aut
disce
aut
discede

James Woods Regional High School

Year 11 IPT Product (Minor Project)	Student name:	Jane O'Smith
	Teacher:	Mr Jones
	Start Date:	27/07/15
	Draft Due:	14/08/15
	Due Date:	31/08/15

Task

Design, develop and evaluate a web based browser game using HTML, PHP and (optionally) CSS, such as hangman, cards, a board game, etc.

Purpose

To demonstrate your application of knowledge in designing, developing and evaluating and algorithm and program.

Length

800 words to 1000 words (not including coded solutions)

Conditions

In class work (some time outside class will need to be devoted to completing the assignment).

Documentation	Design	**Identification** 150 words	**Problem Identification:** What are the requirements of the project? **Rationale:** Why is the project being undertaken?
		Specification 150 words	**Objectives:** What are the general objectives of the game? **Audience:** Who is the intended audience of the game?
		Product Design 50 words	**Flow Chart / Pseudocode:** What is the overall plan for the game? **Appearance:** What will the game look like?
	Develop	**Implementation** Code, screenshots	**File Creation:** Create pictures and other files. Set up directory structure. **Code:** Write the code for the game. *Code must be included in the documentation and should conform to consistent and appropriate style.* **Publish:** Upload the game so it is accessible and playable on the web.
		Testing* 200 words	**Testing:** Test the game for accuracy and attempt to correct errors. Make a note of each error and how it was resolved.
	Evaluate	**Evaluation*** 350 words	**Product Evaluation:** Judge the qualities of the game, and have a peer review the game based on the principles of user friendliness. **Process Evaluation:** Appraise the process undertaken to design, develop and evaluate the game.

*** Testing and evaluation should be ongoing and not done separately at the end of the project.**

Grade		A	B	C	D	E
	Knowledge and Application	**detailed** description and explanation of **links between** sequence, selection, iteration, functions and variables	description and **explanation** of sequence, selection, iteration, functions and variables	**description** of sequence, selection, iteration, functions and variables	**elements** of algorithm and software programming facts	reproduction of isolated algorithm and software programming facts
		detailed and effective application of sequence, single branch selection and simple iteration to create a working game	**effective** application of sequence, single branch selection and simple iteration to create a working game	**application** of sequence, single branch selection and simple iteration to create a working game	use of sequence, single branch selection or simple iteration to **create a partially working game**	elements of sequence, single branch selection or simple iteration used to create elements of a game
	Analysis and Synthesis	**detailed** interpretation and analysis of problems and situations **from multiple perspectives** using step-wise design	**interpretation** and analysis of problems and situations	**analysis** of problems **and** situations	**identification and classification** of problems or situations	restated problems or situations
		designed and developed an **effective** game using multiple selection, nested iteration, functions, variables and outputs	designed and developed a game using multiple selection, nested iteration, functions, variables and outputs	designed and developed a **partial** game using multiple selection, nested iteration, functions, variables or outputs	designed **or** developed **elements of** a game using multiple selection, nested iteration, functions, variables or outputs	superficial elements of multiple selection, nested iteration, functions, variables or outputs used
	Evaluation and Communication	**comprehensive** testing of processes and solutions, application of **self-determined** and prescribed criteria, reasoning and evidence to draw conclusions and make supported recommendations	**reliable** testing of processes **and** solutions, application of prescribed criteria, reasoning **and** evidence to draw conclusions and make **supported** recommendations	testing of processes or solutions, **application of prescribed criteria, reasoning or evidence to draw conclusions and make recommendations**	elements of testing **of processes or solutions to draw inferences**	elements of testing
		comprehensive construction of documentation and **fluent** presentation of information using suitable communication conventions to convey meaning appropriate to the context	**effective** construction of documentation and **effective** presentation of information using **suitable** communication conventions to convey meaning **appropriate to the context**	**construction of documentation** and presentation of information using communication conventions **to convey meaning**	presentation of information **using elements of communication conventions**	presentation of information

Teacher feedback:

Identification

Problem Identification

The requirements of this project are to design, develop and evaluate a web based browser game. According to Wikipedia, a browser game is “a computer game that is played over the Internet using a web browser, created and run using standard web technologies (such as HTML) or browser plug-ins (such as Flash). Browser games include all video game genres and can be single-player or multiplayer. Browser games are also portable and can be played on multiple different devices or web browsers”. For this project, the game will be created using HTML (including CSS) and PHP, and will be single player only.

Rationale

This project is being undertaken to put web based programming theory into practice, and also to create a product that could be used in the real world. The popularity of simple web based games and apps is growing, so the skills learnt and demonstrated in this project have real life applications.

Specification

Objectives

The game will be a simple “guess the number” game.

- A random number between 1 and 100 will be generated.
- The user will attempt to guess the number, and they will be told if their guess is too high or too low.
- Once the user had correctly guessed the number, the number of guesses they took will be displayed.
- The objective of the game is to use up the fewest number of guesses.

