

08240 2D Computer Graphics: Lab2

Aim

The aim of this lab is to put in to practice the moveTo, lineTo, fill, stroke, beginPath, and closePath functions that we learnt about in the lecture this week. We are also going to be making a JavaScript object.

Task 1 - Setting up our Lab2 index.html file

- 1) Open Notepad++.
- 2) Use the Open file dialog to navigate to the Lab2 folder in your SVN checkout folder.
- 3) Select and open the 'index.html' file.

It should be empty as this was the blank file that you exported from the Core folder last week.

Saving some effort

Luckily for us the html file that we created in last week's lab remains largely unchanged. So why don't we save ourselves some typing.

- 4) Open last week's 'index.html' in Notepad++
- 5) Copy and paste its whole contents into our empty Lab2 'index.html' file.

Commit to SVN

Now that you have reached a milestone in your lab (creating the HTML file) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T1' on the first line on the log entry.**

Task 2 - Setting up our canvas.js file

Repeat the above process for the 'canvas.js' file in the javascript folder of Lab 2. Same as with the html file, save effort by copying and pasting the contents from last week's (Lab 1) javascript file in to this new file.

- 6) Next locate the draw function and select the contents.

```
// this function will actually draw on the canvas
function draw(pPosition) {
    // set the draw fill style colour to black
    context.fillStyle = "#000000";
    // fill the canvas with black
    context.fillRect(0,0,canvas.width,canvas.height);
    // choose a font for our message
    context.font = "40pt Calibri";
    // set the draw fill colour to white
    context.fillStyle = "#ffffff";
    // draw the text at the specified position
    context.fillText("Hello World!", pPosition.getX(), pPosition.getY());
}
```

- 7) Delete them, after all 'Hello World' was so last week.

```
// this function will actually draw on the canvas
function draw() {

}
```

