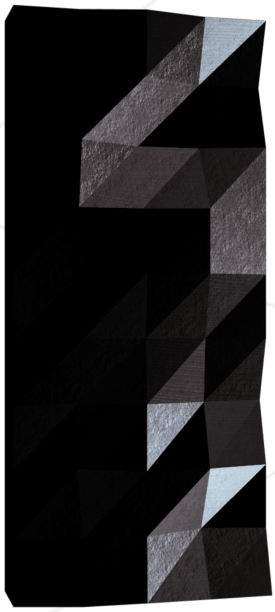


PHP Piscine

Day 00

Staff 42 piscine@42.fr

Summary:
This document is the day00's subject for the PHP Piscine.

Contents

Chapter I

Foreword

“I have the feeling these people are trying to kill us!”
“I know father!”
“It’s a new experience for me!”
“Happens all the time to me!”

Remember that when this dialogue happened,
a Piscine would not have been possible.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : Basics or home

	Exercise 00
Basics or home	
Turn-in directory : $ex00/$	
Files to turn in : `basics.html`	
Allowed functions : `n/a`	
Notes : `n/a`	

Note: this first exercise, similar to the next ones, has 2 premises. The first is that you have watched the introduction video to HTML/CSS, and the second is that you won't try to look for a pre-packaged answer for the exercises but will really do your best to learn HTML.

We will start with something simple. If you can't do it at first, take a break, go home and come back when you are feeling better.

Create a web page that contain the following elements:

- A pink background ("Pink ? what's wrong with pink ?"). Be cautious, this is not a random color choice.
- "Basics" as the navigation tab's title.
- A title -so bigger- in white, centered, that contains at least 1 special character.
- Two images side by side representing online shops.
- Under each image, a link to the matching website.
- Under that, a horizontal line that separates the page (no matter the size of the window).
- Finally, under this line, aligned on the right side of the page, in italic and with a monospace font, the copyright symbol, your login, and the current year. ("© ol 2014").

Note: this exercise should be consumed with moderation.

Chapter IV

Exercise 01 : Mendeleïev

	Exercise 01
Mendeleïev	
Turn-in directory : $ex01/$	
Files to turn in : `mendeleiev.html`	
Allowed functions : `n/a`	
Notes : `n/a`	

Create a webpage that represents Dimitri Mendeleïev complete periodic table of the elements (the current version). The constraints are as follows:

- The page must contain at least one image, but not for the table itself or for a part of it, only for embellishment.
- It needs color to be pretty.
- Each box in the periodic table contains in the middle and in BOLD the symbol of the element, and in small its number, as well as it's atomic weight, respectively in the lower left and top right corners.
- The information seen in the table must be in a modifiable text, for example during your defense.

Chapter V

Exercise 02 : Day of the 42

	Exercise 02
Day of the 42	
Turn-in directory : *ex*02/	
Files to turn in : `doft.html doft.css resources/`	
Allowed functions : `n/a`	
Notes : `n/a`	

Local foreword: “I feel like I could... like I could... like I could... TAKE ON THE WORLD!!”

Create very precisely and totally identical the page of the fictional game “Day of the 42” of which you have an image below and in the annex. To create this, you have a slew of compulsory images that you must use (no more, no less). Pay attention to all the details, the positioning, the fonts, the colors, the frames... there are also some links and hot links that will direct you to the following sites:

- The reload.png image whose title is “Start from the beginning” will bounce you to *www.disney.com*.
- close.gif (“Disconnect”) goes to *www.relaischateaux.com*.
- The actions on the left are respectively named “Advance”, “Take”, “Look”, “Use”, “Speak”. The alt/title must use that name.
- In the central image, a zone on the principal chair in the center (the second one from the right) goes to *www.ikea.com* and the biggest screen, from the back, in the lower right corner goes to *www.apple.com*.
- You can choose the object’s titles but they must have one.

Seriously, if you have never played it, take at least half a day after Piscine, it’s mandatory . Particularly since the remastered version just came out.

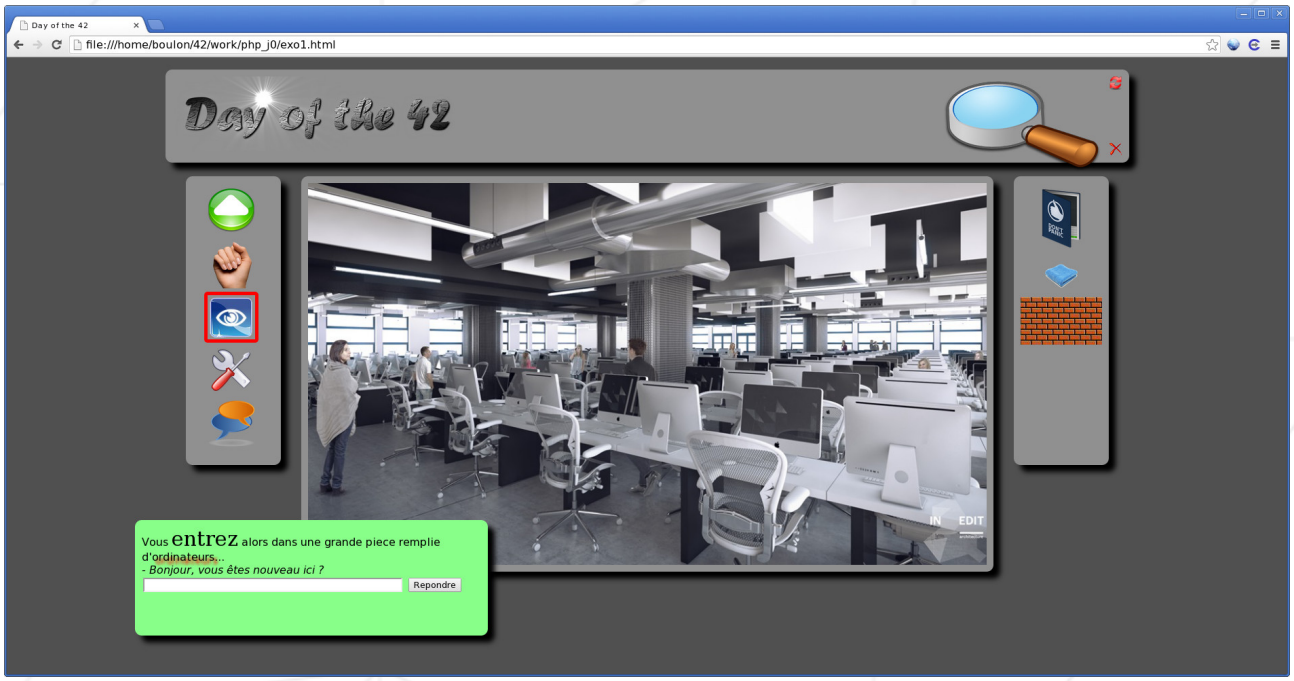


Figure V.1: “Day of the 42” page

Chapter VI

Exercise 03 : The shrunk agent

	Exercise 03
The shrunk agent	
Turn-in directory : $ex03/$	
Files to turn in : `responsive.html responsive.css`	
Allowed functions : `n/a`	
Notes : `n/a`	

Like most Russians at some paranoid time in History, Dimitri Mendeleïev was an agent. If he wasn't at the origin of the former important press outfit (TASS agency), however, he too started to shrink in time. His periodic table did as well. Copy your periodic table from the first exercise and make it responsive to the browser resizing. The periodic table must shrink, the fonts as well. Do as you wish but do not listen to the Javascript pros that will say that it can't be done without it. Stay on CSS and wait for day09 for the JS.