Audience

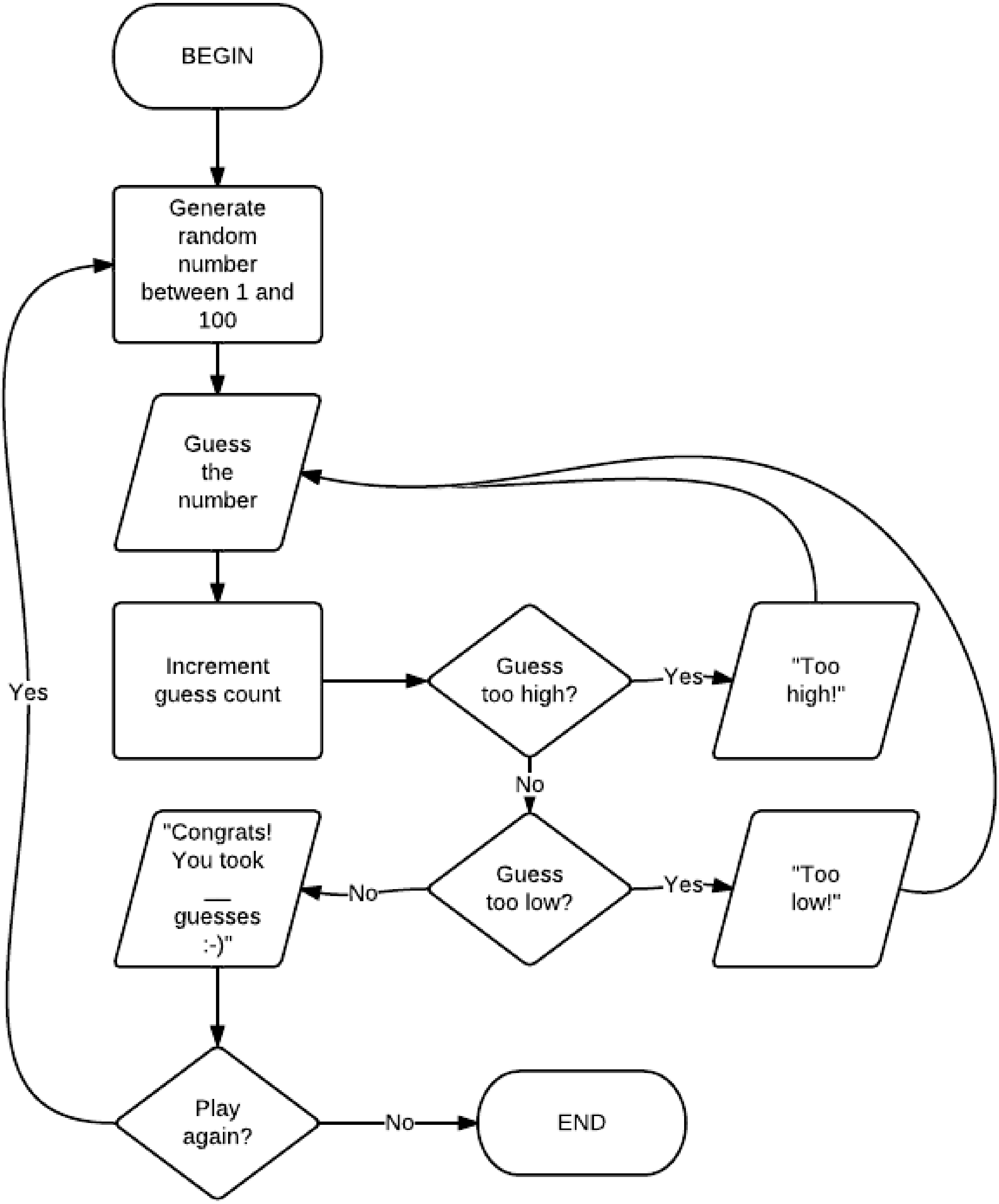
The audience for this game is children learning about basic numbers and comparing numbers to determine which is bigger and which is smaller. In Queensland, this is taught in Prep and Grade 1, so this game would be a useful tool in a primary school. The design and language used will be aimed at young children.

Product Design

Pseudocode

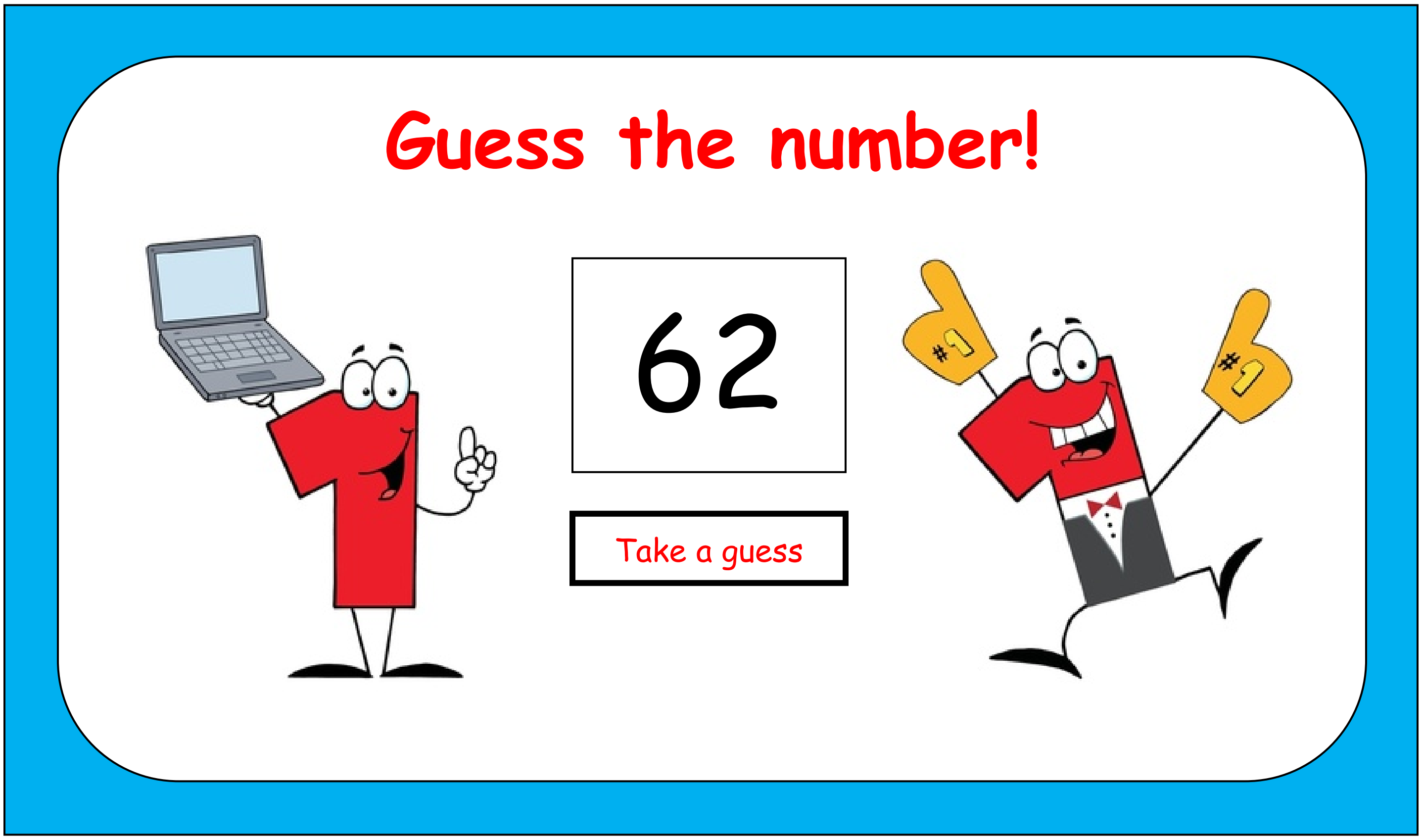
```
repeat
  target = random number (integer) between 1 and 100
  reset number of guesses
  get the guess from the user
  increment number of guesses
  if the guess is too high
    tell the user the guess is too high
  else if the guess is too low
    tell the user the guess is too low
  else
  tell the user they have guessed correctly and tell them the number of guesses they took
until the user quits
```

