```
1   /**
2   Copyright © 1997-2016, Stephen Gose LLC. All rights reserved.
3   Visit our game show case:   http://www.renown-games.com
4   License information:        http://makingbrowsergames.com/starterkits/
5   Affiliate Information:
    http://www.sgose.com/products/affiliates-program/
6
7    * The above copyright notice and this permission notice shall be
     included in
8    * all copies or substantial portions of the Software.
9    *
10   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
     EXPRESS OR
11   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
     MERCHANTABILITY,
12   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
     SHALL THE
13   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
14   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
     ARISING FROM,
15   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
     IN
16   * THE SOFTWARE.
17   *
18   *
19   * File Name: index.html
20   * File URL:
     http://makingbrowsergames.com/starterkits/quiz/game1/index.html
21   * Description: File for controlling and displaying game scenes as a
22   * single web page application; managing global variables throughout game
     state.
23   * Author: Stephen Gose
24   * Version: 0.0.0.18
25   * Author URL: http://www.stephen-gose.com/
26   * Support: support@pbmcube.com
27   *
28   * Do not sell! Do not distribute!
29   * This is a licensed file. Please refer to Terms of Use
30   *   and End Users License Agreement (EULA).
31   * Search for [ //**TODO** ] to tailor this file for your own use.
32   * Changes will void any support agreement.
33   *
34   * Redistribution of part or whole of this file and
35   * the accompanying files is strictly prohibited.
36   *
37   */
38  //game set-up
39  console.log("%c      Starting my awesome MMoG game Prototype!
    \n          Phaser Quiz & Trivia Starter Kit         \n        Copyright
    \u00A9 1974-2017,  Stephen Gose.      \n  | http://shop.pbmcube.net/
                          \n  | \u2665\u2665\u2665\u2665\u2665 -> $120
    License included in book!        \n  | Book available at:
    http://leanpub.com/LoRD |   ",
40      "color:white; background:blue");
41
42  //Game variables
43  var answerBtn = [];     //current answer selection stored here
44  var AnswerTxt = [];     //answers labels sit on top of answer buttons
45  var correctAnswers = 0; // track of correct answers
```

```
46  var highScore;          // preserve highScore in local storage
47  var hudText = "";       // heads up display dynamic text
48  var gamePhase = false;  //sentinel for game session
49  var localStorageName = "mathQuiz";
50  var maxSum = 10;        //maximum questions asked per quiz
51  var nextLevel = 400;    // score needed to increase difficulty
52  var progress = 1;       //which question we're on?
53  var questions = [];     // pool of questions
54  var score = 0;          //current achieved score.
55  var scoreText = "";     //score is concatenated with HUD
56  var thisQuestion = "";  //Math question composed
57  var questionText = "";  //question text dynamically updated
58
59  //text styles
60  var styleHUD = {font: "bold 20px Arial",fill: "#FFF",align:"left"};

61  var styleQ = {font: "bold 32px Arial",fill: "#FFF",align:"center"};

62  var styleN = {font: "bold 9px Arial",fill: "#000",align:"center"};
63  var styleB = {font: "bold 16px Arial",color: "blue",fill: "#444",align:
    "center"};
64
65  //Build the questions' data structure
66  //create a question pool for this quiz as an objects array
67  // Instead of a newly generated question per turn, it is easier
68  // to store all generated questions into an array (pool) then
69  // march through the list of questions using array index.
70  //So, let's use a loop to create maximum questions
71  // ranging from 1 to our maxSum variable
72  for(var i = 0; i <= maxSum; i++){
73      var v1 = Math.floor((Math.random() * 6 + 1));
74      var v2 = Math.floor((Math.random() * 6 + 1));
75      var cA = v1 + v2;       //correct answer
76      var iA1= v1 + v2 + 1;   //incorrect answer1
77      var iA2= v1 + v2 - 1;   //incorrect answer2
78      var iA3= v1 + v2 + 2;   //incorrect answer3
79      var iA4= v1 + v2 - 2;   //incorrect answer4
80
81      // defining questions[i] as a composite of 3 array
82      // remember that arrays start counting from 0 not 1
83      //Design Note:
84      // this version is "locked" into addition; we could make
85      // opns a random math operation of +, -, * , or /
86      // we could even to far as to make the gamer guess the
87      // correct operations. For example: 1 ? 1 = 2
88      questions[i]={
89          ndx: i,         //question index number
90          first: v1,      //value #1
91          opns: " + ",    //change operations; future enhancement
92          second: v2,     //value #2
93          correct: cA,    //correct answer
94          allAnsw: shuffleAnswer(cA,iA1,iA2,iA3,iA4)
95      };
96  }
97  //debug quiz question pool
98  console.log("=== Quiz Questions: ");
99  console.log(questions);
100 console.log("=== End Questions ===");
101 // ========================================================
```

```
102  // use Fisher-Yates Shuffle after combining all possible answers.
