DIY : A Pacman clone using HTML5 Canvas

“If you've never played Pacman before, please get a life.” - Famous old bearded Greek guy

Gaming, these days, is synonymous with Call of Duty, DOTA 2, Counter Strike, etc. But there are these small games which might not be 3d but offer a lot of entertainment. As technology gets advanced and more accessible, making games has become easier. So, lets make Pacman!

You'll need:

Google Chrome – to read supplementary code. (Firefox works too, don't get me started on IE.)

Some Programming experience – you should know variables, functions and objects in some language.

We'll be using JavaScript for our game.

Let's dive in!

Crash Course in JavaScript -

Syntactically, it's like Java so put a semi-colon after every statement and put braces around blocks of code like functions, loops, if-else constructs, etc. But it has a dynamic type system so just say -

var n = 10; // var stands for variable

and JavaScript knows its a number. In JavaScript, functions are objects so you can pass them around as arguments to other functions. Functions are created like so -

function blah (arg1, arg2) { // named function, and this is a single line comment

console.log(arg1, “says, Hi I am arg1”); // notice the comma, it adds a space in the string.

console.log(arg2 + “ says, And I am arg2”); // or just use + and give a space yourself.

}

var funky = function (arg1) { // unnamed function, also called lambda functions

console.log(“What's in a name?”); // console.log prints given string to console

}; // notice the semicolon at the end

// calling the functions

blah(“HTML5”, “JavaScript”);

funk(“Not used argument”);

Lastly, you need objects! They are quite different in JS compared to Java. In JS, objects are just property:value pairs. Here is an object -

var pacman = {

x : 10,

y : 12,

f : function() { console.log(“This is how you define a method of an object in JS”);}

};

That's all you need. Pretty neat, eh?

Now go to [www.github.com](http://www.github.com/) and figure out how to download source code from it and then download my repository which is at - <https://github.com/amanj2718/pacman-clone.git> . It's got a file called base.html which we will be playing with a lot. Read it carefully as it introduces many new concepts in the comments. Understand the HTML structure so you can make your own file and play around with JS in your file. But mainly, base.html is pluggable so all the game code you learn and cook up goes in specific places in that file. If you ever get stuck, check out how the same problem is handled in pacman.html. Moving on to canvas animation!

HTML is a markup language to structure web content. It's got these different kinds of tags (also called elements) for different kinds of content. So you got one for headings, one for forms, another for images and so on. HTML5 introduces the canvas tag which is a drawing surface. There is no brush and paint, one draws on this surface using scripts which are written in JavaScript.

The general work-flow in an animation is to save a stable state, do your animation and then restore that stable state. When this is done rapidly enough, you get some delightful animations.

In the game, Pacman kept yapping away all the time. That's an animation we are going to reproduce now.

ctx.save(); // ctx is our context variable, the save() function saves the // current state of the canvas.

var open = true;

var x = 10;

var y = 10;

var openMouth = new Image(); // declare new Image objects

var closedMouth = new Image();

openMouth.src = “assets/open.png”; // tell JS where to find the images.

closedMouth.src = “assets/closed.png”;

openMouth.onload = function () {

closedMouth.onload = function () {

if(open)

{

ctx.drawImage(openMouth, x, y);

open = false;

}

else

{

ctx.drawImage(closedMouth, x, y);

open = true;

}

}

}

ctx.restore();

Try this out in a separate file and then in base.html. Finally, check out how its done in pacman.html.

User interaction in games is through handling keyboard events. We use a helper library called Kibo(https://github.com/marquete/kibo) to do that for us. It's extremely straightforward. Follow the link and learn it up and make pacman move according to arrow keys. Hack it up in your file and plug it into base.html.

The final part of this article is introducing the ghost. To keep it simple, instead of writing all the AI needed for a ghost, I am just including it as another user-controllable character. Refer to lines dash through dash in pacman.html and find out the key bindings of ghost. Play pacman.html with a friend for a while. As you see, game ends when the ghost and pacman collide and you'll have to refresh the page to restart the game. So let's cook up collision detection and plug it in base.html.

if ((pacman.x >= ghost.x && pacman.x <= ghost.x + ghost.width)

&& (pacman.y >= ghost.y && pacman.y <= ghost.y + ghost.height))

gameOver = true;

There are other details that we took for granted here like establishing game boundaries(was a particularly tough problem for me) and the drawing and eating of food but for completeness, their explanation is included in the pacman.html file. There's a lot more to do in this game though. If you find any bugs, fix them. You could add sound, more ghosts, more levels(maps, food patterns), maybe even introduce a completely new game mechanic. Google is your friend! :-)

Hope you enjoyed developing a functional(barely :-P) game. Happy hacking!