Flowchart


BEGIN
Generate random number between 1 and 100
Guess the number
Increment guess count
Guess too high?
Yes
"Too high!"
No
Guess too low?
Yes
"Too low!"
No
"Congrats! You took __ guesses :-)"
Play again?
Yes
No
END

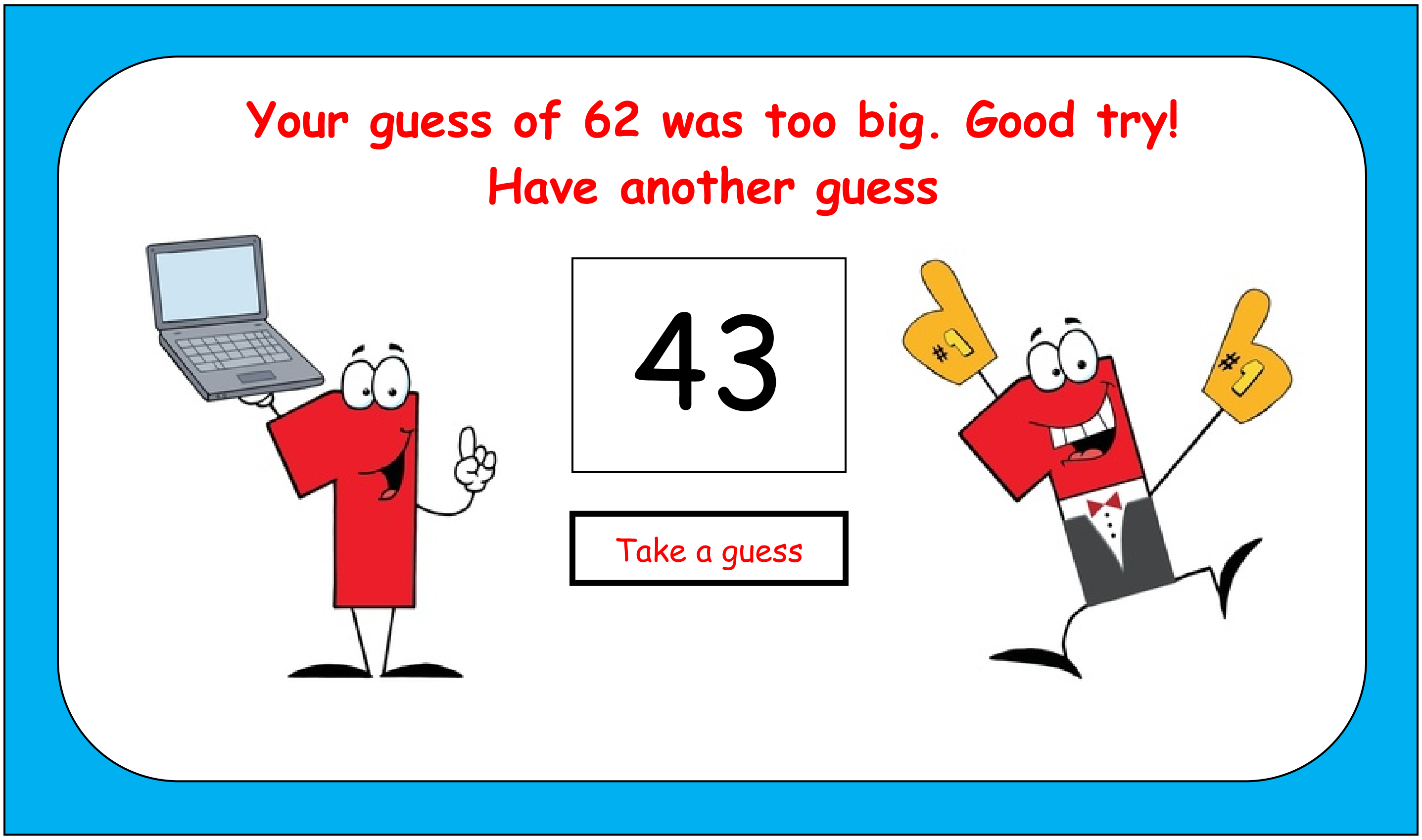
Appearance

Because the audience is young children, the entire game will take place on a single screen to avoid navigation problems. The colour and feel of the game will be colourful and fun, but simplistic. The language used will be very basic and will encourage the student where necessary, and praise them for good behavior.



Guess the number!