103  // answers must be shuffled and NOT sorted because iA are consistently
104  // plus/minus 1 or 2;
105  function shuffleAnswer(ca,ia1,ia2,ia3,ia4){
106      // take all possible answers and create an array
107      var answerArray = [ca,ia1,ia2,ia3,ia4];
108      // set up Fisher-Yates variables
109      var ctr = answerArray.length, temp, index;
110
111      // While there are elements still in the array
112      while (ctr > 0) {
113      // Pick a random index
114      index = Math.floor(Math.random() * ctr);
115      // Decrease ctr by 1
116      ctr--;
117      // And swap the last element with it "Fisher-Yates Shuffle"
118      temp = answerArray[ctr];
119      answerArray[ctr] = answerArray[index];
120      answerArray[index] = temp;
121      }
122      // return the newly shuffled answer array to this question
123      return answerArray;
124  }
125
126  // ========================================================
127  // Non-traditional single-state game deployment
128  //    playGame has several internal phases/states
129  //    0) intro/pre-game setup
130  //    1) present a quiz question and await a response
131  //    2) gamer responds to question, then grade/check/validate input
132  //    3) reveal verdict on this answer
133  //    4) if no more questions? quiz has finished
134  // ========================================================
135  // Creating a JavaScript Game Object Linking to Phaser.Game object
136  var playGame = new Phaser.Class({
137
138      Extends: Phaser.Scene,
139      initialize: function playGame () {
140          Phaser.Scene.call(this, { key: 'playGame' });
141      },
142      //Phaser III Essential Functions per Scene
143      // when our single play state preloads ...
144      // it is a substitute for the formal boot and preload js files
145      preload: function(){
146
147          //-- preload.js ---
148          //The following are typically performed in the preload.js
149          // preload game background image
150          this.load.image("bkgrnd", "assets/blackboard.jpg");
151
152          // preloading a spritesheet where each sprite is 387x30 pixels
153          this.load.spritesheet("buttons", "assets/mmog-sprites-silver.png"
             ,{frameWidth:129, frameHeight:30});
154          //-----
155      },
156
157      // ========================================================
158      // prior to starting the single Play state
159      create: function(){
```

```
160             //place background theme image for the game
161             // I chose a blackboard; does anyone remember blackboards?
162             this.add.image(config.width/2,config.height/2,"bkgrnd");
163
164             // preserve the highScore value in local storage
165             //  if anything; otherwise set to 0
166             highScore = localStorage.getItem(localStorageName) == null ? 0 :
                localStorage.getItem(localStorageName);
167
168             //Prepare the answer button input menu:
169             // loop through the 5 standard answer buttons
170
171             // create answer buttons.
172             for(var i=0;i<5;i++){
173                 answerBtn[i] = this.add.sprite(75+(75* i),(config.height/2)+53
                    , "buttons");
174             }
175
176             // take each button sprite and set additional parameters
177             // centering dynamic text on top of it associated button
178             answerBtn.forEach( function (key, index) {
179                 key.name = "aBtn"+index;    //used later
180                 key.setInteractive({ useHandCursor: true });//visual clue
181                 key.setFrame(1);            //sprite button starts on frame 1
182                 key.value = questions[progress].allAnsw[index];
183                 key.alpha = 1;  //future enhancement; adjust alpha?
184                 //refer to "Phaser III Game Prototyping" pg 137
185                 // mobile button size should be a minimum of 44x44 pixels
186                 // available from LeanPub.com
187                 // - https://leanpub.com/phaser3gamedesignworkbook or
188                 // - https://leanpub.com/phaser3gameprototyping
189                 key.setScale(0.5,0.8);  //make it big enough for mobile
190                 key.setOrigin(0);       //gfx different than text
191                 key.enableBody = true;
192
193                 //Monitor gamer's activities
194                 key.on('pointerover',function (pointer){
195                 console.info(key.name + " over. ");key.setFrame(2);}, this
                    );    //mostly debug here
196                 key.on('pointerup',function (pointer){
197                 console.info(key.name + " clicked. ");key.setFrame(0);}, this);
198                 key.on('pointerout',function (pointer){
199                 console.info(key.name + " off. ");key.setFrame(1);}, this);
200                 key.on('clicked',this.checkAnswer, this);