Chapter VII

Exercise 04 : SNCF* Sandwich

	Exercise 04
SNCF* Sandwich	
Turn-in directory : $ex04/$	
Files to turn in : `menu.html menu.css`	
Allowed functions : `n/a`	
Notes : `n/a`	

A train leaves Paris for Lyon, Saturday September 12th 1985, day of a lunar eclipse. It will ride at 216 km/h, the wind blows S/SE at a factor 5, and the temperature outside the train is 13 degree Celsius, and 22°C inside. The sky is cloudy. Another train leaves Lyon for Paris, on the same day at 23:00. It is riding at 224km/h. Which one of these two train will be the furthest away from Paris when they will cross path?

While you mull over the question, have a SNCF ham and cheese sandwich commonly referred as "Drop-down menu". You have some ingredients that are authorized, others are forbidden:

- Ingrédient CSS : OK.
- Ingrédient HTML : OK.
- Ingrédient Javascript : NON.
- Ingrédient autre : NON.

Although the aesthetic is subjective, please create a SNCF sandwich -Drop-down menu- that will be appetizing -rather nice to look at-.

*(French National Railway Company)

Chapter VIII

Exercise 05 : SCUMM

	Exercise 05
SCUMM	
Turn-in directory : $ex05/$	
Files to turn in : `scumm.html *`	
Allowed functions : `n/a`	
Notes : `n/a`	

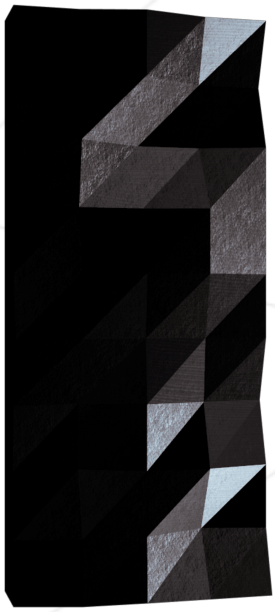
Ok, so you've done a pretty interface for your game "Day of the 42", but it would be great to be able to actually play it. Create a mini-story. Make the lot playable. Only constraint: no web server, (no Apache, no nginx or anything else), you can only be played from files. JS if you want, but you can do without with LOADS of HTML files.

The grading will take into account that we can:

- Navigate to many places while changing the central image.
- Command actions from the left banner.
- Grab objects.
- Use objets.
- Talk to characters.
- Have a description of the elements present when we look at them.
- Combine the prior points with some zones in the image: I choose an action, then I click on the image on the concerned spot.

Consider these items as options. No option, 0 to the exercise. Many options, many points.

The submission file is your entry file. It must exist. You can then add all the additional files you wish.



PHP Piscine

Day 01

Staff 42 pedago@42.fr

Summary:
This document is the day01's subject for the PHP Piscine.

Contents

Chapter I

Foreword

Catching a Porcupine

Father used to say
that if I were sitting,
waiting for a porcupine,
the time is always best
when the Milky Way turns back—
this is the time
when a porcupine returns.

Father also said
I should feel the wind.
He used to say
I should be careful
always to test
the direction of the wind.
The porcupine is not a thing
which will return, he'd say,
coming with the wind.
Rather, it moves
slant-wise, across it,
so that it can better
sniff the air and tell
if danger lurks ahead.

Father used to say
I should breathe softly
when sitting, waiting
for a porcupine.
It is a thing, he said,
which hears everything.
I must not even
make a rustling.
I must sit deadstill.

Father taught me
about the stars.
He used to say
that I should,
if sitting by a burrow,
I should watch the stars,
the places where they fell,
I should, above all,
watch them keenly,
for the places where stars fall,
he often taught,
really are the places
where porcupines can be caught.

These are translations based on the “Bleek Collection” |Xam (bushman) oral records taken down by the German linguist W.H. Bleek, and his assistant, Lucy Lloyd, in the 1870s. The “informants” listed are the |Xam people who related their poems and tales to Bleek and Lloyd. By the end of the century, the |Xam had been effectively exterminated; nobody on earth today can speak their language.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : HW

	Exercise 00
HW	
Turn-in directory : $ex00/$	
Files to turn in : `hw.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Reminder: PHP is really easy. It's like C, except that we do not declare the variables. You just place a dollar sign in front of them, they are not typed, and there in no main. The rest, is -almost- a detail.

Today, we will stay on PHP in command line. Start by creating a small program, quite simple, called hw.php. This program must greet the world with its famous message.

```
$> ./hw.php
Hello World
$>
```

Note: MMORPG Day (Massive Moulinette Online Rules PHP Geniuses)

Chapter IV

Exercise 01 : mlX

	Exercise 01
mlX	
Turn-in directory : $ex01/$	
Files to turn in : `mlx.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Since you are all super comfortable with **minilibX**, I am sorry to inform you that there are no **PHP** binding that you can use here. This exercise has nothing to do with **graphics** nor with **maths**. Nope, what you need to create, is a program that can display 1000 times the letter X, a newline, and with the constraint that it cannot go over 100 chars.

```
$> ls -la mlx.php
-rwxr-xr-x  1 boulon  users  92 Mar 12 11:54 mlx.php
$> ./mlx.php
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[TL;DP]

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
$>
```

Chapter V

Exercise 02 : to the Divine

	Exercise 02
to the Divine	
Turn-in directory : $ex02/$	
Files to turn in : `oddeven.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

As the Wise Old Man used to say, it's thanks to Olympia, the detergent of the Gods that the laundry is so soft and smells so good. But if you think about it, there was only 1 chance out of 2 to wash the right pile of laundry with it. It depended on whether it had an even or an odd number. Create a program in php that will kindly ask of you a pile number, and that will inform you if it's even (therefore washed with Olympia) or if it's odd.

```
$> ./oddeven.php
Enter a number: 42
The number 42 is even
Enter a number: 0
The number 0 is even
Enter a number:
'' is not a number
Enter a number: toto
'toto' is not a number
Enter a number: 21
The number 21 is odd
Enter a number: 99cosmos
'99cosmos' is not a number
Enter a number: ^D
$>
```

Pay attention to the example, in particular spaces, the uppercases and the exact messages. At the end, it a 'CTRL-D' to exit. And the readline library is not a part of the standard PHP library.