62

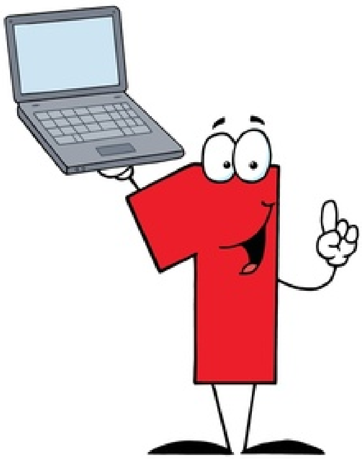


Your guess of 62 was too big. Good try! Have another guess

43

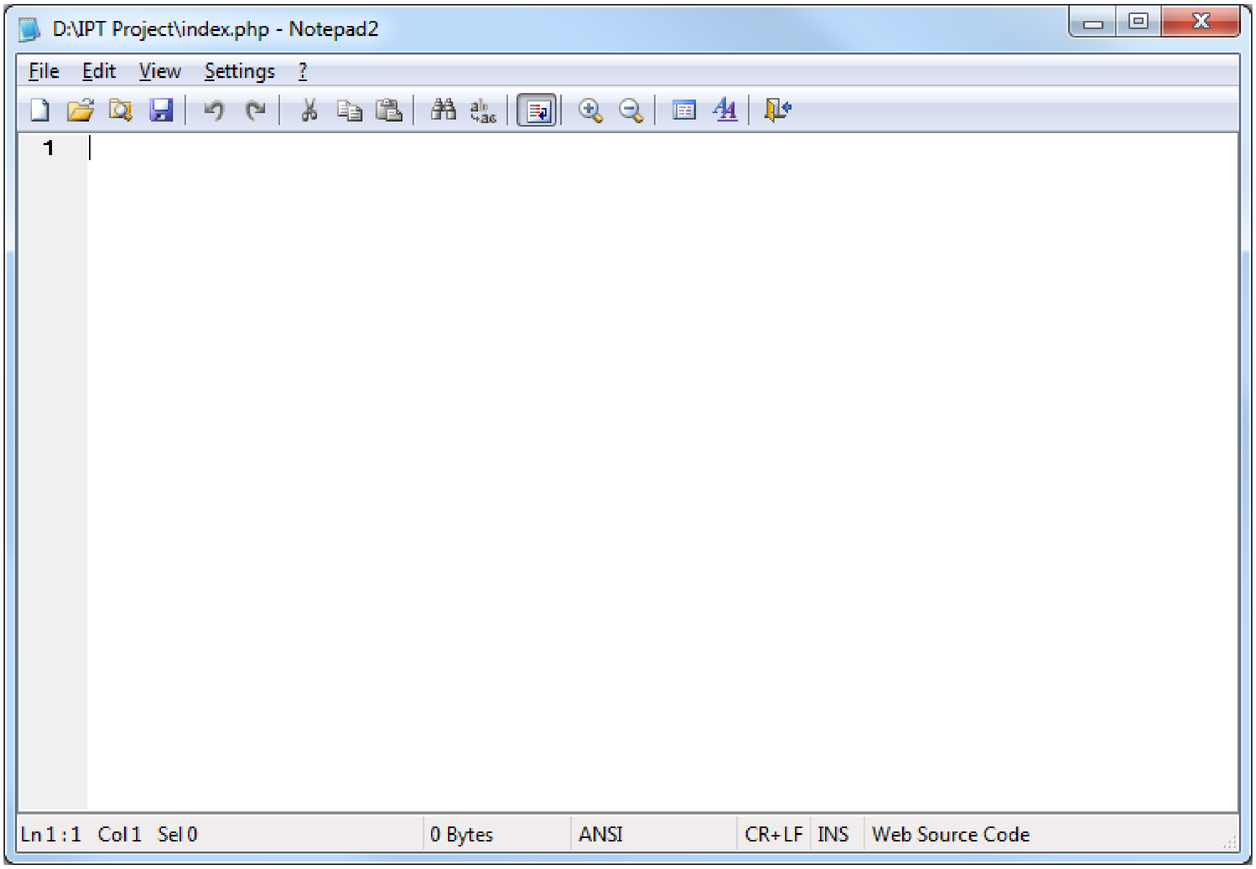
Implementation

File Creation



number1.png

number2.png

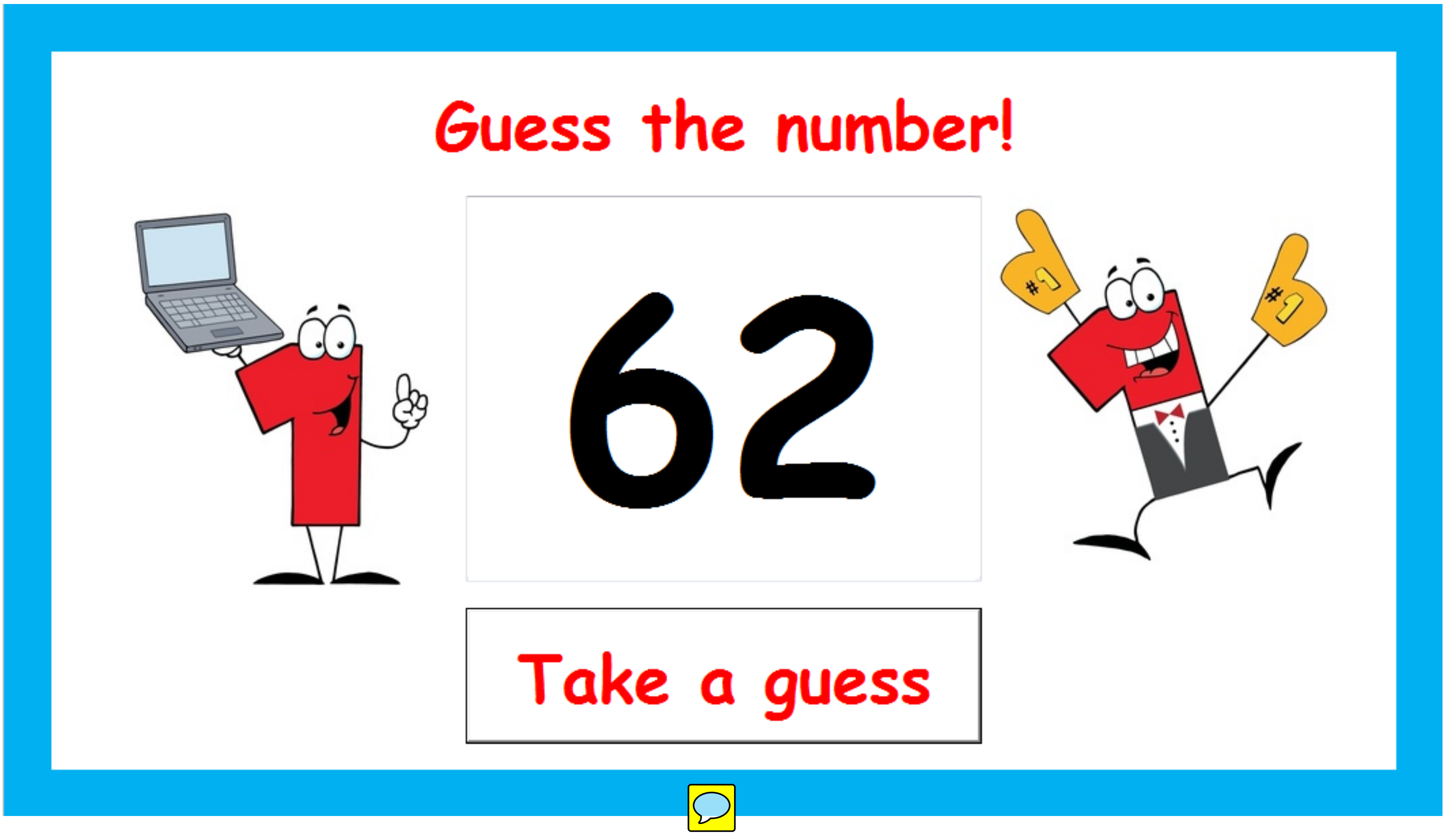


- IPT Project
  - index.php
  - images
    - number1.png
    - number2.png

Code

Initial HTML/CSS code to setup layout (the game form does not yet function):

```
<html>
   <head>
      <title>Guess the number!</title>
      <style type="text/css">
         body {
            margin: 0px;
         }
         #main {
            background-color: white;
            border-width: 35px;
            border-style: solid;
            border-color: #00B0F0;
         }
         #message {
            background-color: white;
            color: red;
            font-family: 'Comic Sans MS';
            font-size: 48;
            font-weight: bold;
            text-align: center;
         }
         #game {
            background-color: white;
            text-align: center;
         }
         #guess {
            color: black;
            font-family: 'Comic Sans MS';
            font-size: 200;
            font-weight: bold;
            width: 380px;
            text-align: center;
            vertical-align: top;
         }
         #submitbtn {
            color: red;
            font-family: 'Comic Sans MS';
            font-size: 48;
            font-weight: bold;
            width: 380px;
            text-align: center;
            background-color: white;
         }
      </style>
   </head>
   <body>
      <div id="main">