201
202                 //Setup dynamically displayed label on top of this button
203                 AnswerTxt[index] = this.add.text(108+(75* index), (config.
                    height/2)+65, questions[progress].allAnsw[index], styleB);
204                 AnswerTxt[index].setOrigin(0.5,0.5);
205
206             }, this);
207             //console.log("AG: "+
                this.AnswerGroup.children.entries.length);    //debug
208             console.log(answerBtn); //debug
209             console.log(AnswerTxt); //debug
210
211             //Create math question:
212             // `thisQuestion` is the currently composed quiz question
213             thisQuestion = (questions[progress].first + " + " + questions[
```

```
            progress].second + " = ??");
214         // questionText is the text object content displayed
215         questionText = this.add.text(config.width/2 , (config.height/2)-20
            , thisQuestion, styleQ);
216         // make it pretty
217         questionText.setShadow(3, 3, 'rgba(0,0,0,0.7)', 5);
218         // setting questionText registration point from 0,0 default
219         questionText.setOrigin(0.5,0.5);
220         //-----
221         //Setup HUD:
222         // scoreText will keep the current score
223         hudTxt = "Question #: "+questions[progress].ndx+" (of "+maxSum+
            ")\nScore: "+score+" ("+correctAnswers+" of "+maxSum+" )";
224         scoreText = this.add.text(35, 18, hudTxt, styleHUD);
225         //-----
226         //Setup Copyright notice:
227         // create dynamically updated year on notice
228         var d = new Date();         //get JavaScript Date
229         var n = d.getFullYear();    //get full year
230         //Design Note: ( C ) is NOT a legal copyright notice
231         // read more on proper copyright notices in
232         // Phaser Game Design Workbook available at
233         // https://leanpub.com/phaserjsgamedesignworkbook
234         var Copr = this.add.text(120, config.height-20,"Copyright ©
            1974-"+n+", Stephen Gose. All rights reserved.",styleN);
235         Copr.setShadow(3, 3, 'rgba(0,0,0,0.5)', 5);
236
237         // If a Game Object is clicked on, this event is fired.
238         //  We can use it to emit the 'clicked' event on the game object
            itself.
239         this.input.on('gameobjectup', function (pointer, gameObject)
240         { gameObject.emit('clicked', gameObject); }, this);
241         //  and a "roll over" activity monitoring event
242         this.input.on('gameobjectup', function (pointer, gameObject)
243         { gameObject.emit('pointerover', gameObject); }, this);
244     },
245
246     // ========================================================
247     update: function(){
248         //New use of update: it becomes an finite state engine sampling @
            ~60fps
249         // it uses a switch to only perform code
250         // for the current phases/states
251         //    playGame has several internal phases/states
252         //    0) intro/pre-game play
253         //    1) present the quiz question and await response
254         //    2) gamer responds to question, check/validate input
255         //    3) reveal verdict on answer
256         //    4) quiz finished
257
258         //check for more questions?