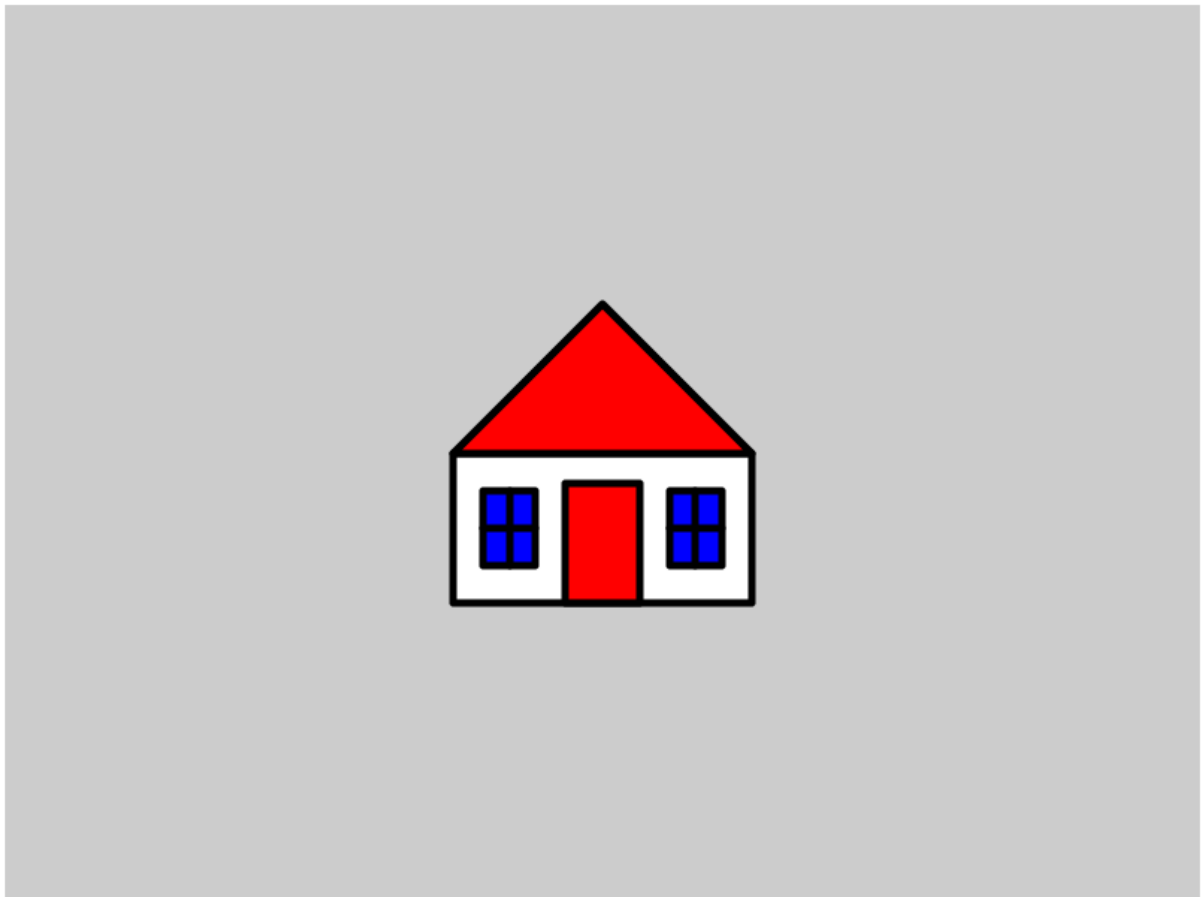
Commit to SVN

Now that you have reached another milestone in your lab (creating the javascript file) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T2' on the first line on the log entry.**

Now we are set up with a clean slate, we can start with the interesting part.

Task 3 - The house that [insert name] built

First of all we want to draw a house. This house:



To draw my house I used `beginPath`, `moveTo`, `lineTo`, `closePath`, `fillStyle`, `strokeStyle`, `fill`, and `stroke` which we covered in this week's lecture. If you need to refer to the lecture slides, you can get them on SharePoint. I also used the `fillRect` function, which we used in last week's lab, to fill the canvas with grey.

- 8) Use your knowledge of these functions to draw your own house in the now empty draw function.
- 9) Finally, the lines are not the default width. You can set this using the `lineWidth` attribute of the context object.

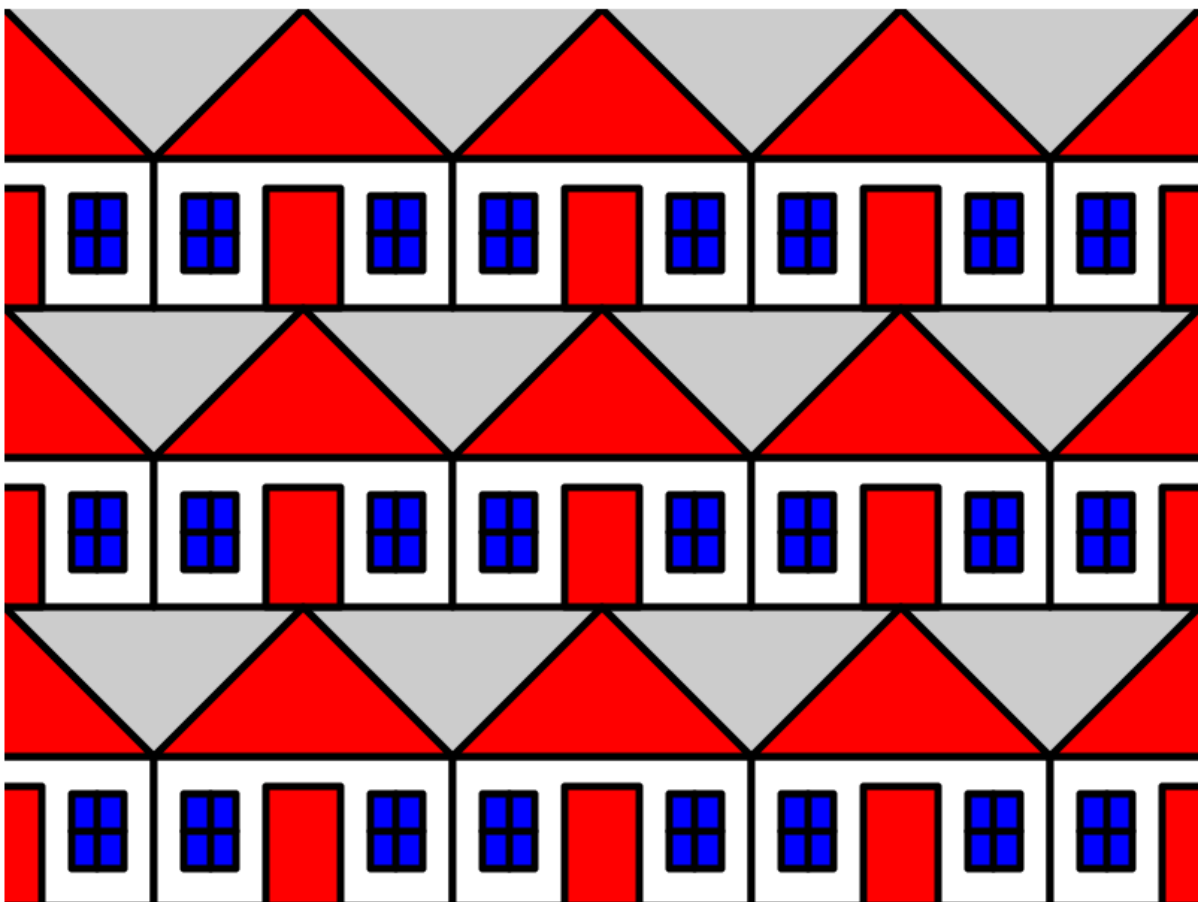
```
// this function will actually draw on the canvas
function draw(pPosition) {
    context.lineWidth = 5;
}
```