         <br><div id="message">
            Guess the number!
         </div><br>
         <div id="game">
            <form name="gameform" method="post" action="index.php">
            <img src="images/number1.png" />
            <input type="text" name="guess" id="guess" />
            <img src="images/number2.png" /><br>
            <input type="submit" name="submitbtn" id="submitbtn" value="Take a guess" />
            </form>
         </div><br>
      </div>
   </body>
</html>
```



Guess the number!

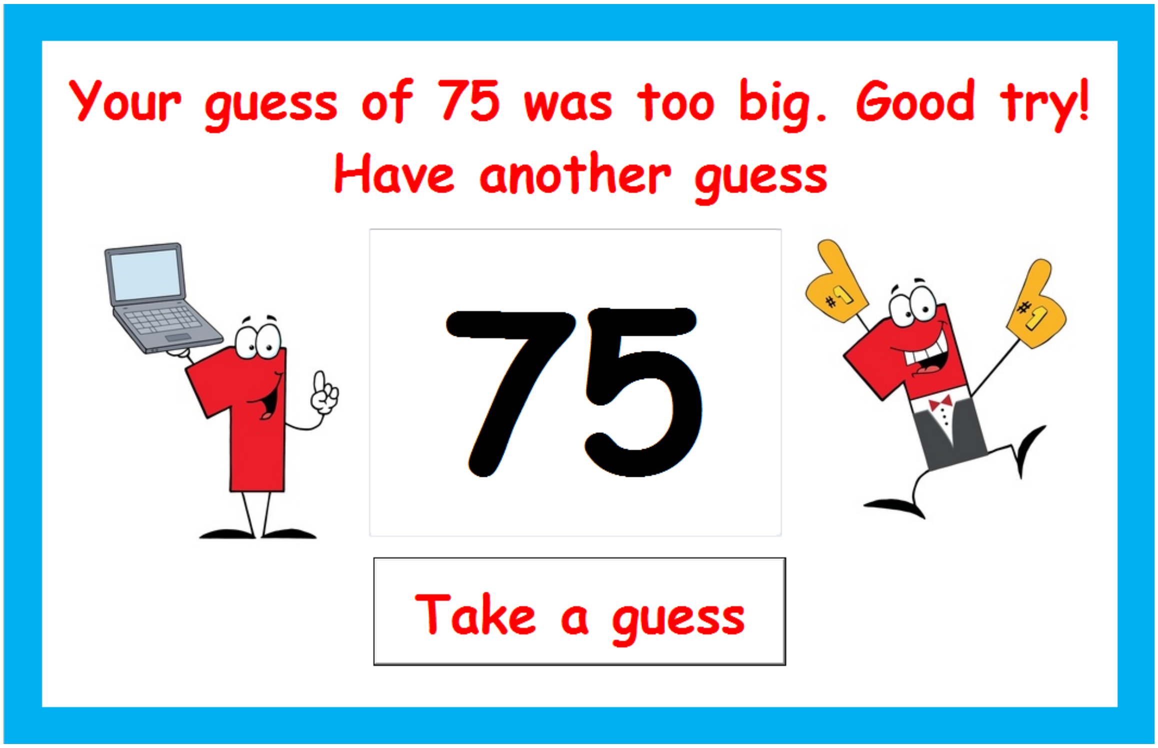
62

Addition of PHP to handle the game:

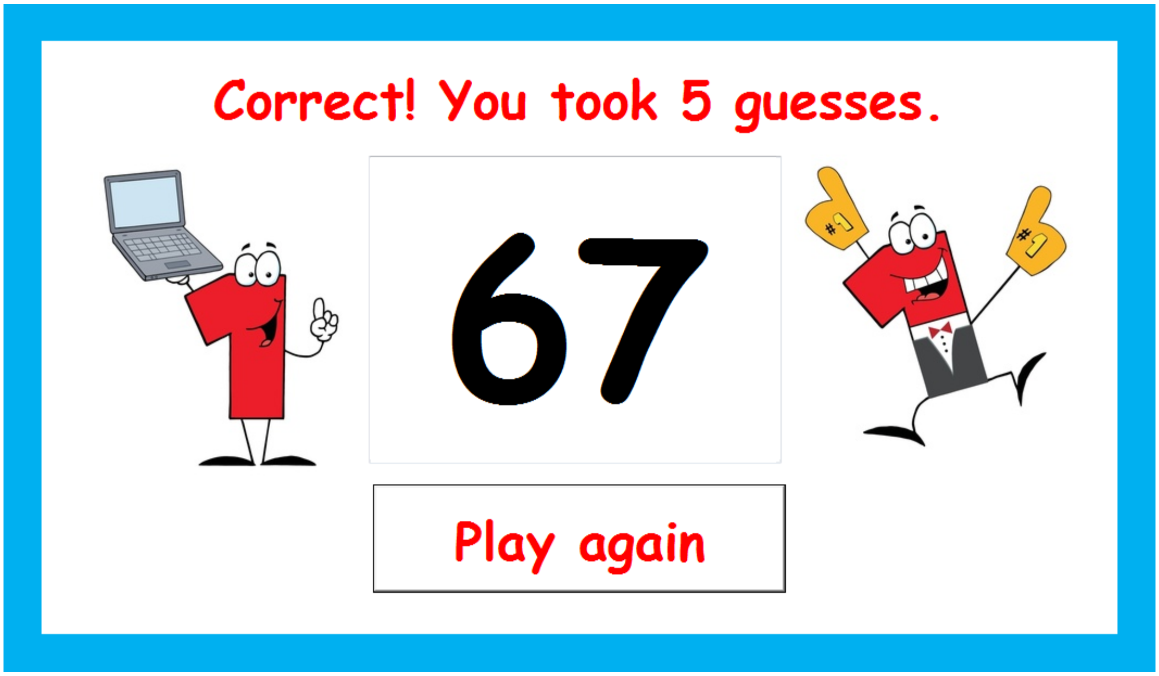
```
<?php
   $submitbtn = 'Take a guess';
   function newgame() {
      global $guess, $target, $guesses, $message, $submitbtn;
      $guess = '';
      $target = (rand()%100)+1;
      $guesses = 0;
      $message = 'Guess the number!';
      $submitbtn = 'Take a guess';
   }
   if (isset($_POST['submitbtn'])) {
      $target = $_POST["target"];
      if ($target != 'new') {
         $guess = $_POST["guess"];
         $guesses = $_POST["guesses"]+1;
         if ($guess > $target) {
            $message = 'Your guess of '.$guess.' was too big. Good try!<br> Have another guess';
         } else if ($guess < $target) {
            $message = 'Your guess of '.$guess.' was too small. Good try!<br> Have another guess';
         } else {
            $message = 'Correct! You took '.$guesses.' guesses.';
            $target = 'new';
            $submitbtn = 'Play again';
         }
      } else {
         newgame();
      }
   } else {
      newgame();
   }
?>
```

