Lab Sheet Week 10

This week, you will integrate the game with a database to store the scores. As an example here, the game stops when the player reaches certain score (6 in this case). The term “stops” here means changing from the game state to game over state. In the game over state, the player can save score as well as the player’s name.

Before implementing the database to the game, you will learn to implement the States in different way. On the previous version (as seen on index.php), the states are declared when Phaser is initialised (line 9) and all states are implemented on the same file. Although it is useful for simple games, the disadvantage of this approach is the states are less flexible to be expanded or added and it is executed in a sequence.

The other approach is to implement the states dynamically in different JavaScript files so each state transition can be called manually on demand.

Task 1: Restructure the states

1. Load the resources

```
<script src="js/boot.js"></script>
<script src="js/preload.js"></script>
<script src="js/gametitle.js"></script>
<script src="js/thegame.js"></script>
<script src="js/gameover.js"></script>
```

2. Add the states to the game and then start the first state

```
game.state.add("Boot",boot);
game.state.add("Preload",preload);
game.state.add("GameTitle",gameTitle);
game.state.add("TheGame",theGame);
game.state.add("GameOver",gameOver);
game.state.start("Boot");
```

3. Implement the states on each js file (e.g: boot.js) and point to the next state

```
var boot = function(game) {
    console.log("%cStarting my awesome game", "color:white; background:red");
};

boot.prototype = {preload: function() {},
create: function() {
                this.game.state.start("Preload");
        }
}
```

4. Load the other resources in the preload state

5. Create transition from game state to game over state when the score is 6 (on function collisionHandlerScoring in Game state) and then pass the score to the game over state

```
If (this.score > 5)
    this.game.state.start ("GameOver",true,false,this.score);
```

6. In Game Over state, receive the variable sent from Game State

```
init: function (score) {
        this.theScore = score;
        // Alert ("You scored: " + score)
}
```

7. In Game Over state, add the text showing the score and a button to save

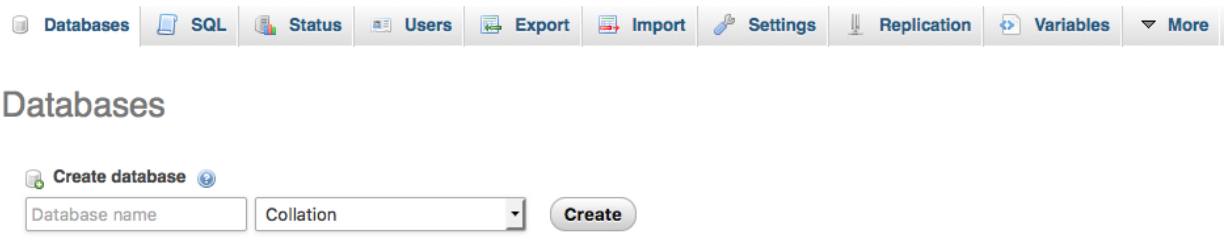
```
this.game.add.text (5, 250, 'This is the end of game. Your score is ' + this.theScore + '.',
{ fill: '#ffffff', font: '14pt Arial' });
var Thebutton = this.game.add.button (400, 230, 'buttonSprite', this.actionOnClick, this, 2,
1, 0);
Thebutton.onInputDown.add(this.onDown, this);
```