Commit to SVN

Now that you have reached another milestone in your lab (drawing your first house) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T3' on the first line on the log entry.**

Task 4 - I'll take 12

Sometimes when you see something that you like, you want more of it. So next we are going to alter our code to draw some more houses.



I hope that it is clear that you don't want to be copying and pasting the house code that you wrote and working out how to change the coordinate parameters of your `lineTo` and `moveTo` functions in each copy to put them in a different place; that would be boring, error prone, and a colossal waste of time.

This is what you will do instead.

- 10) Make a new function above your `draw` function and call it `drawHouse`. You want it to take a single parameter that is a position vector `pPosition`.

```
// this function will draw a house on the canvas
function drawHouse(pPosition) {

}
```

The pPosition vector will hold the x and y coordinates on the canvas that you want to draw your house at.

I object! I haven't got a Vector object

Yes you have, you made one in the previous lab, it is called 'vector.js'.

Task 5 - Using the vector for our drawHouse function

- 1) Select your house drawing code from the draw function and cut and paste it into the drawHouse function.

```
// this function will draw a house on the canvas
function drawHouse(pPosition) {
    // cut and paste your house drawing code here
}
```

- 2) Next you want to alter the code so that, instead of having absolute values for the moveTo and lineTo function calls, you make them relative to the x and y coordinates from the pPosition vector that you want your house to be draw at. For example:

```
// this function will draw a house on the canvas
function drawHouse(pPosition) {
    context.moveTo(100,100); // absolute bad
    context.moveTo(pPosition.getX()-10, pPosition.getY()-10); //
relative good
}
```

My personal preference is to make all my coordinates relative to the centre of the house. In my case this is the middle of the horizontal line of the bottom of the roof. Then when we call our function we are saying where to put the middle of the house. It is fairly intuitive then where you are positioning the house. Feel free to make your relative position different; another common anchor point is the top left most point of your house.

- 3) Once you have done this you can call the drawHouse function from your draw function several times from different coordinates to tile the drawings of your house. Note that we are passing the parameter from the draw function to do this.

```
drawHouse(pPosition);
```

- 4) If you are particularly clever you will use a for loop to handle the multiple draw calls.

A for loop in JavaScript is very similar to the for loop in C#. The only difference are that as JavaScript is weakly typed, you declare 'i' using 'var' rather than a type like 'int' that you would in C#. Another less strict difference is that by convention you increment your 'i' using 'i+=1' and not 'i++'.