259         if (progress < maxSum+1){
260             //dynamically update question text
261             hudTxt = "Question #: "+questions[progress].ndx+" (of "+maxSum
                +")\nScore: "+score+" (Correct: "+correctAnswers+" of "+maxSum
                +" )";
262             scoreText.setText(hudTxt);
263             //future enhancement: change/track math operations as +, -,
                /, *
```

```
264                 thisQuestion = (questions[progress].first + " + " + questions[
                    progress].second + " = ??");
265
266                 switch(gamePhase){
267                     case 1: //present the quiz question and await response
268                         //dynamically update question and answer text
269                         questionText.setText(thisQuestion);
270                         AnswerTxt[0].setText(questions[progress].allAnsw[0]);
271                         AnswerTxt[1].setText(questions[progress].allAnsw[1]);
272                         AnswerTxt[2].setText(questions[progress].allAnsw[2]);
273                         AnswerTxt[3].setText(questions[progress].allAnsw[3]);
274                         AnswerTxt[4].setText(questions[progress].allAnsw[4]);
275                     break;
276                     case 2: //gamer responds to question, check/validate input
277                         this.input.on('pointerover',this.btnOver, this);
278                         this.checkAnswer();
279
280                     break;
281                     case 3: //reveal answer verdict
282                             //future enhancement: provide praise or scold gamer
283                             //future enhancement: game timer - how quickly
                            answered
284                     break;
285                     case 4: //game over; show score; submit & tweet score
286                         scoreText.setText(hudTxt);
287                         questionText.setText("Game Over.");
288                         for(var i=0;i<5;i++){
289                             answerBtn[i].setVisible = false;
290                         }
291                         //future enhancement: Play Again button
292                         //future enhancement: new game phase & leader-board
293                         //future enhancement: social media bragging rights
294                         gamePhase = 0;  //change internal game phase to "Intro"
295                     break;
296                 }
297
298             }else{ //no more questions in the pool
299                 //dynamically update question text
300                 scoreText.setText(hudTxt);
301                 questionText.setText("Game Over.");
302                 this.gameOver();
303             }
304
305         },
306
307         // ========================================================
308         // ----------------------------------------
309         // Supporting game Function & Classes
310         //  *** (functions in alphabetical order) ***
311         // ----------------------------------------
312         // **TODO**:
313         // - Change name-space to some generic GAMEAPP for your project
314         // - re-factor and adjust for your game deployment
315         // - remove console debug information before public deployment
316         // ========================================================
317         //
318         answeredQ: function(){
319             //avoid race condition between Phaser and human response time
320             var temp = questions[progress].ndx;
```

```
321             progress = temp;
322
323         },
324         //
325         // ============================================================
326         btnOver: function (butn){
327             //finds currently selected button by assigned name
328             // this is mostly a debug function, but supports future
                enhancements
329             switch(butn.name){
330                 case "aBtn0":
331                     console.log("over Answer 1; value: "+ butn.value);
332
333                 break;
334                 case "aBtn1":
335                     console.log("over Answer 2; value: "+ butn.value);
336
337                 break;
338                 case "aBtn2":
339                     console.log("over Answer 3; value: "+ butn.value);
340
341                 break;
342                 case "aBtn3":
343                     console.log("over Answer 4; value: "+ butn.value);
344
345                 break;
346                 case "aBtn4":
347                     console.log("over Answer 5; value: "+ butn.value);
348
349                 break;
350             }
351         },
352         // ============================================================
353         //check the submitted answer
354         checkAnswer:  function(butn){
355             // check the answer only if the game has not finished yet
356             var currentQ = progress;
357             //debug
358             console.log("Value submitted: "+butn.value);
359             console.log(" Btn value: "+butn.value+"; Correct Answer: "+
                questions[currentQ].correct);
360             //-----
361             butn.setFrame(1);   //reset button to normal frame
362             //assign new game phase in the update function's finite state
                machine
363             gamePhase = 3;      //allow future enhancements here
364
365             // is button chosen the correct answer?
366             if(butn.value == questions[currentQ].correct){
367                 //adjust scoring variables
368                 score += 10;
369                 correctAnswers += 1;
370                 hudTxt = "Question #: "+questions[progress].ndx+" (of "+maxSum
                    +")\nScore: "+score+" (Correct: "+correctAnswers+" of "+maxSum
                    +" )";
371             }
372             //whether correct or incorrect move onto the next question
373             this.nextQuestion();
374         },
```

```
375
376         // ============================================================
377         // this ends our game. The argument is the stuff to save
378         gameOver:  function(){
379
380             // now it's game over time; provide a notice
381             questionText.setText("Game Over.");
382
383             // updating and save this score into local storage
384             localStorage.setItem(localStorageName, Math.max(this.score, this.
                highScore));
385         },
386
387         // ============================================================
388         // moving to the next question
389         nextQuestion:  function(){
390             //reset values and text labels
391             for(var i=0;i<5;i++){
392                 //reset button values
393                 answerBtn[i].value = 0;
394                 //debug: console.log("Butn Reset:" +answerBtn[i].name +";
                    "+answerBtn[i].value);
395                 //reset button label values
396                 //AnswerTxt[i].setText("");
397                 //debug: console.log("Butn Label Reset:" +AnswerTxt[i].text
                    +"; _TXT: "+AnswerTxt[i]._text);
398                 AnswerTxt[i].destroy();
399                 console.log("Destroyed: old answer button text labels.\n");
                    //debug
400             }
401
402             //debug check
403             console.log(AnswerTxt); //debug
404             console.log("LN 403 Last question was: #"+progress);    //debug
405
406             if (progress <= maxSum){
407                 //use internal question index; this avoid race conditions
408                 //var temp = questions[progress].ndx;
409                 //progress = temp;
410                 //console.log("Temp question #: "+progress);    //debug
411                 progress += 1;  //next question count (here only)
412                 gamePhase = 1;  //change internal game phase
413
414                 //Reset all button values and their label values with new
                    question
415                 console.log("LN 414 Now question is: #"+progress);  //debug
416                 console.log("QQQ Index is: "+questions.length); //debug
417
418                 //RECheck the Quiz Question Queue since progress was increased
419                 if(progress > maxSum){
420                     //future enhancement: present Play Again button??
