B0 - HUB

B-INN-000

JS WORKSHOP

Canvas HTML5 ans Javascript

0.1

JS WORKSHOP

language: JS

STEP 1 - BUILD A SCENE

00 - DISCOVER HTML5 CANVAS

Create a html file, "index.html", you should create a html structure document and a HTML5 canvas with a size of 380 by 280, you need to store these values for later use.

HTML5 canvas.

01 - GET THE CANVAS

You can retrieve any HTML objects with Javascript, you just need know the object id. You need to be sure the DOM is loaded before trying to have access the canvas.

HTML5 document.
Event interface with Web API for Javascript.
Window interface.

02 - Draw a line

Now that you have retrieved your canvas, you can draw a line inside it, to do this you need retrieve the 2D context of the canvas and draw a line.

Canvas Rendering

03 - Draw a rect

You need to draw a rect inside your canvas with the following dimension 8x8.

Canvas Rendering

03 - Draw a cercle

Try to draw a circle inside your canvas.

04 - Create a function to draw a line

As you have seen to display a line, you need to write at least 6 lines of code, why not create a function "draw_line(startX, startY, endX, endY, color)" to use it later.

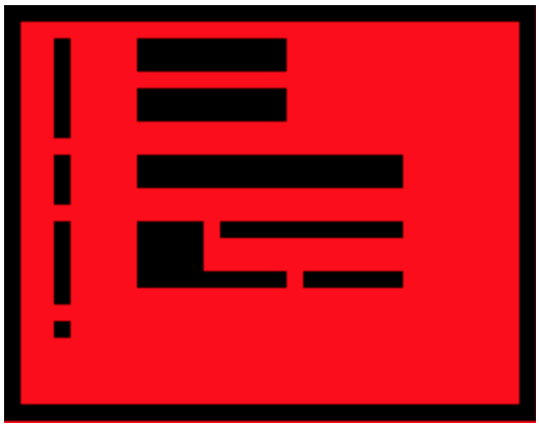
05 - Prepare your scene

You can to fill your canvas with a Rect with the size of your canvas and a default color like RED, it will represent your scene. To do this, you need to create function, called "clear_screen", you will use this function later to clear your scene on every frame.

06 - Create a game loop

To go forward, you need to create a game loop to refresh your scene. So to do it, when you known the DOM's contents are loaded by using the event listener, you can set an interval to call your game loop function every X millisecond, we will set the frame per second as 1000 / 30, is almost 30 FPS. Don't forget to call your clear_screen function at the begin this function to clear the screen before the next frame render.

STEP 2 - DRAW A MINIMAP



00 - CREATE AND STORE A MAP

Create a function that create an array to represent your map with the 32x24 blocks, this function need to return the array. For this workshop, we will represent the walls by the value 1 and empty cells by the value 0.

Examples:

Terminal

```
~/B-INN-000> cat map.js
var map = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,3,0,3,0,0,1,1,1,2,1,1,1,1,1,2,1,1,1,2,1,0,0,0,0,0,0,0,0,1],
[1,0,0,3,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,1,1,1,1],
[1,0,0,3,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
..........
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
```

01 - DRAW A MINIMAP

Create a function that take as parameter a map and draw a minimap inside a new canvas, "minimap". You nned to set the size of the cell, we will use the value 8 as a size, you can also set a scale value if you want manage the scaling.

Examples:

Terminal

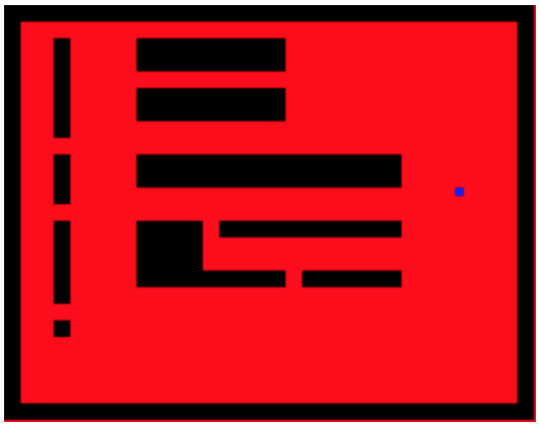
```
~/B-INN-000> cat draw_rect.js
function draw_rect() {
  var canvas = document.getElementById("canvas_scene");
  canvas.width = width;
  canvas.height = height;
  context = canvas.getContext('2d');
  context.fillRect(0, 0, 8, 8);
}
```

You can iterate through the map by looping through each nested array, and any time we need to access the wall type for a given block, we can get to it through a simple map[y][x] lookup.

02 - Where is the player ?

Now, we have displayed the minimap, we can represent the player before displaing it. So, you have two ways to do it, you can create a class or juste javascript object as you want. You need to have different variables, like the position, the angle of view and the speed at least.

After that, you can display your player on the minimap by representing it with a rect for example.



03 - Can you move your player ?