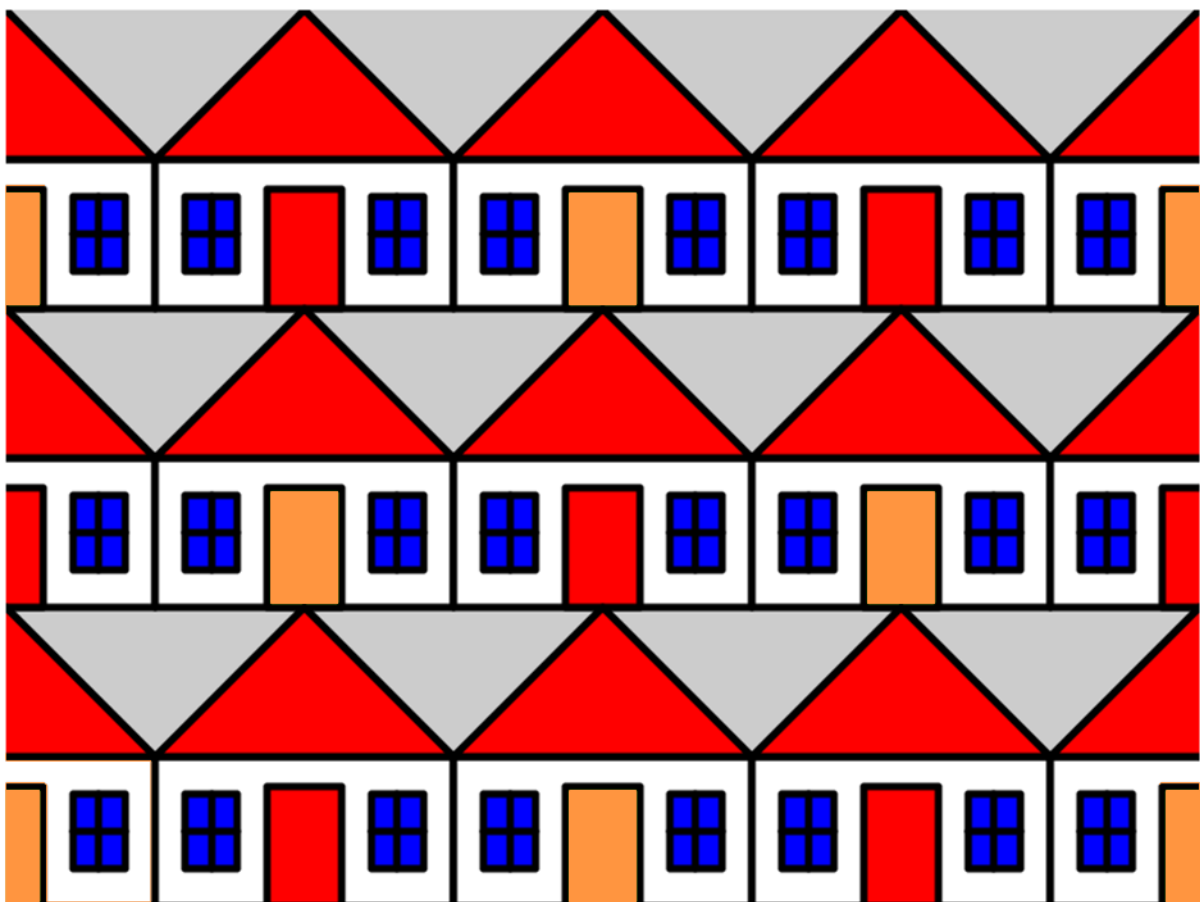
```
for(var i = 0; i < 5; i+=1){  
  }  
}
```

Commit to SVN

Now that you have reached another milestone in your lab (adding and using a draw house function) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T5' on the first line on the log entry.**

Task 6 - I see a red door and I want it painted orange (orange is the new black)

Now that you have a function for drawing houses, try adding a parameter to it that allows you to specify the colour of the door.