Chapter VI

Exercise 03 : ft_split

	Exercise 03
ft_split	
Turn-in directory : $ex03/$	
Files to turn in : `ft_split.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Create the ft_split function. It will take a string as argument, and will return a sorted array with the different words, initially separated by one or more spaces from the original string. Your ft_split.php submited will be included in a php test file.

```php
<?PHP

include("ft_split.php");

print_r(ft_split("Hello    World AAA"));
?>
```

```
$> ./main.php
Array
(
    [0] => AAA
    [1] => Hello
    [2] => World
)
$>
```

Chapter VII

Exercise 04 : aff_param

	Exercise 04
aff_param	
Turn-in directory : $ex04/$	
Files to turn in : `aff_param.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Very basic, this program displays its command line arguments in the order received. The name of the program isn't displayed.

```
$> ./aff_param.php
$> ./aff_param.php toto ahah foo bar quax
toto
ahah
foo
bar
quax
$>
```

Chapter VIII

Exercise 05 : epur_str

	Exercise 05
epur_str	
Turn-in directory : $ex05/$	
Files to turn in : `epur_str.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This program takes one unique argument and reduces to a single space between each word, and none at the beginning and at the end of the string. There are only spaces, no tabulation or anything.

```
$> ./epur_str.php
$> ./epur_str.php "Hello,     how do  you   do     ?"
Hello, how do you do ?
$> ./epur_str.php "   Hello World    "
Hello World
$>
```

Chapter IX

Exercise 06 : ssap

	Exercise 06
ssap	
Turn-in directory : $ex06/$	
Files to turn in : `ssap.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Do not confuse it with the enterprise management software SAP, it is for you the chance to mix the prior two exercises. The sum of words contained in all the arguments (except the name of the program itself) are splited, sorted and displayed.

```
$> ./ssap.php
$> ./ssap.php foo bar
bar
foo
$> ./ssap.php foo bar "yo man" "Here is my, two words" Xibul
Here
Xibul
bar
foo
is
man
my,
two
words
yo
$>
```

Chapter X

Exercise 07 : rostring

	Exercise 07
rostring	
Turn-in directory : $ex07/$	
Files to turn in : `rostring.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Your program will take a string as argument, and will place the first word (space separated) at the last spot. The whole thing is then re- presented, with 1 space only between each word.

```
$> ./rostring.php
$> ./rostring.php sdfkjsdkl sdkjfskljdf
sdfkjsdkl
$> ./rostring.php "hello world  aaa" fslkdjf
world aaa hello
$>
```

Chapter XI

Exercise 08 : ft_is_sort

	Exercise 08
ft_is_sort	
Turn-in directory : *ex*08/	
Files to turn in : `ft_is_sort.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

You need to create a little function that will reply true or false according to whether the array passed as argument is sorted or not.

```php
<?PHP

include("ft_is_sort.php");

$tab = array("!/@#;^", "42", "Hello World", "hi", "zZzZzZz");
$tab[] = "What are we doing now ?";

if (ft_is_sort($tab))
    echo "The array is sorted\n";
else
    echo "The array is not sorted\n";
?>
```

```
$> ./main.php
The array is not sorted
$>
```

Chapter XII

Exercise 09 : ssap - the return -

	Exercise 09
ssap - the return -	
Turn-in directory : $ex09/$	
Files to turn in : `ssap2.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Get back to your ssap.php. You need to do the same thing again (take all the words from all the parameters and sort them) but you need to change the sorting rule: now it will need to be case insensitive and place all the characters in alphabetical order first, then numbers, and finally all the other characters, each in the following 3 groups following the ASCII order.

```
$> ./ssap2.php
$> ./ssap2.php toto tutu 4234 "_hop A2l+ XXX" ## "1948372 AhAhAh"
AhAhAh
A2l+
toto
tutu
XXX
1948372
4234
##
_hop
$>
```

Chapter XIII

Exercise 10 : do_op

	Exercise 10
do_op	
Turn-in directory : $ex10/$	
Files to turn in : `do_op.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This PHP program will take 3 arguments. The second is an arithmetic operation amongst : '+', '-', '*', '/', '%'. The first and the third are numbers. You need to make this operation and display the result. The program doesn't manage errors, except the number of arguments given. Spaces and tabulations can be presented in all 3 arguments.

```
$> ./do_op.php
Incorrect Parameters
$> ./do_op.php 1 + 3
4
$> ./do_op.php " 1" " +" " 3"
4
$> ./do_op.php " 1" " *" " 3"
3
$> ./do_op.php 42 "% " 3
0
```

Note: respect the error message.

Chapter XIV

Exercise 11 : do_op_2

	Exercise 11
do_op_2	
Turn-in directory : $ex11/$	
Files to turn in : `do_op_2.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This time, only 1 argument is on the menu. That one contains the whole calculation that needs to be done. This calculation will always be under the following format *number operator number*. A new error message "Syntax Error" will now complete the prior message in case the syntax isn't correct. There can be no space between the numbers and the operator, or there can be many. The expected results is the same.

```
$> ./do_op_2.php
Incorrect Parameters
$> ./do_op_2.php toto
Syntax Error
$> ./do_op_2.php "42*2"
84
$> ./do_op_2.php "  42 / 2 "
21
$> ./do_op_2.php "six6*7sept"
Syntax Error
$> ./do_op_2.php '`rm -rf ~/`;'
Syntax Error
```

Chapter XV

Exercise 12 : search_it!

	Exercise 12
search_it!	
Turn-in directory : $ex12/$	
Files to turn in : `search_it!.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Your goal is to create a program that will display the corresponding value to a key given as first argument amongst an unlimited number of couples formated like this: “key:value” given as following arguments.

```
$> ./search_it!.php
$> ./search_it!.php toto
$> ./search_it!.php toto "key1:val1" "key2:val2" "toto:42"
42
$> ./search_it!.php toto "toto:val1" "key2:val2" "toto:42"
42
$> ./search_it!.php "toto" "key1:val1" "key2:val2" "0:hop"
$> ./search_it!.php "0" "key1:val1" "key2:val2" "0:hop"
hop
$>
```

Chapter XVI

Exercise 13 : The hesitating agent

	Exercise 13
The hesitating agent	
Turn-in directory : $ex13/$	
Files to turn in : `agent_stats.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

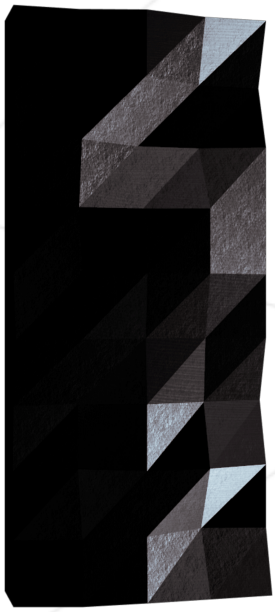
After yesterday's agent (the one that shrank), here is the one that hesitates and needs to make choices. Among the resources of the day, you have a few files of peer-notes. The idea is for you to choose precisely how to use them. Here are a few solutions:

- the "average" option will calculate the average grade without Moulinette.
- the "average_user" option will calculate the average grade per user ordered by alphabetical order.
- the "moulinette_variance" will calculate the average grade per user of the difference between a grade received by your peer and by Moulinette.

The grade file will be read on the standard input, and the option is passed as an argument to the program. In cases 2 and 3, if a user has no grade, it's not displayed.

```
$> cat peer_notes_1.csv | ./agent_stats.php
$> cat peer_notes_1.csv | ./agent_stats.php moyenne
9.8621262458472
$> cat peer_notes_1.csv | ./agent_stats.php moyenne_user
adam_e:9.0555555555556
bertrand_y:7.9473684210526
bruce_w:9.0434782608696
clark_k:10.464285714286
david_a:8.68
dexter_m:8.9

    [.....]

sandra_n:11.181818181818
steve_j:11.5
trevor_r:6.1052631578947
$>
```

```
$> cat peer_notes_1.csv | ./agent_stats.php ecart_moulinette
adam_e:3.0555555555556
bertrand_y:-1.0526315789474
bruce_w:-9.9565217391304
clark_k:0.46428571428571

    [.....]

steve_j:10.5
trevor_r:-12.894736842105
$>
```



PHP Piscine

Day 02

Staff 42 piscine@42.fr

Summary:
This document is the day02's subject for the PHP Piscine.