Modified code for the displayed elements (dynamic values so appropriate messages are displayed):

```
<br><div id="message">
  <?php printf("%s", $message);?>
</div><br>
<div id="game">
   <form name="gameform" method="post" action="index.php">
   <img src="images/number1.png" />
   <input type="text" name="guess" id="guess" value="<?php printf("%s", $guess);?>"/>
   <input type="hidden" name="target" id="target" value="<?php printf("%s", $target);?>" />
   <input type="hidden" name="guesses" id="guesses" value="<?php printf("%s", $guesses);?>" />
   <img src="images/number2.png" /><br>
   <input type="submit" name="submitbtn" id="submitbtn" value="<?php printf("%s", $submitbtn);?>" />
   </form>
</div><br>
```


Your guess of 75 was too big. Good try!
Have another guess

75


Correct! You took 5 guesses.

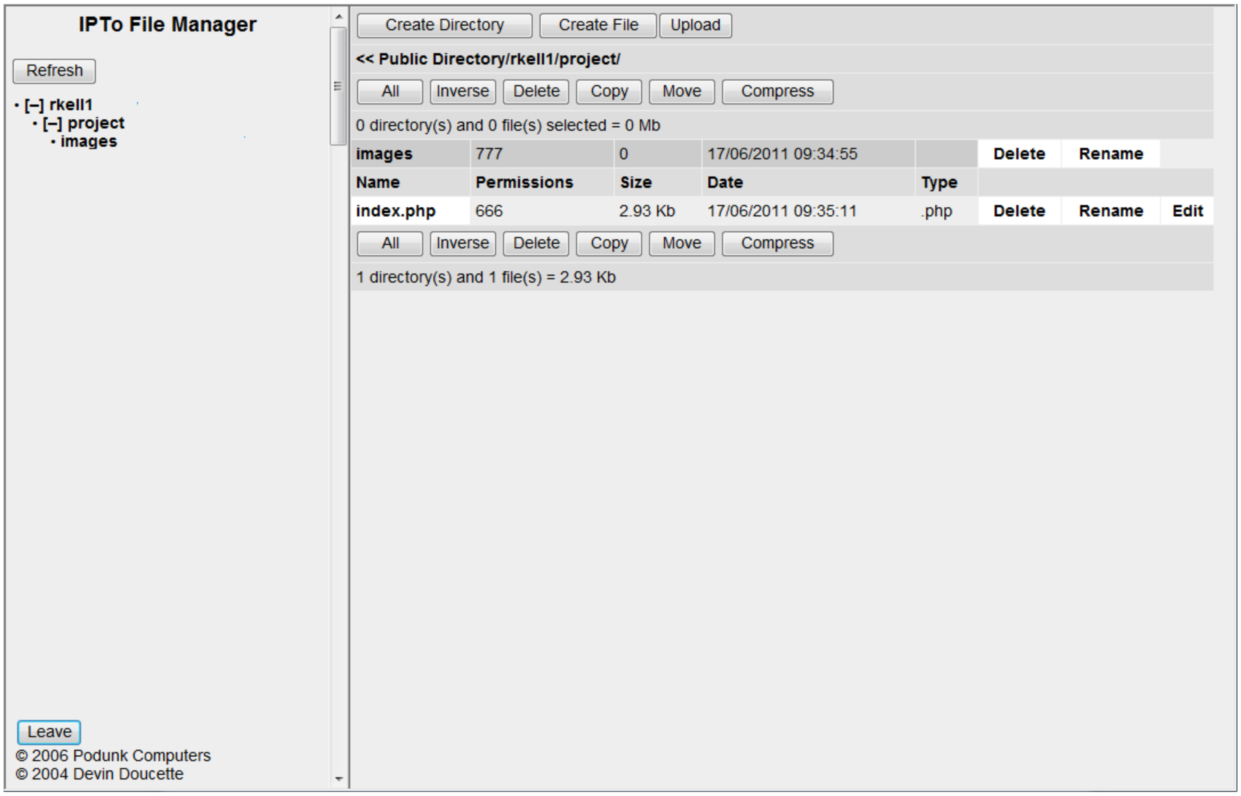
67

Completed code:

```
<?php
   $submitbtn = 'Take a guess';
   function newgame() {
      global $guess, $target, $guesses, $message, $submitbtn;
      $guess = '';
      $target = (rand()%100)+1;
      $guesses = 0;
      $message = 'Guess the number!';
      $submitbtn = 'Take a guess';
   }
   if (isset($_POST['submitbtn'])) {
      $target = $_POST["target"];
      if ($target != 'new') {
         $guess = $_POST["guess"];
         $guesses = $_POST["guesses"]+1;
         if ($guess > $target) {
            $message = 'Your guess of '.$guess.' was too big. Good try!<br> Have another guess';
         } else if ($guess < $target) {
            $message = 'Your guess of '.$guess.' was too small. Good try!<br> Have another guess';
         } else {
            $message = 'Correct! You took '.$guesses.' guesses.';
            $target = 'new';
            $submitbtn = 'Play again';
         }
      } else {
         newgame();
      }
   } else {
      newgame();
   }
?>
<html>
   <head>
      <title>Guess the number!</title>
      <style type="text/css">
         body {
            margin: 0px;
         }
         #main {
            background-color: white;
            border-width: 35px;
            border-style: solid;
            border-color: #00B0F0;
         }
         #message {
            background-color: white;
            color: red;
            font-family: 'Comic Sans MS';
            font-size: 48;
            font-weight: bold;
            text-align: center;
         }
         #game {
            background-color: white;
            text-align: center;
         }
         #guess {
            color: black;
            font-family: 'Comic Sans MS';
            font-size: 200;
            font-weight: bold;
            width: 380px;
            text-align: center;
            vertical-align: top;
         }
         #submitbtn {
            color: red;
            font-family: 'Comic Sans MS';
            font-size: 48;
            font-weight: bold;
            width: 380px;
            text-align: center;
            background-color: white;
         }
      </style>
   </head>
   <body>
      <div id="main">
         <br><div id="message">
            <?php printf("%s", $message);?>
         </div><br>
         <div id="game">
            <form name="gameform" method="post" action="index.php">
            <img src="images/number1.png" />
            <input type="text" name="guess" id="guess" value="<?php printf("%s", $guess);?>"/>
            <input type="hidden" name="target" id="target" value="<?php printf("%s", $target);?>" />
            <input type="hidden" name="guesses" id="guesses" value="<?php printf("%s", $guesses);?>" />
            <img src="images/number2.png" /><br>
            <input type="submit" name="submitbtn" id="submitbtn" value="<?php printf("%s", $submitbtn);?>" />
            </form>
         </div><br>
      </div>
   </body>
</html>
```