Commit to SVN

Now that you have reached another milestone in your lab (adding and using a pDoorColour parameter) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T6' on the first line on the log entry.**

Task 7 – Sub functions

You may have noticed that our drawHouse function is pretty cluttered with a big jumble of code for drawing all the parts of the house. In order to keep things tidy we should refactor the function to a number of sub functions to draw the different parts of the house ie. drawRoof, drawDoor, drawWall,

drawWindow. You will notice of course that you have more than one window, but in the same way that we don't need 12 drawHouse functions to draw 12 houses, we can call our drawWindow function twice with different position vectors.

I recommend that you do each sub function one at a time (maybe start with the drawWall function). Each time, replace the code for that part in the drawHouse function with a call to the new sub function. Committing after each sub function (see below).

Commit to SVN

Each time you add a sub function, commit it to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T7.x' on the first line on the log entry**. The 'x' in '7.x' should start at 1 and increment for each sub function that you add e.g. L2T7.1 added drawWall, L2T7.2 added drawDoor, etc.

I object (again)!

Having a vector object to store our position coordinates was useful. Now we are going to have a look at encapsulating the drawing behaviour in to a house object.

To keep things nice and tidy we are going to keep our house object definition in a separate file called 'house.js'.

Task 8 – Set up the 'house.js' reference in the 'index.html' file

Since we are going to be calling code from this file we need to add it to the list of scripts in the head of the html file.


```
<head>
  <title>Lab2</title>
  <script type="text/javascript" src="javascript/canvas.js"></script>
  <script type="text/javascript" src="javascript/vector.js"></script>
  <script type="text/javascript" src="javascript/house.js"></script>
</head>
```

We have finished with the html file now so go ahead and commit it.

Commit to SVN

Now that you have reached another milestone in your lab (adding the house.js reference) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T8' on the first line on the log entry**.

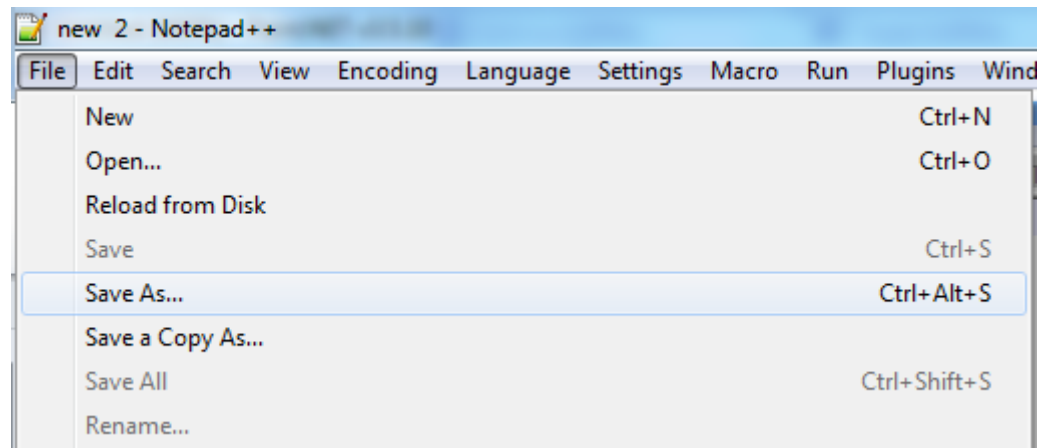
Task 9– Create the 'house.js' file

- 1) Click on the 'New file' icon.  This opens up a blank page.