421                     gamePhase = 4;  //change internal game phase to "Game Over"
422                     return;
423                 }
424
425                 for(var i=0;i<5;i++){
426                     //load new button values
427                     answerBtn[i].value = questions[progress].allAnsw[i];
428                     //console.log("New Butn value for: " +answerBtn[i].name
```

```
                    +" is "+answerBtn[i].value);    //debug
429                 //reset button to normal frame
430                 answerBtn[i].setFrame(1);
431
432                 //Setup dynamically displayed labels on top of their
                    buttons
433                 //create new button label values
434                 AnswerTxt[i] = this.add.text(108+(75* i), (config.height/2
                    )+65, questions[progress].allAnsw[i], styleB);
435                 AnswerTxt[i].setOrigin(0.5,0.5);
436             }
437             console.log("Rebuilt: new text labels.");
438             console.log(answerBtn); //debug
439             console.log(AnswerTxt); //debug
440
441         }else{
442             gamePhase = 4;  //change internal game phase to "Game Over"
443         }
444     }
445
446 }); //End JavaScript Game Object Linking to Phaser.Game
447
448 //record and administration
449 highScore = localStorage.getItem(localStorageName) == null ? 0 :
    localStorage.getItem(localStorageName);
450
451 //creates our Phaser III Game configurations.
452 var config = {
453     //dozens of configurations parameters available.
454     type: Phaser.AUTO,  // .WEBGL or .Canvas
455     width:    500,      // x width - best results use Golden Ratio
456     height:   312,      // y width - best results use Golden Ratio
457
458     title: 'Phaser3 Game Prototyping Starter Kit',      //Game Title
459     url: 'http://makingbrowsergames.com/p3gp-book/',    //Game URL location
460     version: '0.0.1.0 semver ', //http://semver.org/spec/v1.0.0.html
461     input: {
462         keyboard: true,
463         mouse: true,
464         touch: true,
465         gamepad: false
466     },
467     backgroundColor: 0, //Custom RGB color. (0x000 - 0xFFF)
468     physics: {
469         default: 'arcade'
470     },
471     scene:  playGame,
472     //parent: document.body,
473     parent:   "game-area",
474     pixelArt: false,
475     antialias: true
476 };
477 /**
478  * Standard global pollution in Phaser v3.x.x preparations
479  * Configuration as an object literal: NOT HOISTED
480 //Using function method; JS Hoisted!
481 var config = function config() {
482         var width = window.GAMEAPP.viewportWidth,
483         height = window.GAMEAPP.viewportHeight,
```

```
484             type = Phaser.AUTO,
485             parent = gameCanvas;
486         };
487     */
488     console.log("Configure Obj: === Ext? "+Object.isExtensible( config ));
489     //console.log(Object.values(config));
490     //console.log(Object.getPrototypeOf(config));
491     console.log(Object.getOwnPropertyDescriptors(config));
492
493     /**
494      *  DEPRECATED METHOD - NEVER EVER USE THIS AGAIN!
495      *  See "Phaser III Game Design Workbook" for complete explanation
496      *  http://leanpub.com/phaser3gamedesignworkbook
497      *
498      *  window.onload = function () {
499      *      let game =  . . . . . .
500      *  };
501     */
502
503     var game = {};
504     //preferred game launch method.
505     document.addEventListener('DOMContentLoaded', function(){
506         //This is a different launch style than Phaser v2.x.x
507         // Phaser III accepts a function or object
508         game = new Phaser.Game(config);
509     }, false);
510
511     console.log("Game prototype (Phaser.Game): === Ext? "+Object.isExtensible(
         Phaser ));
512     //console.log(Object.values(Phaser));
513     //console.log(Object.getPrototypeOf(Phaser));
514     console.log(Object.getOwnPropertyDescriptors(Phaser));
515     //
516     // ================================================
517     /** End of file */
518     /** Location: ./p3game.js */
519
```