição X que ela saiu.  
 if (0 > this.x + this.diametro || this.x > LARGURA) {  
 if (pdle === jogador1)  
 jogador2.pontos += 1;  
 else  
 jogador1.pontos += 1;  
 this.configurar(pdle === jogador1 ? 1 : -1);  
 }  
 },

Ufa! Faça a chamada do método “configurar” do objeto “bola” criado, e a função “atualizar” na função principal “atualizar”.

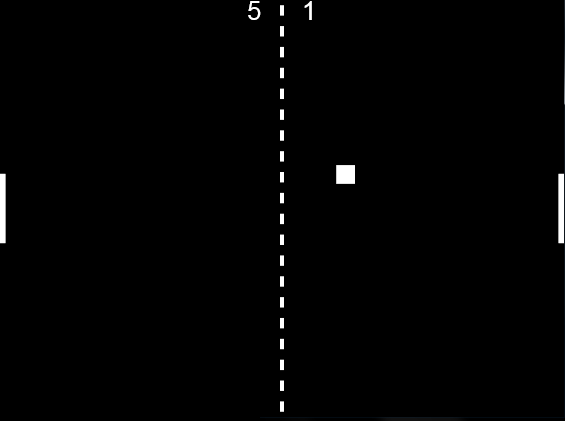
A função main no final das contas estará dessa forma:

function main(){  
 canvas = document.createElement("canvas");  
 canvas.width = LARGURA;  
 canvas.height = ALTURA;  
 ctx = canvas.getContext("2d");  
 document.body.appendChild(canvas);  
 bola.configurar(1);  
 var repintar = function() {  
 atualizar();  
 desenhar();  
 window.requestAnimationFrame(repintar, canvas);  
 };  
 window.requestAnimationFrame(repintar, canvas);  
 }

E a função atualizar estará assim:

function atualizar() {  
 bola.atualizar();  
 jogador1.atualizar();  
 jogador2.atualizar();  
 }

Pronto! O resultado esperado do game deve ser semelhante a esse:



Referência:

<https://github.com/maxwihlborg/youtube-tutorials/blob/master/pong/index.html>

<http://trackingjs.com/docs.html#introduction>