Publish

Files were uploaded using file manager. The game is accessible at http://www.ipto.com.au/games/guessthenumber



<< Public Directory/rkell1/project/

0 directory(s) and 0 file(s) selected = 0 Mb

images	777	0	17/06/2011 09:34:55		Delete	Rename	
Name	Permissions	Size	Date	Type			
index.php	666	2.93 Kb	17/06/2011 09:35:11	.php	Delete	Rename	Edit

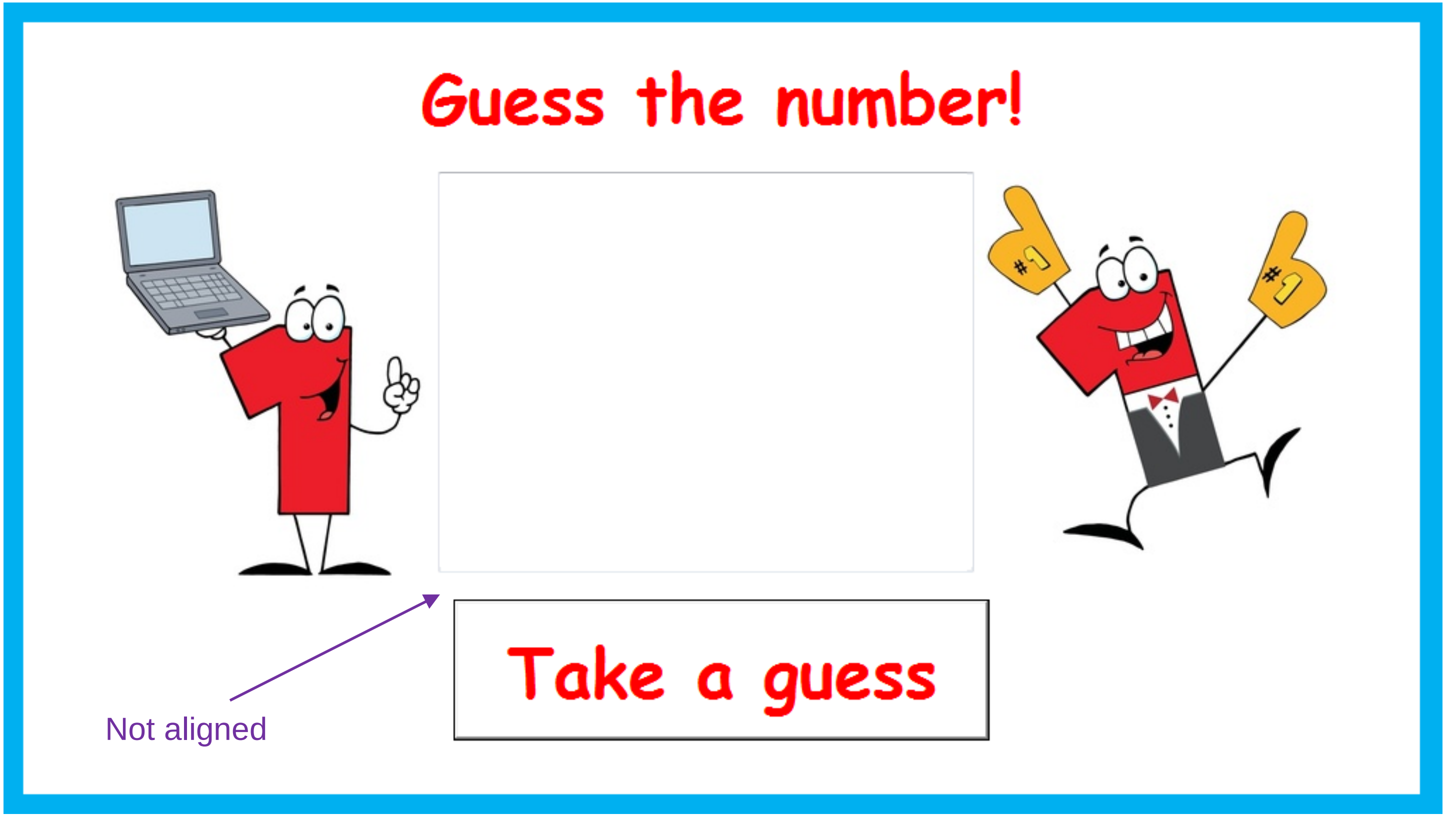
1 directory(s) and 1 file(s) = 2.93 Kb

© 2006 Podunk Computers
© 2004 Devin Doucette

Testing

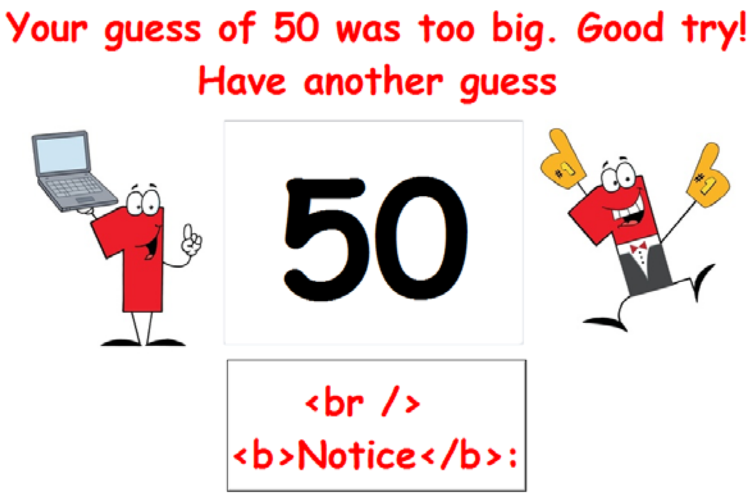
Ongoing testing during the development process revealed many errors along the way.