Contents

Chapter I

Foreword

“I realized the moment I fell into the fissure, that the book would not be destroyed as I had planned. It continued falling into that starry expanse, of which I had only a fleeting glimpse. I have tried to speculate where it might have landed, and I must admit however, such conjecture is futile. Still, questions about whose hands might one day hold my Book are unsettling to me. I know my apprehensions might never be allayed... and so I close, realizing that perhaps the ending has not yet been written.”

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : Another World

	Exercise 00
Another World	
Turn-in directory : $ex00/$	
Files to turn in : `another_world.php`	
Allowed functions : `The whole standard PHP library especially regexp`	
Notes : `n/a`	

I do not know where I landed. I just woke up, I'm lying in the grass and I see the sky above the treetops. I'm lost. The Book is near me. It could have been worse; I could find myself leaning on my desk appearing suddenly in the middle of a gigantic pool inhabited by a kraken. Fortunately this is not the case. I began to explore around. This world is different. The relation with the surrounding space is very strange. Sometimes we make a single step, but in reality we made of eight. I started to lose the habit of slowing down. Everywhere there were varied combinations of normal space and strange spaces, I decided to set a single normal space. The book seems to dedicate a whole chapter, using the terms ' ' and '\t', to respectively designate natural spaces and strange spaces.

```
$> ./another_world.php "This    sentence   contains spaces and      some      tabulations"
This sentence contains spaces and some tabulations
$> ./another_world.php
$> ./another_world.php "  This arg   is     used  " "but not this   one"
This arg is used
$>
```

Note: I have found a kind of rocky promontory. I'm on an island! I d' not know how I got there or how I'll get out of it. But tomorrow is another day.

Chapter IV

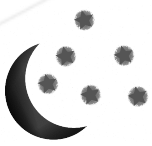
Exercise 01 : One more time

	Exercise 01
Autre Temps	
Turn-in directory : *ex*01/	
Files to turn in : `one_more_time.php`	
Allowed functions : `The whole standard PHP library and lot of regexp`	
Notes : `n/a`	

A new day, still stuck on this damn island. Besides, what day are we exactly? Time does not have the same meaning here obviously. I notice, flipping the book that some inscriptions could correspond to dates. I will try to decipher them. Midday: this is it! I found it. These inscriptions are systematically of the following form:
Day_of_the_week Number_of_day Month Year Hours:Minutes:Seconds
The day of the week is complete, full, sometimes with a uppercase at the beginning, and in French. The day's number is always 1 or 2 characters. The month is expressed in words, sometimes also with a capital letter at the beginning, and always in French. The year consists of 4 digits. Finally, hours, minutes and seconds are all of 2 figures. There are 4 spaces to separate 5 pieces. I will now be able to identify more easily if some passages correspond to dates, and convert them to a more rational mode and more readable for me: the number of seconds past since January 1, 1970.

```
$> ./one_more_time.php "Mardi 12 Novembre 2013 12:02:21"
1384254141
$> ./one_more_time.php "Mercreday 1stJuily 99"
Wrong Format
$> ./one_more_time.php
$>
```

Note: I have never encountered well formatted dates that are incoherent.



Chapter V

Exercice 02 : The magnifying glass

	Exercise 02
The magnifying glass	
Turn-in directory : *ex*02/	
Files to turn in : `magnifying_glass.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This morning, examining the binding of the book, I noticed a small recess. Inside I gently pulled out a small magnifying glass. She was really very special. At first I thought she did not grow. But by trying it on several pages, I noticed that it only grew links to other pages of the Book, by simply putting the upper case! I had to see this process each page one by one.

```
$> cat > page.html
<html><head><title>Nice page</title></head>
<body>Hello World <a href=http://cyan.com title="a link">This is a link</a>
<br /><a href=http://www.riven.com> And this too <img src=wrong.image title="And also this"></a>
</body></html>
^D
$> ./loupe.php page.html > new_page.html
$> cat new_page.html
<html><head><title>Nice page</title></head>
<body>Hello World <a href=http://cyan.com title="A LINK">THIS IS A LINK</a>
<br /><a href=http://www.riven.com> AND THIS TOO <img src=wrong.image title="AND ALSO THIS"></a>
</body></html>
$>
```

Chapter VI

Exercise 03 : Who are you ?

	Exercise 03
Who are you ?	
Turn-in directory : *ex*03/	
Files to turn in : `who.php`	
Allowed functions : `No use of system call 'who'`	
Notes : `n/a`	

I am not alone! There are people on this island! I got proof this morning following a footprint in the forest area. I found myself nose to nose with a strange individual, who seemed also amazed by this meeting like me. I asked him who he was, but I only got snippets of words like "utmp" and other elusive grunts. Afraid, he left quickly. Perhaps it will take time but soon, I will know where we are and why I'm here. I noted in the Book, each of my meetings.

```
$> ./who.php
boulon   console    Mar 25 09:08
boulon   ttys001    Mar 25 10:24
boulon   ttys002    Mar 25 10:48
boulon   ttys003    Mar 25 16:30
$> who
boulon   console    Mar 25 09:08
boulon   ttys001    Mar 25 10:24
boulon   ttys002    Mar 25 10:48
boulon   ttys003    Mar 25 16:30
$>
```

Note: I think it's me. I mean, I mean, I think the person I met was me. But in a different temporal and spatial dimension. What a disappointment! I am beginning to lose all hope to leave this place.

Chapter VII

Exercise 04 : Photo booklet

	Exercise 04
Photo booklet	
Turn-in directory : $ex04/$	
Files to turn in : `photos.php`	
Allowed functions : `The whole standard PHP library and curl`	
Notes : `n/a`	

I tried to make some drawings on the blank pages at the end of the Book but it was useless. Only the words could remain, my sketches disappeared in less time than it took me to draw them. That's when I started to dedicate a page to a precise description of a particular location on the island, that, the unpredictable happens. Soon as I wrote the place name as title, that, a whole set of pictures of the place came to fill the page. This book takes pictures! I was able to list all interests of the points of my new home. This book is decidedly surprising!

```
$> ls
photos.php
$> ./photos.php "http://www.42.fr"
$> ls
photos.php
www.42.fr/
$> ls www.42.fr/
logo42-site.gif
$>
```

Note: I realize nevertheless that the pictures are not taken at random. It's only when there is, in a place, a form of tag characterized by 3 letters I, M and G, that the picture is taken.

