Pits Of Doom Lesson 4: And Array We Go

In the last lesson we covered how to create a function, how to use some of php's built in functions, and how to deal with sessions. Now we'll take it even a step further.

If you've tried the little game we have setup right now you'll agree with me that's its not only boring, it's really REALLY boring. Right now you can click one button and either win or loose the game. In the big scheme of things we want our character to walk around different maps, moving from place to place as well as fighting or falling into pits before they die.

But frankly you're not at that point yet. So again we'll take a step back and go at from an easier approach — letting the character move around a larger area before they run into the possibility of either finding the treasure or falling into a pit. So let's say we have a map like this:

Map Name: Tutorial Map
Map Level: 1 (lower numbers are closer to the surface, higher numbers are deeper down)

```
W     W     W     W     W     W     W
W     S     E     E     E     E     W
W     W     E     X     E     E     W
W     E     E     E     T     X     W
W     W     W     W     W     W     W
```

S marks the starting place of our character on the map. The number E is an empty space, meaning the character can move there. The number W is a wall, the character can't move onto that spot. X is a pit and T is our treasure. Now we already have a much more interesting game already. In order to give our character more movement we can use a database or an array. Since a database is in a way a really huge array we're going to start with an array so you don't have to worry about all the finicky problems that pop up with the database and then move on to the database later.

Lesson Concepts

Arrays
In short an array is a way to hold lots information at one place. There are two types of arrays, single dimension arrays and multi-dimension arrays. We'll cover single dimensional arrays first.

The easiest way to think of an array is a list of items. The list always starts at zero and grows up in it's number. Let's take a look at a simple array I've made called path. In each of the fields in the path I've stored information about what kind of item you might find there (treasure, walls or empty spaces)

```
<?php
$path[0] = “S”;
$path[1] = “E”;
$path[2] = “E”;
$path[3] = “E”;
$path[4] = “W”;
?>
```

As you can see, out path has a starting position of a character, three empty spaces, and then a wall. If we were to try and build any kind of character movement using this array it would be pretty difficult. The character would start at position zero, be able to move up to 3 spaces forward, and then get stuck when it hits the wall at position 4. The only other option this character would have would be to turn around and go back to it’s starting position. How boring would something like this be? We want our character to be able to move in multiple directions, not just forward and back. That’s where multi-dimensional arrays come in. Take a look at this…

```
<?php
$path[0][0] = “W”; $path[0][1] = “W”; $path[0][2] = “W”; $path[0][3] = “W”;
$path[1][0] = “W”; $path[1][1] = “S”; $path[1][2] = “E”; $path[1][3] = “W”;
$path[2][0] = “W”; $path[2][1] = “E”; $path[2][2] = “E”; $path[1][3] = “W”;
$path[3][0] = “W”; $path[3][1] = “E”; $path[3][2] = “T”; $path[3][3] = “W”;
$path[4][0] = “W”; $path[4][1] = “W”; $path[4][2] = “W”; $path[4][3] = “W”;
?>
```

Does this look familiar? Good! It should! This is a multi-dimensional array. I’ve tried to write it out so it looks similar to the map we have planned out above, only this is a much smaller version. With a multi-dimensional array you can store more then 1 coordinate at a time. Now we can have a left and a right direction along with an up and a down.

In the multi-dimensional array our character starts out where we’ve marked it with an S, at (1,1). This notation (1,1) is what we’ll call a coordinate. It notes the x and y location (x,y) of the character.

If this character were to move up, we’d subtract one from the x coordinate. If they want to move down we add 1 to the x coordinate. If they want to move left or right, we add or subtract one to the y coordinate. Now what happens if the character starts at (1,1) and they move to the right? They are given the position (1,(1+1)) or (1,2). Scroll back up and look at the array. Is there a free space at (1,2)? If there is, then we’d want to allow this move. If there’s a wall in position (1,2) then we’d give the character an error message.

Now it’s your turn to try. Write out the coordinates you’d need to get from the character’s starting position to the treasure. There are actually several ways to do it, can you find more then one way?

Once we have a larger array we have to always test both coordinates when the character moves from place to place, otherwise we won’t be pulling accurate information from the maze. Here’s how you’d test to see what value is in an array at a certain position.

```
<?php
$x = 0; //first coordinate
$y = 0; //second coordinate
echo $path[$x][$y]; //now we show the value of the path for the given coordinates
?>
```

I'm sure you can see where this is going. You can test an array for a value just like you can test any other variable.

```
<?php
$x = 0;
$y = 0;
if ($path[$x][$y] == “E“)
{
echo “You're on an empty space!“;
}
else if ($path[$x][$y] == “W“)
{
echo “How did you get stuck in a wall?!?“;
}
?>
```

That's it! Seems pretty easy doesn't it?

Summary
In this lesson we covered single and multi-dimensional arrays. In order to complete this lesson you'll need to know how to assign values to a multi-dimensional array, access the values in the array, display the values in an array, and then write if-else statements to respond differently based on what's in the given position in the array.

To make this lesson easier I've included a custom function that draws the map to the screen and shows you the character's position as it moves around. The character is drawn in red until you find the treasure, then it turns green!

Game Files

Working Version: character.php — we have enough of a functioning game that you can start playing it for yourself!
Source Code: character.txt
Try it yourself: character2.txt

Lesson Reasoning

This lesson helps us work up to getting character movement based off of values from another data source. Right now we're using data we've entered ourselves, but pretty soon we'll be pulling this information from our database. This is only one level below reading from a database and it'll help you understand character movement on a baser level before we add the complication that is SQL.

Now comes the question, can we change the map as we play? Let's say, instead of the game being over when the character finds the treasure, we instead want them to move onto a new map. Sounds easy, but the more maps you have the more difficult this becomes!

And did you notice how much time it took to make your own custom map? Pff, we don't want to do that for all the levels in the game. We need something much faster and more powerful, which brings us to the next lesson.

In the next lesson you'll learn how to dynamically load different map files into the map array from a file instead of having to have it typed on the character's page. That way we can relocate the character to another map once they've found the treasure. Also, we'll create a map editor so we can quickly and easily make map files.

Once we have those two things done we'll be ready to move on to putting all the character movement into a database table, creating maps and saving them directly to the database, and letting our character move up and down in the map levels using ladders.

Find lesson 5 at http://design1online.com!