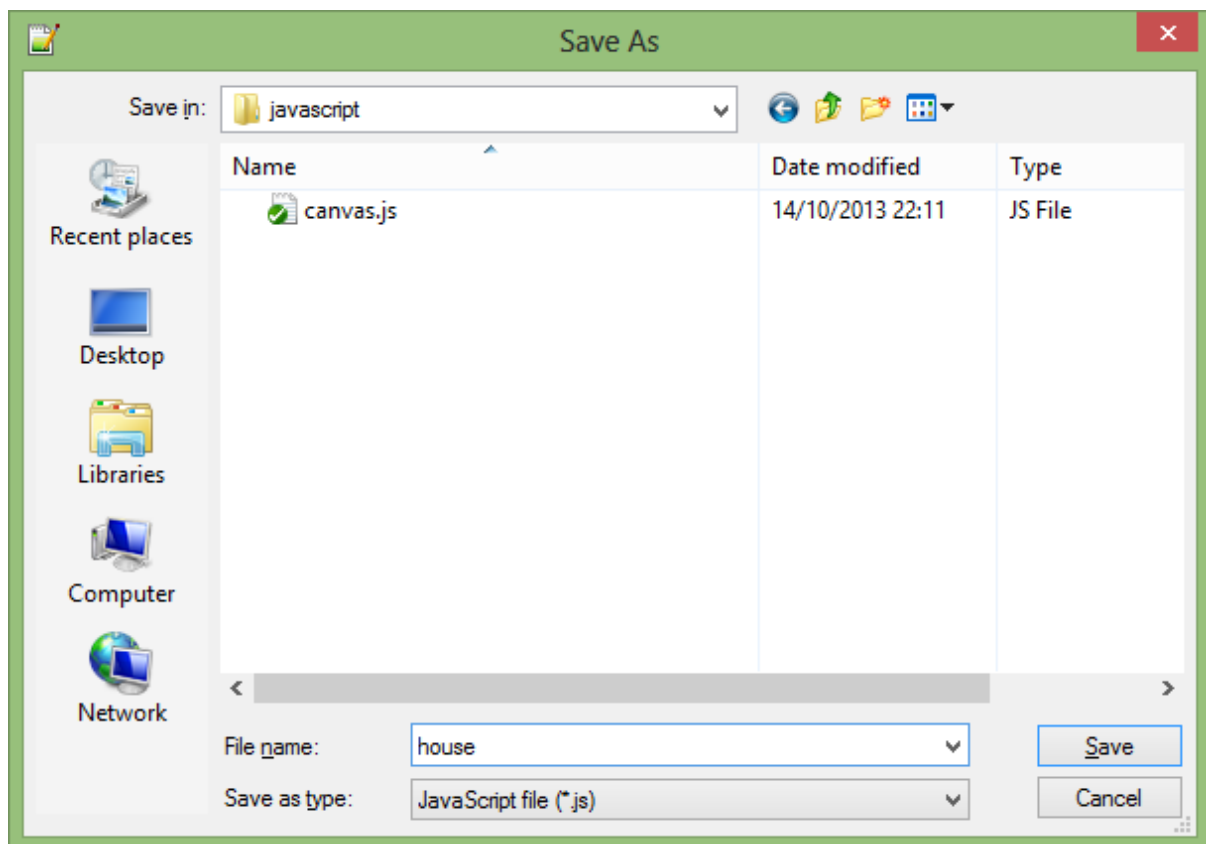
At the moment, by default, this is a normal text file; you can see this in the bottom left of the Notepad++ window. What we want though is a JavaScript (*.js) file.

Normal text file

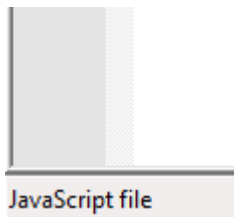
- 2) Using the menu, click File > Save As...



- 3) Select the JavaScript folder in your Lab2 folder and set the 'Save as type' to 'JavaScript file'. Type 'house' in to the 'File name' field and click on the 'Save' button.



You will notice the file type in the lower left of the Notepad++ window:



Commit to SVN

Now that you have reached another milestone in your lab (creating the house.js file) it is a good time to commit your work to SVN. You will need to 'add' the house.js file to SVN first. Don't forget to add a meaningful log message **starting with the tag 'L2T9' on the first line on the log entry.**

Task 10 – Writing our house object definition

Next we are going to define the House object in the house.js file starting with a constructor.

```
var House = (function () {  
    function House(pPosition, pDoorColour) {  
  
    }  
  
    return House;  
})();
```

In the above code we have defined an empty constructor that returns a House object.

Commit to SVN

Now that you have reached another milestone in your lab (defining our house object) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T10' on the first line on the log entry.**

Task 11 – Adding getters and setters to our house object

We also need some functions so we can actually use our house, starting with a getter and setter for the position vector of our house.