Task 2: Create database

1. Open PHPMyAdmin on XAMPP

```
// localhost:<port name>/phpmyadmin
```

2. Click “Databases” on PHPMyAdmin menu on the top options panel

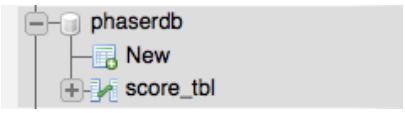


3. Input the database name (in this case, lets input “phaserdb”)

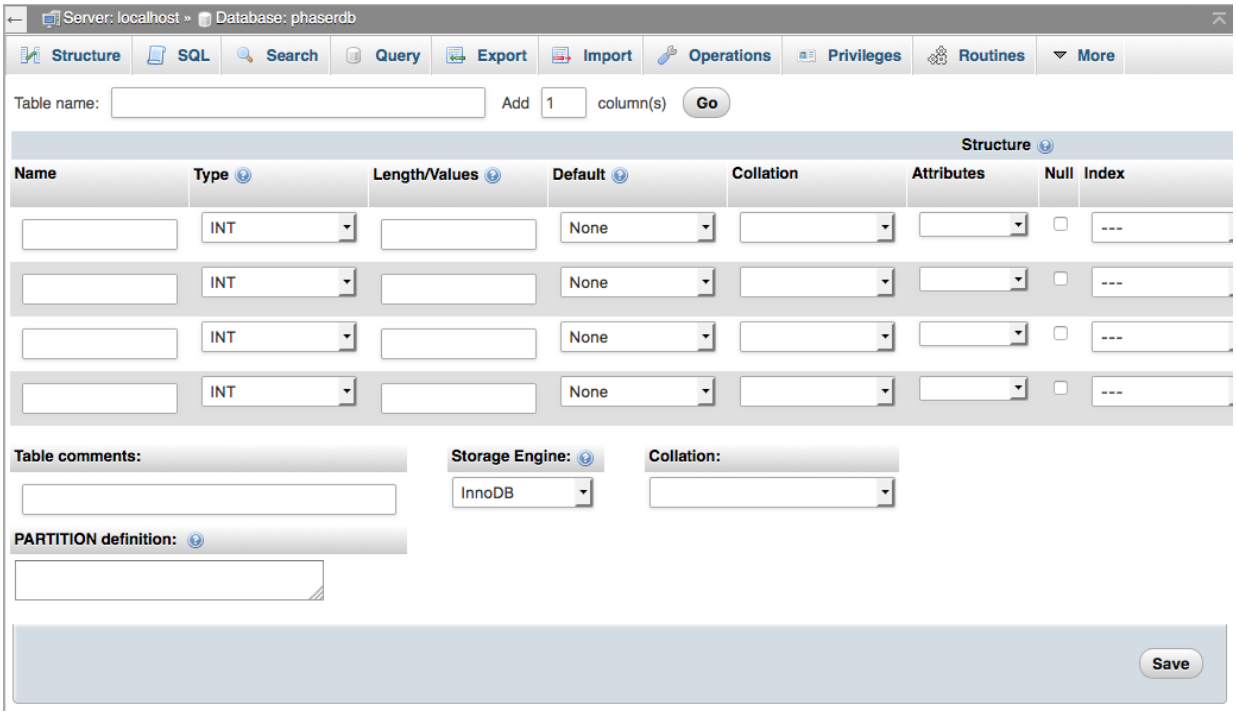
4. Once the database is created, the name of the database you created should appear on the list on the left side panel.

5. If you need to delete a database, click “Databases” on PHPMyAdmin menu on the top options panel and, tick the database you want to delete and then click “Drop” button below the list. Please remember that PHPMyAdmin has default databases which you should not delete.

6. To add table to the database, click the sign “+” on the name of the database on the left panel and then click “New”.



7. Input the table name (in this case it is called “score_tbl”, the name and the type of the column. In this case, the first column should be: 1) userid with varchar data type and 25 character in length, and 2) userscore with integer data type. Once you saved it, the created table will appear on the left panel under the database it belongs to.



8. If you need to modify or remove the table, click the table name on the left panel and then click “Structure”. To modify, click “Change” and to remove the table, click “Drop”

Task 3: Sending the data

Sending data from JavaScript using HTTP when the button on the Game Over State is clicked. There are two data to be sent: the score and the player name which the player can input

1. Create a popup textbox

```
var person = prompt("Please enter your name", "Harry Potter");
```

2. Prepare the variables and values to be passed onto the server. The variables and values will be formatted as a JavaScript object

```
var obj = {'user': person, 'thescore': this.theScore };
```

3. If the input is not empty, pass the variable to the server

```
if (person != null) {}
```

4. Initiate XMLHttpRequest method

```
var xhr = new XMLHttpRequest();
```

5. Specify the type of the passing and the target file. In this case the method is “Post” and the target file is the PHP file called “savescore.php”

```
xhr.open('POST', 'savescore.php');
```

6. Set the file transfer header

```
xhr.setRequestHeader("Content-Type", 'application/x-www-form-urlencoded');
```

7. Format the data to JSON

```
jsonData = JSON.stringify(obj);
```

8. Create a condition to check if the transfer of the JavaScript object containing the variables and values is successful

```
xhr.onreadystatechange = function() {
        if (xhr.status === 200) {
                // An error from the database
                alert(xhr.responseText);
        }
};
xhr.send('json=' + jsonData);
```

Task 4: PHP Files

Create PHP file to handle the parameters sent from the client (JavaScript) to the database.

1. Create connection between PHP to the database

```
// Connect to the database(host, username, password)
$con = mysql_connect('localhost','root','');
```

2. Select which database to send the query to

```
// Select the database. Enter the name of your database (not the same as the table name)
$db = mysql_select_db('phaserdb');
```

3. Check if there is a data with the parameter called “json” being passed from Javascript. We use $_POST because we declared POST in JavaScript

```
if (isset($_POST['json'])) {}
```

4. Transform the JSON sent from JavaScript to an array

```
$request_json = json_decode($_POST['json'], true);
```

5. Access the variables and values from the array. The variables are the same as when declared in JavaScript

```
$theuser = $request_json['user'];
$thescore = $request_json['thescore'];
```

6. Create a SQL Insert Statement and execute it

```
$sql = "INSERT INTO score_tbl (userid,userscore) VALUES
('".$theuser."',$thescore)";$query = mysql_query($sql);
```

7. Exporting SQL

Click “Export” on the top menu and choose “SQL” as the file type. The generated file will have an extension .sql

8. Importing SQL

Click “Import” on the top menu and choose the file you wish to import

9. Testing SQL statement

Choose a database and click “SQL” on the top menu. You can try to retrieve all data from the table you typing “SELECT * FROM score_tbl;

Further Task:
Save your game data to database
Retrieve data from database and use it in your game