Try to move your player on the minimap by using your keyboard arrows Up and Down. If you use correctly the user's position when you show your player, it will move. You need to create a function "move_player" and call it to your game loop.

Event interface with Web API for Javascript.

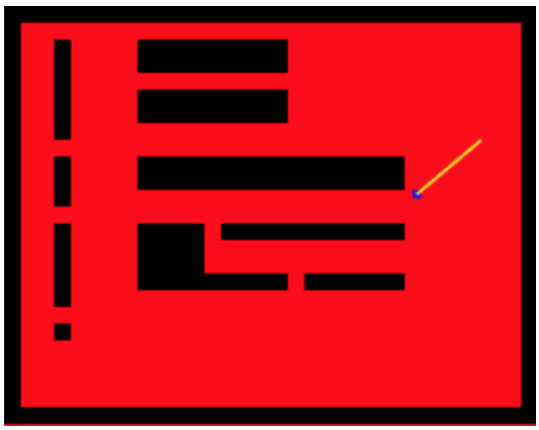
04 - Your player can move

Try to move the player's view by using your keyboard arrows Left and Rigth. If you use correctly the user's position when you show your player, it will move.

Event interface with Web API for Javascript.

05 - Show player view direction

Try to draw a line from the player's current position using the player's view angle.



Trigonometric functions.

06 - Get all rays to cast ray

You need to get all rays we want to display to use them to do some calcul. A ray is represent by the angle and the distance from the vertical or horizontal intersection of the player's view to the object. To calculate the distance between two coordinates, you can use a mathematical formala.
You need to creation a function, called "getRays" that will call castRay function.

Trigonometric functions.
Hypotenuse.
Math function Javascript.

Example a function to calculate horizontal intersect

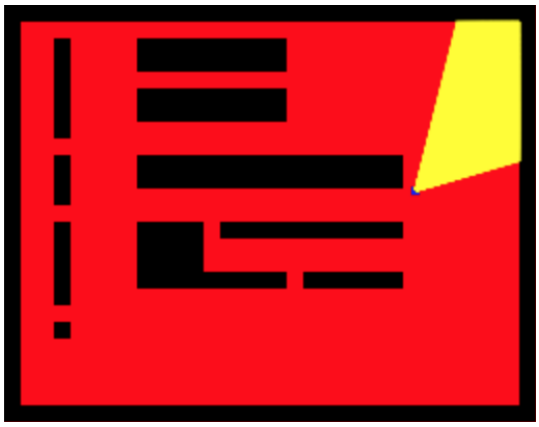
Terminal

```
~/B-INN-000> cat intersect.js
function getIntersectH(angle) {
//Check if the player's view is UP or DOWN
const isUp = Math.abs(Math.floor(angle / Math.PI) % 2);

const intersecY = isUp \?
    Math.floor(player.y \/ CELL_SIZE) * CELL_SIZE :
    Math.floor(player.y \/ CELL_SIZE) * CELL_SIZE + CELL_SIZE;

const intersecX = player.x + (intersecY - player.y) / Math.tan(angle);
const y = isUp ? -CELL_SIZE : CELL_SIZE;

const x = y / Math.tan(angle);

\//check if there are a wall
let wall;
let nextX = intersecX;
let nextY = intersecY;
while (!wall) {
    const cellX = Math.floor(nextX / CELL_SIZE);
    const cellY = isUp ? Math.floor(nextY / CELL_SIZE) - 1 : Math.floor(nextY /
        CELL_SIZE);

    if (outOfMapBounds(cellX, cellY)) {
        break;
    }
    wall = current_map.getmap[cellY][cellX];
    if (!wall) {
        nextX += x;
        nextY += y;
    }
}
return { angle: angle, distance: getDistance(player.x, player.y, nextX, nextY),
    vertical: false };

}
function castRay(angle) {
var collisionV = getIntersectV(angle);
var collisionH = getIntersectH(angle);

return collisionH.distance >= collisionV.distance ? collisionV : collisionH;

}
```

When you have done the same function to calculate the vertical intersect, you can fill an array that contains the angle and ray according the number of rays from the player's angle and the field of view, FOV. The FOV should be set in radians, we use 60 degree as the angle of view as value.

07 - Try cast rays from the player's view angle.

Now you have the rays, we can them to cast the rays from the player's view. So, for each ray you need to calculate the new position of X and Y coordinates to draw a line.



STEP 3 - DRAW A WALL

00 - MINIMAP SCALE DOWN

You should scale down your minimap before trying to display walls by using a scale variable.

01 - DISPLAY A CELL AND FLOOR

You can display a floor and a ceil before displaying the walls. Don't forget to divide by two some values.

02 - DISPLAY A WALL

Now, you need to create a function that will render you scene, to do it, you need to iterate on each ray you want to calculate the wall height according the CELL_SIZE and distance of the current ray. You need to define the distance between player eyes and the screen, we will use 300 as value.

03 - Collision time

Now, you can move across your map, but, you still don't have collision, so, can you implement a collision system management.