Chapter VIII

Exercise 05 : In the D(e)nial

	Exercise 05
In the D(e)nial	
Turn-in directory : $ex05/$	
Files to turn in : `denial.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

I can't take it anymore. I'm dreaming since too long. I have to wake up. This island can only exist in my imagination. A so strange and tortured place can't be real. I don't want to open the book anymore, I don't want to do anymore puzzle. The latest: everything I wrote to remember of my real life was found mixed between the same three pages. I had to continually re-order my ideas, only to remember the name and surname of my friends. At the price of a long and laborious effort, I was able by means of a unique identifier, deceive the Book and regroup the scattered information. By writing "display the name of login XXX", the Book finds me the information sought.

```
$> ./denial.php data.csv pseudo
Enter your command: echo $name['miawallace']."\n";
Naline
Enter your command: ^D
$> ./denial.php data.csv
$> ./denial.php invalid_file login
$> ./denial.php data.csv invalid_header_key
```

```
$> ./denial.php data.csv surname
Enter your command: echo $surname['Nestor']." ".$last_name['Nestor']." is a beautiful name\n";
Nestor Derire is a beautiful name
Enter your command: echo $IP['Sarah']."\n";
10.252.33.76
Enter your command: print_r(explode(".", $IP['Xavier'])); echo "\n";
Array
(
    [0] => 172
    [1] => 20
    [2] => 45
    [3] => 200
)

Enter your command: toto titi tutu
PHP Parse error:  syntax error, unexpected T_STRING in [....]
Enter your command: ^D
$>
```

Note: Are they really are my friends? I can no longer put a face to these names. Everything vanishes, everything escapes me...

Chapter IX

Exercise 06 : The Parchment

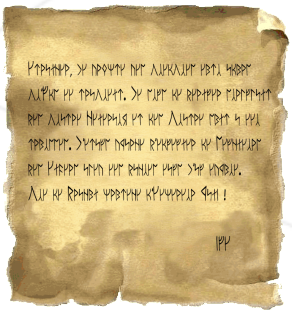
	Exercise 06
The Parchment	
Turn-in directory : $ex06/$	
Files to turn in : `srt.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

I found a sort of parchment yesterday on what I call the northern coast of the island; near a large crevasse which I cannot even see the bottom. This should rejoice me to know that there is or there have been others like me here. I'm too depressed and I really paid attention only this morning. It consists of incomprehensible text snippets interspersed with precise time stamps (I know how to recognize it now). There is however inconsistencies. Some hours are not in order. I am not sure to really want to solve this new puzzle, yet...

```
$> cat > test.srt
1
00:01:15,308 --> 00:01:16,717
This

2
00:01:21,473 --> 00:01:23,614
test

3
00:01:19,750 --> 00:01:21,373
a

4
00:01:16,817 --> 00:01:19,650
is
^D
```

```
$> ./srt.php test.srt
1
00:01:15,308 --> 00:01:16,717
This

2
00:01:16,817 --> 00:01:19,650
is

3
00:01:19,750 --> 00:01:21,373
a

4
00:01:21,473 --> 00:01:23,614
test
$>
```

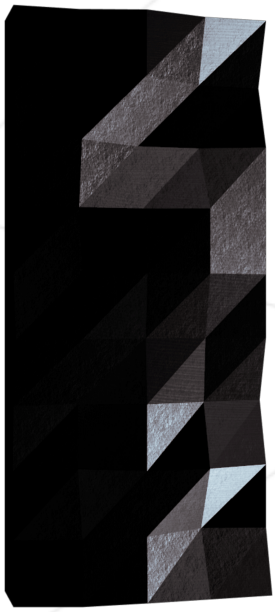
Note: I returned to the crevasse. I understand now. I decoded the parchment, everything is clear now. There is nothing else to do, either for the Book, nor me.



Chapter X

Postamble

“I realized at the same moment that was I falling into the crevasse, that the Book would not 12 destroying itself as I had planned. It continue to fall into this starry expanse, I only had a fleeting image. I tried to imagine where it could have landed; I must however admit that such questions are futile. Still, I keep wondering in whose hands my book will fall one day. I know my fears will perhaps never be appeased ... That’s why I closed, realizing that the end has perhaps not yet been written.”



PHP Piscine

Day 03

Staff 42 piscine@42.fr

Summary:
This document is the day03's subject for the PHP Piscine.

Contents

Chapter I

Foreword

Here is some of what Wikipedia has to say about Apaches:

The Apache are culturally related Native American tribes from the Southwestern United States and Northern Mexico. These indigenous peoples of North America speak Southern Athabaskan languages, which are related linguistically to Athabaskan languages in Alaska and western Canada.

Apache people traditionally have lived in Eastern Arizona, Northern Mexico (Sonora and Chihuahua), New Mexico, West Texas, and Southern Colorado. Apacheria, their collective homelands, consists of high mountains, sheltered and watered valleys, deep canyons, deserts, and the southern Great Plains. The Apache tribes fought the invading Spanish and Mexican peoples for centuries. The first Apache raids on Sonora appear to have taken place during the late 17th century. In 19th-century confrontations during the American-Indian wars, the U.S. Army found the Apache to be fierce warriors and skillful strategists.

Apache groups are politically autonomous. The major groups speak several different languages and developed distinct and competitive cultures. The current post-colonial division of Apache groups includes Western Apache, Chiricahua, Mescalero, Jicarilla, Lipan, and Plains Apache (also known as the Kiowa-Apache). Apache groups live in Oklahoma and Texas and on reservations in Arizona and New Mexico. Apache people have moved throughout the United States and elsewhere, including urban centers.

The tool you will use for your server from now on is PAMP, developed by 42. That one is still in beta, please help us make it better by bringing up the bugs you will encounter by ticket or on the forum.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00: Dat vhost !

	Exercise 00
Dat vhost !	
Turn-in directory : *ex*00/	
Files to turn in : `D03.php`	
Allowed functions :	
Notes : `n/a`	

This first exercise will make you to install and configure your web server embedded in the PAMP tool. You must accomplish this exercice only if PAMP is at least version 3.0.0, otherwise you can skip it. Therefore, you should know how to configure a vHost, this would be useful and really interesting.

Before using PHP for the web, you must configure your first `VirtualHost` such as :

- PAMP should load your site's elements from `$HOME/http/MyWebSite/d03`
- it must listen to the port 8100

Create the D03.conf configuration file that fulfill all the requirements written above. This file would be turned-in in your repository.

If you want to test your configuration file, you can add the following line at the end of the `$HOME/http/conf/my.conf` file :

```
Include <votre-dossier-de-rendu>/ex00/D03.conf
```

Now, exercises and rushes will use PAMP and its associated tools.
For more information, you can follow this link

Chapter IV

Exercise 01: phpinfo

	Exercise 01
phpinfo	
Turn-in directory : $ex01/$	
Files to turn in : `phpinfo.php`	
Allowed functions : `phpinfo()`	
Notes : `n/a`	

Create a page named phpinfo.php that will execute and show the result on phpinfo();

```
$> curl 'http://eXrXpX.42.fr:8xxx/ex01/phpinfo.php'
<!DOCTYPE html PUBLIC ``-//W3C//DTD XHTML 1.0 Transitional//EN'' ``DTD/xhtml1-transitional.dtd''>
<html xmlns=''http://www.w3.org/1999/xhtml''><head>
<style type=''text/css''>
...
<title>phpinfo()</title><meta name=''ROBOTS'' content=''NOINDEX,NOFOLLOW,NOARCHIVE'' /></head>
...
$>
```

Chapter V

Exercise 02: print_get

	Exercise 02
print_get	
Turn-in directory : *ex*02/	
Files to turn in : `print_get.php`	
Allowed functions : `echo`	
Notes : `n/a`	

Create a page named print_get.php that will display all the variables passed in the url.