During the initial setup of the layout, the submit button didn't line up with the guess text box:



After a lot of modifications to the HTML / CSS, it was discovered that the error actually occurred because the two pictures of the numbers were not the same width. The picture file was edited, and the error was resolved.

This error was encountered when adding the PHP:



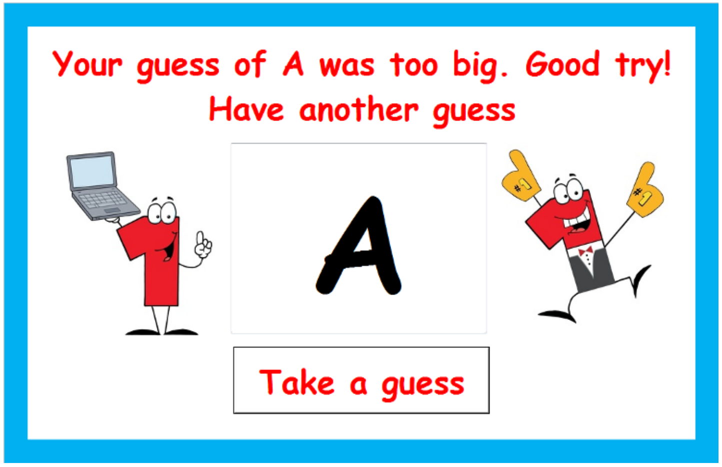
Your guess of 50 was too big. Good try!
Have another guess

```
<br />
<b>Notice</b>:
```

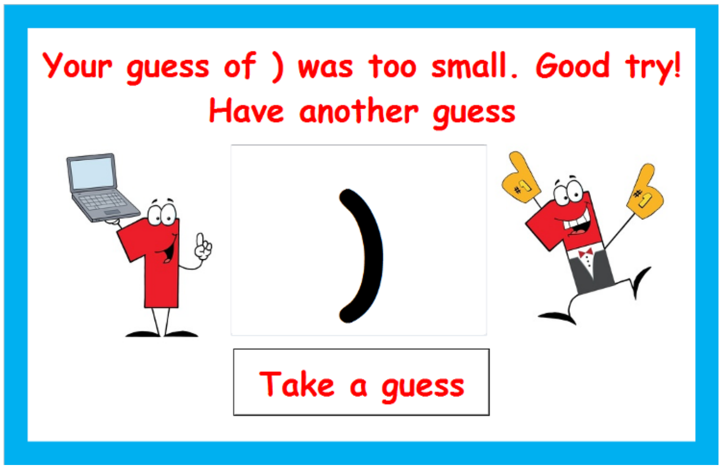
It was difficult to debug because the error message was hidden in the submit button. Investigation revealed that the $submitbtn variable was not initialized. This occurred because the variable was only set at the beginning of the game. $submitbtn = 'Take a guess'; was added to the beginning of the code to overcome this problem.

When testing the game initially, it was noticed that the numbers never seemed to be greater than 10. The code for generating the random number was $target = (rand()%10)+1; - missing a zero. This was easily fixed.

Many different non-numerical inputs were tested:


Your guess of A was too big. Good try!
Have another guess

A

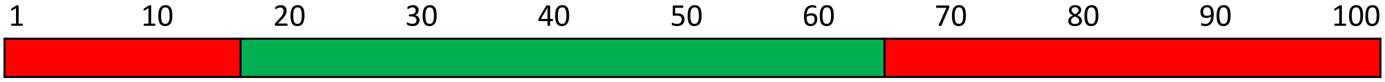

Your guess of) was too small. Good try!
Have another guess

)

None actually caused an error, but non-numerical inputs really should be excluded from entry in the first place.

Evaluation

Product Evaluation
The game achieves all the objectives set in the specification phase of the project. It will generate a random number between 1 and 100, allows the user to guess the number, and their guess is checked to see if it is too high, too low or correct. It counts the number of guesses and displays this when they guess correctly. The target audience was young children (Prep – Grade 1). The finished product caters to this audience in terms of colour, images, language and layout. To make it even easier for a child to use the game, the submit button could be triggered by pressing enter, and the contents of the guess text box could be highlighted so that typing overwrites the value (instead of having to select the value and type over the top). This would mean the game could be used with no mouse, which may suit the target audience more. Other improvements could be made to the game, such as a number line showing where the target number lies which is updated after each guess:



This could be achieved using an HTML table and filling the cells depending on the guesses already made. This would be a useful addition for older children to teach them about the number line.

Much of the code required to make the game was not known, but a few searches on Google revealed some examples of code, such as:
php random number
css div border
php check if submit button clicked
input text box style
In all four cases, the first or second hit on Google had the code that was needed.

Peer Review of User Friendliness

Is the game easy to learn?	There are no instructions for the game, and it doesn’t tell you the number you are trying to guess is between 1 and 100. After you have worked that out, the game is pretty straightforward.
Is the game easy to use?	The controls are easy if you use a mouse to select the text box before typing the guess, and for clicking the guess button. It would be good to not have to do this every time. The GUI itself is simple and looks good.
Is the game consistent and predictable?	The same keys and buttons are used each time, however after the first screen, the message up the top takes up more room, which pushes the other content down (moving the text box and button). It would be better if everything stayed where it was initially.
Is the game fault tolerant and user proof?	Anything can be typed into the text box, including letters and characters. This does not appear to break the game, but it means you have wasted a guess (and being told W is too big for example doesn’t make much sense).
Is the game error free?	The game was tested 10 times and it ran to completion every time with no errors.
Is the game intuitive?	The game is fairly intuitive, but the “Take a guess” button doesn’t actually look like a button, so it took a second to realize that was where you had to click.

Process Evaluation

During the design phase of the project, everything needed to complete the development of the game was put in place. The objectives set out were reasonable and attainable. Perhaps more ambitious features could have been planned, such as dynamic messages (instead of just “Nice try!” every time) or the number line mentioned above.

The development phase took the longest, and many of the things that needed to be included in the game (such as some of the CSS and generating a random number in PHP) required independent research on the Internet to find the correct code, and sometimes this code did not achieve the correct outcome. Some of the errors were not logged, and were forgotten by the end of the project – in future, a more detailed error log needs to be kept.

Overall, the process ran smoothly and enabled the game to be properly developed.