JavaScript uses a concept called prototypes to allow for inheritance between objects. It is not important to understand the details of this now, but if you are interested then you can review the object slides from the first lecture or try Googling 'javascript prototype'. For now just know that all objects have prototypes and you can add functions to them like below.

Note that I am using a 'p' at the start of my variables where they are a parameter (p for parameter) and an 'm' where the variable is a member variable of the object (m for member). This just makes it easier to distinguish them.


```

var House = (function () {
    function House(pPosition, pDoorColour) {

    };
    House.prototype.getPosition = function() {

        return this.mPosition;
    };
    House.prototype.setPosition = function (pPosition) {
        this.mPosition = pPosition;
    };

    return House;
})();

```

Now that we have a get and set function for the position vector we can use it in the constructor to store the x coordinate parameter.

```

function House(pPosition, pDoorColour) {
    this.setPosition(pPosition);
};

```

It is important to note the use of 'this' when calling the functions of the house object. In C# the use of the 'this' keyword can be implied, meaning that you can omit it. I.e. 'setPosition(pPosition)' and 'this.setPosition(pPosition)' are the same thing. This is not the case in JavaScript so don't forget your 'this'.

Repeat the process for the 'pDoorColour' parameter.

Commit to SVN

Now that you have reached another milestone in your lab (added getter and setter functions) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T11' on the first line on the log entry.**

Task 12 - The house draw function

Finally we are going to add a draw function to allow the houses to draw themselves. You need to give the draw function a context parameter which it can use to draw to. Cut and paste the contents of your previous drawHouse function into your new draw function for your house object. Now, in your draw function, replace each occurrence of 'pPosition.getX()' with 'this.getPosition().getX()', 'pPosition.getY()' with 'this.getPosition().getY()', and 'pDoorColour' with 'this.getDoorColour()'.

```

var House = (function () {
    function House(pPosition, pDoorColour) {
        // your constructor calls of the set functions are here
    };
    // your getter and setter functions are here

    House.prototype.draw = function(pContext) {
        pContext.moveTo(this.getPosition().getX()+45, this.getPosition().getY()+75);
        // example
    }
    return House;
})();

```

You will also need to make prototype functions for each of your draw sub functions (drawWall etc.).

Commit to SVN

Each time you add a sub function, commit it to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T12.x' on the first line on the log entry**. The 'x' in '12.x' should start at 1 and increment for each sub function that you add e.g. L2T12.1 added drawWall, L2T12.2 added drawDoor, etc.

Task 13 – One more thing

Now we are going to use the house objects in our canvas.js file. Locate the onLoad function and add a variable called houses to it. We are going to use this as an array to store our houses in.

```
function onLoad() {  
    var canvas, context, houses;
```

Next find the initialise function and at the bottom of it initialise the houses variable to be a new Array.

```
houses = new Array();
```

This makes a JavaScript array that behaves much like a C# List object. To add things to it we use the 'push' function. Use push to add some new House objects to the array.

```
houses.push(new House(new Vector(400,300), '#ff0000'));
```

Now we have made them we can draw them so locate the draw function. Write a for loop that will iterate as many times as the array is long using the array's length property.

```
for(var i = 0; i < houses.length; i+=1) {  
}
```

Access each house in the array by indexing the array with 'i' and call the draw function of the house in the array passing the context as a parameter.

```
for(var i = 0; i < houses.length; i+=1) {  
    houses[i].draw(context);  
}
```

Commit to SVN

Now that you have reached another milestone in your lab (using our house objects) it is a good time to commit your work to SVN. Don't forget to add a meaningful log message **starting with the tag 'L2T13' on the first line on the log entry**.

Challenge – Something extra special

Now you have the tools that you need, try making new objects in your scene. Perhaps you could use the arc function from the lecture to draw smoke from a chimney or to make flower petals for a

flower object. Use your imagination to draw me a pretty picture. If I see any that are particularly nice I will display them in next week's lecture for everyone to admire your magnificence.

Also don't forget: Commit to SVN! Starting with the tag 'L2T14 Challenge' on the first line on the log entry.

Summary

You created a function that would draw a house in a position relative to the coordinates that you gave it, with a door of a specified colour. Then you went one step further and created a house object that could draw itself.