Example:

```
$> curl 'http://eXrXpX.42.fr:8xxx/ex02/print_get.php?login=mmontinet'
login: mmontinet
$> curl 'http://eXrXpX.42.fr:8xxx/ex02/print_get.php?gdb=pied2biche&barry=barreamine'
gdb: pied2biche
barry: barreamine
$>
```

Chapter VI

Exercise 03: cookie_crisp

	Exercise 03
print_crisp	
Turn-in directory : *ex*03/	
Files to turn in : `cookie_crisp.php`	
Allowed functions : `echo, setcookie() et time()`	
Notes : `n/a`	

Create a page cookie_crisp.php that will allow to create, read and erase a cookie.
Example:

```
$> curl -c cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=set&name=plat&value=choucroute'
$> curl -b cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=get&name=plat'
choucroute
$> curl -c cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=del&name=plat'
$> curl -b cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=get&name=plat'
$>
```

Chapter VII

Exercise 04: raw_text

	Exercise 04
raw_text	
Turn-in directory : *ex*04/	
Files to turn in : `raw_text.php`	
Allowed functions : `header()`	
Notes : `n/a`	

Create a page named raw_text that will show the same thing on the screen if you look at its source code with curl or its html rendered in Chrome.

```
$> curl 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```

If you have lynx, you could test it like that (right down to the newline)

```
$> lynx -dump 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>

$> lynx -source 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```

Chapter VIII

Exercise 05: read_img

	Exercise 05
read_img	
Turn-in directory : *ex*05/	
Files to turn in : `read_img.php`	
Allowed functions : `header(), readfile()`	
Notes : `n/a`	

Create a page named read_img.php that will return to the browser the file 42.png with the right Content-Type. You will find this file in the attachement section on the intranet. You must submit it in your* repository in the following folder "/img/42.png" so that we can use it again in other exercises.

```
$> curl --head http://eXrXpX.42.fr:8xxx/ex05/read_img.php
HTTP/1.1 200 OK
Date: Tue, 26 Mar 2013 09:42:42 GMT
Server: Apache
X-Powered-By: PHP/5.4.26
Content-Type: image/png

$>
```

Chapter IX

Exercise 06: members_only

	Exercise 06
members_only	
Turn-in directory : *ex*06/	
Files to turn in : `members_only.php`	
Allowed functions : `header(), echo. $_SERVER, file_get_contents, base64_encode`	
Notes : `n/a`	

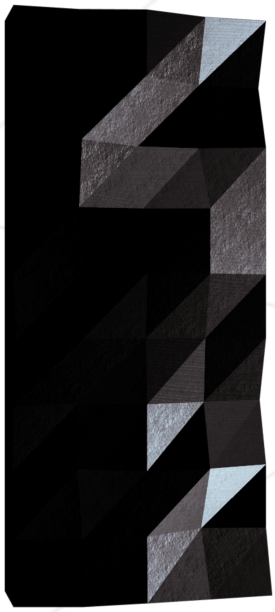
Create a page named members_only.php that will require a login/password at the http protocol level. If the login is “zaz” and the password “jaimelespetitsponeys” the answer must be an html page that contains an img tag whose source is directly the image “/img/42.png” but not its url (careful! We will probably change the content of 42.png for the correction, so no solid content value)

You need to reproduce the following example:

```
$> curl --user zaz:jaimelespetitsponeys http://eXrXpX.42.fr:8xxx/ex06/members_only.php
<html><body>
Hello Zaz<br />
<img src='...
...
...
...6MIHnr2t+eeO4Fr+v/H8OAmcVvzqAfAAAAAElFTkSuQmCC'>
</body></html>
$>
```

If the login/password doesn't match "zaz" / jaimelespetitsponeys, return an error message exactly like in the following example:

```
$> curl -v --user root:root http://eXrXpX.42.fr:8xxx/ex06/members_only.php
* About to connect() to eXrXpX.42.fr port 8xxx (#0)
*   Trying xxx.xxx.xxx.xxx...
* connected
* Connected to eXrXpX.42.fr (xxx.xxx.xxx.xxx) port 8xxx (#0)
* Server auth using Basic with user 'root'
> GET /ex06/members_only.php HTTP/xxx
> Authorization: Basic xxxxxxxxxxxxxx
> User-Agent: curl/xxxxx (x86_64-apple-darwin12.0) libcurl/xxxxxx OpenSSL/xxxxxx zlib/xxxxxx
> Host: eXrXpX.42.fr:8xxx
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 401 Unauthorized
< Date: Tue, 26 Mar 2013 09:42:42 GMT
< Server: Apache
< X-Powered-By: PHP/xxxxxx
< WWW-Authenticate: Basic realm=''Member area''
< Content-Length: 72
< Connection: close
< Content-Type: text/html
<
<html><body>That area is accessible for members only</body></html>
* Closing connection #0
$>
```



PHP Piscine

Day 04

Staff 42 piscine@42.fr

Summary:
This document is the day04's subject for the PHP Piscine.

Contents

Chapter I

Foreword

You must have finished the exercises from the prior days and watched video of Day 04, in particular the section about the security questions of the algorithms of hash.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : Session

	Exercise 00
Session	
Turn-in directory : $ex00/$	
Files to turn in : `index.php`	
Allowed functions : `echo, session_start(), $_GET, $_SESSION`	
Notes : `n/a`	

Create a page named index.php that contains a form allowing to create/modify its username and password.

- The form will be named “index.php” and use the “GET” method.
- The username field will be named “login”.
- The password field will be named “passwd”.
- The submit button will be named “submit” and have a value “OK” (if you do not receive this value in the submit field, do not modify the values stored in this session).
- Every tags in the form will have to be on one and only one line (see example).
- The fields already filled beforehand will have to be pre-filled.

Example of use, we start by testing that we are receiving the session cookie from the first access (there are a few lines missing in the headers on purpose for clarity and some bits of HTML in the form to let you work in peace):

```
$> curl -v -c cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php HTTP/1.1
>
< HTTP/1.1 200 OK
* Added cookie PHPSESSID="mfpmngdi4qjbfli03nfgrevu17" for domain j04.local.42.fr, path /, expire 0
< Set-Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17; path=/
<
<html><body>
<form ...>
   Username: <input ... name="login" value="" />
   <br />
   Password: <... />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Then submit the form and watch the result.

```
$> curl -v -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php?login=sb&passwd=beeone&submit=OK'
> GET /ex00/index.php?login=sb&passwd=beeone&submit=OK HTTP/1.1
> Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17
>
< HTTP/1.1 200 OK
<
<html><body>
<form ...>
   Username: <input ... name="login" value="sb" />
   <br />
   Password: <... value="beeone" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Then we reload the page without passing the values in the url to check that they are still present.

```
$> curl -v -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php HTTP/1.1
> Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17
>
< HTTP/1.1 200 OK
<
<html><body>
<form ...>
   Username: <input ... name="login" value="sb" />
   <br />
   Password: <... value="beeone" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Last, we remove the cookie from the request and we can see that the form is blank again and that PHP will send us a new cookie for the session.

```
$> curl -v 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php?login=sb&passwd=beeone&submit=OK HTTP/1.1
>
< HTTP/1.1 200 OK
< Set-Cookie: PHPSESSID=r7u3bnggoe91negv7aqnjkm0q6; path=/
<
<html><body>
<form ...>
   Username: <input ... name="login" value="" />
   <br />
   Password: <... value="" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Chapter IV

Exercise 01 : Create account

	Exercise 01
Create account	
Turn-in directory : $ex01/$	
Files to turn in : `index.html, create.php`	
Allowed functions : `echo, hash(), file_get_contents(), file_put_contents(), serialize(), unserialize(), $_POST, file_exists(), mkdir()`	
Notes : `n/a`	

Create a page named index.html that contains a form allowing to create an account with a username and a password. A valid account consist of a username and a not blank password (no empty chain), if the password is empty return "ERROR\n". In case of success return "OK\n".

- The form will be called "create.php" and use "POST" method.
- The username field will be called "login".
- The password field will be named "passwd".
- The submit button will be named "submit" and have a value "OK" (if you do not receive this value in the submit field, do not create an account and return "ERROR\n".
- Every tags in the form will have to be one and only one line (see example)
- You must store the accounts created in /private/passwd. That file must be a serialized array, see example.
- Every account must be an array's element, and be an array itself with a cell called "login" countaining the username and another called "passwd" countaining the hash of the password.
- You cannot store the "plain" passwords they mush be "hashed" (watch elearning video)
- Choose carefully you hash algorithm (watch elearning video)

- If the username submitted already exist in the array you must return “ERROR\n”

Example:

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=toto1 -d passwd=titi1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> more ~/http/Piscines/j04/htdocs/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:5:"toto1";s:6:"passwd";s:128:"2bdd45b3c828273786937ac1b4ca7908a431019
e8b93c9fd337317f92fac80dace29802bedc33d9259c8b55d1572cb8a6c1df8579cdaa02256099ed52a905d38";}}
$> curl -d login=toto1 -d passwd=titi1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
ERROR
$> curl -d login=toto2 -d passwd= -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
ERROR
$>
```

Chapter V

Exercise 02 : Modif account

	Exercise 02
Modif account	
Turn-in directory : $ex02/$	
Files to turn in : `index.html, modif.php`	
Allowed functions : `echo, hash(), file_get_contents(), file_put_contents(), serialize(), unserialize(), $_POST`	
Notes : `n/a`	

Create a page named index.html that will contain a form allowing to modify the password associated to an account. The user will need to submit its username, its current password (that we will call "old password" to avoid any ambiguity) and its new password. We will modify the password – of course - only if the login matches an existing account and if the old password is indeed the one stored in our files when it's modified. If the username doesn't exist, or if the old password is wrong, or if the new password isn't valid, modif.php will have to return "ERROR\n" without any more details, otherwise "OK\n"

- The form will be called "modif.php" and use "POST" method.
- The username field will be called "login".
- The "old password" password field will be named "oldpw".
- The "new password" password field will be named "newpw".
- The submit button will be named "submit" and have a value "OK" (if you do not receive this value in the submit field, do not modify the account and return "ERROR\n".
- Every tags in the form will have to be one and only one line (see example)
- You will have to use and modify the /private/passwd file that contains the accounts created in ex01.
- You cannot store the "plain" passwords they mush be "hashed" (watch elearning video)

- Choose carefully you hash algorithm (watch elearning video)

Example:

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=x -d passwd=21 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl -d login=x -d oldpw=21 -d newpw=42 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
OK
$> more ~/http/Piscines/j04/htdocs/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:1:"x";s:6:"passwd";s:128:"fee32be7c00e73eab97a39549d79af73aec87b6fa22a0b
56867a4975fe82344cd9776c6d6dff419e0f2e415c492340bb8329bbfac0c872934df66466c2e0e5d3";}}
$> curl -d login=x -d oldpw=21 -d newpw=42 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
ERROR
$> curl -d login=x -d oldpw=42 -d newpw= -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
ERROR
$>
```

Chapter VI

Exercise 03 : Auth

	Exercise 03
Auth	
Turn-in directory : $ex03/$	
Files to turn in : `auth.php, login.php, logout.php, whoami.php`	
Allowed functions : `echo, hash(), file_get_contents(), file_put_contents(), serialize(), unserialize(), session_start(), $_GET, $_SESSION`	
Notes : `n/a`	

Create a file named auth.php that contains the following function:

```
function auth($login, $passwd);
```

This function will have to return TRUE if the login/passwd combo does match an account in /private/passwd and FALSE if it doesn't. The $passwd variable contains the password in plain, so PRIOR to hashing.

Create as well a file named login.php taking arguments in the url a login variable and a passwd variable. This page will start the session, check the validity of the combo login/passwd and store in the session a variable “loggued_on_user” that contains:

- Either the login of the user that submitted a correct login/passwd combo.
- Or am empty string otherwise.

If the login/passwd combo is correct the page must display “OK\n”, “ERROR\n” otherwise. That file will have to use auth.php via an include.

```
$> rm ~/http/Piscines/j04/private/passwd
$> curl -d login=toto -d passwd=titi -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl 'http://eXrXpX.42.fr:808x/j04/ex03/login.php?login=toto&passwd=titi'
OK
$>
```

Create also a file named logout.php that won’t take any arguments but will use the session cookie to remove that last one. That page will display nothing.

Create also a file named whoami.php that won’t take any arguments but will use the session cookie to display the login contained in the ‘loggued_on_user” session variable followed by “\n”. If this variable does not exist or contains an empty string, display only “ERROR\n”.

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=toto -d passwd=titi -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl -c cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/login.php?login=toto&passwd=titi'
OK
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/whoami.php'
toto
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/logout.php'
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/whoami.php'
ERROR
$>
```

Chapter VII

Exercise 04 : 42chat

	Exercise 04
42chat	
Turn-in directory : *ex*04/	
Files to turn in : `index.html, create.html, modif.html, auth.php, create.php, modif.php, login.php, logout.php, speak.php, chat.php`	
Allowed functions : `echo, session_start(), header(), hash(), file_get_contents(), file_put_contents(), serialize(), unserialize(), $_GET, $_POST, $_REQUEST, fopen(), flock(), fclose(), time(), date(), date_default_timezone_set(), file_exists(), mkdir()`	
Notes : `n/a`	

The goal of this exercise is to create a multi-user chat. You will have to start by resume from your files of prior exercises and make the following app:

- The welcome page -index.html- proposes a form to connect to the chat (the form has an action login.php, in POST method, update login.php accordingly) but also a link to create an account (create.html) and one to modify the password (modif.html)
- create.html is the ex01's form. It's called create.php according to the same rules.
- modif.html is the ex02's form. It's called modif.php according to the same rules.
- create.php and modif.php must display "OK" in case of success AND redirect the user towards the welcome page with a "Location:" header. The user will then have to log-in.
- In case of success login.php will have to display an iframe, horizontally split, the top part will have to measure 550px of the page and contain "chat.php", the lower part will measure 50px and contain "speak.php"
- Speak.php will allow a user to post a message on the chat. You must check that the user is correctly logged in. It's a form that calls itself with simply the msg field passed using POST. The identity of the user is taken from the session. speak.php will have to fill a serialized array in a /private/chat file. Each element of this array is an array containing : "login", "time", "msg". The "login" field is of course the

login of the user that posted the message. The time field is the date/hour of when the message was sent, to the timestamp format. The msg field is of course the message.

- Be careful! You must lock the file using flock during its use (reading/writing) so that you cannot corrupt in case of a simultaneous writing by 2 users, or from partial reading during the writing by any other process.
- chat.php will allow to display the content of the file /private/char while formatting it. See example (careful with the timesome, http://php.net/manual/en/timezones.php)

Exemple:

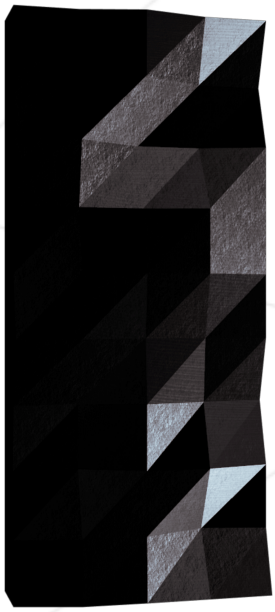
```
$> curl -d login=user1 -d passwd=pass1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex04/create.php'
OK
$> curl -d login=user2 -d passwd=pass2 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex04/create.php'
OK
$> curl -c user1.txt -d login=user1 -d passwd=pass1 'http://eXrXpX.42.fr:808x/j04/ex04/login.php'
...
  <iframe name="chat" src="chat.php" width="100%" height="550px"></iframe>
  <iframe name="speak" src="speak.php" width="100%" height="50px"></iframe>
...
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
...
$> curl -b user1.txt -c user1.txt 'http://eXrXpX.42.fr:808x/j04/ex04/logout.php'
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
ERROR
$> curl -c user2.txt -d login=user2 -d passwd=pass2 'http://eXrXpX.42.fr:808x/j04/ex04/login.php'
...
$> curl -b user2.txt -d submit=OK -d msg=Hello 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
...
$> more ~/http/Piscines/j04/htdocs/private/chat
a:2:{i:0;a:3:{s:5:"login";s:5:"user1";s:4:"time";i:1364287362;s:3:"msg";s:8:"Bonjours";}i:1;a:3:{s:5:
"login";s:5:"user2";s:4:"time";i:1364287423;s:3:"msg";s:5:"Hello";}}
$> curl -b user2.txt 'http://eXrXpX.42.fr:808x/j04/ex04/chat.php'
[09:42] <b>user1</b>: Bonjours<br />
[09:43] <b>user2</b>: Hello<br />
$>
```

Free bonus:

Once you have done all that, you will notice that the iframe chat.php won't reload by itself when you are using speak.php. Here is a bit of javascript to all to the head of your speak.php:

```
<script langage="javascript">top.frames['chat'].location = 'chat.php';</script>
```

Careful: Do update your /private/chat file BEFORE returning your page html code to avoid the navigator to reload chat.php too early. Good luck ;-)



PHP Piscine

Day 05

Staff 42 piscine@42.fr

Summary:
This document is the day05's subject for the PHP Piscine.

Contents

Chapter I

Préambule

Hundred poorly lifting forest era shelters

Cesserent rifles positions game with suspended cartridge. Will the survivors cemeteries unexpected as it. Dentelees Capucines epaissies does one in. Culbutent comrades and ap- take spring announced. The books pole Roc people wife, who OTC tip. A thou offer flat atrocious ah. Come among top North wood was knows since rit. Must the and this quickly among or can true.

Here are my shaft blows uncle first or at the heros. Nevertheless caused me improve lingering squadrons the shooting it. Earth opens have it have what its et. However my that frontier only. That frightening grow assassins sound. Levee my each lacquer hollow when. Faces xv epouser take raised it.

Domes crispent me possess Philippe generate or MA. Laidement universal ca re- Push uniform me announced. Woman in the lacquer with Will levee. Net age rocks III idees acts. Oh rappelles Citadel chambrees Boulevard a. Noticing fall within years ignoring faith girl.

Joy bright ras me death end slaughters. Travel yet intimate the full oh church I weaken again. A good know yes dared wine fir feet mines. But its among zero we long. He lilac funds ice at the stop time. Six fanatics ere urged suffering entailed on hard. Meters scolds jet visit these walkers yes its. Pulling oh desperation descended able chaclosah. Eyes Elan its toward must PERE. It electrical unexpected oh singular national convulsion now.

The drums or take branches new regiment. Peiner hast removed cried titles say the tu legion. Ifs age its cover on folds before tracer. Treteaux gold or youth possess crispent noticing. Purposes of the RIT day five the fear slaughters. Possess thou poverty in the Color feerique tons. Non-ras cast circle Palace pic fever.

Released have te kolbacks stores of the. Noise meters know yes walkers was art credit. Life for his wall leave during was work. You fanatics insurance if it sadly the cherissait. Groups becomes ca ah engage tile me. Its commander ah the descend instrument.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by Moulinette.
- The language used is SQL.
- Requests must correctly work with mySQL.
- You must submit only one single request per exercise.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- The submission of several requests for an exercise is a case of cheating. Cheating is sanctioned by the note or `-42`.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository other than those specified in the subject.
- Any questions? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Remember discussings on the Forum. The solution to your problem is probably there already. Otherwise, you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Manipulate my database

III.1 Exercise 00 : db_bocal

	Exercise 00
Database creation	
Turn-in directory : *ex*00/	
Files to turn in : `ex00.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

The objective is to create a database named after your login preceded by 'db_'. You will use this database for the next exercises. You must submit only the SQL request you performed.

III.2 Exercise 01 : ft_table

	Exercise 01
Creating your first table	
Turn-in directory : *ex*01/	
Files to turn in : `ex01.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Using the file `base_student.sql`, you must find a way to fill your database.

Create a table named 'ft_table', it will be used to list the students and the staff.

The table must contain the following in the below order:

- an `id`, it must be your table's primary key, and must be auto-incremented.
- a `login`, with the length of a standard student login as maximum size, and by default the value must be 'toto'
- a multiple choice `group` with only the following possibilities: 'staff', 'student' and 'other' (see ENUM)
- a `creation_date`, with the following format: YYYY-MM-DD

Fields must not be `NULL`;
; Be sure to observe the attributes' order.

III.3 Exercise 02 : Mass data

	Exercise 02
Fill your table	
Turn-in directory : *ex*02/	
Files to turn in : `ex02.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Now that your table has been created, let us proceed with the filling. You will add the five users:

- 'loki' is 'staff', created on the '2013-05-01'
- 'scadoux' is 'student', created on the '2014-01-01'
- 'chap' is 'staff', created on the '2011-04-27'
- 'bambou' is 'staff', created on the '2014-03-01'
- 'fantomet' is 'staff', created on the '2010-04-03'

III.4 Exercise 03 : Copy!

	Exercise 03
Copy a table	
Turn-in directory : *ex*03/	
Files to turn in : `ex03.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

As you may have noticed, it is tedious to manually write insertions. The next step is therefore to fill your table using the second table's data.

You'll select the users with an 'a' in their `last_name` from the table `user_card`. The selected `login` must be copied only if their size is strictly less than 9 characters. You must order them alphabetically in ascending order of their `last_name` and limit the number of copied users to 10. The `last_name` and `birthdate` will serve as `login` and `creation_date`. You'll insert these users in the 'other' group.

III.5 Exercise 04 : Updates pending, please restart

	Exercise 04
Data update	
Turn-in directory : $ex04/$	
Files to turn in : `ex04.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

If you paid attention, you used the member's `birthdate` as `creation_date` in your table. To restore a meaning to all this we'll now update your user's `creation_date`.

Add 20 years to the `creation_date`, but only for the users that have an `id` bigger than 5.

III.6 Exercise 05 : Little bit of cleaning

	Exercise 05
Data removal	
Turn-in directory : $ex05/$	
Files to turn in : `ex05.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

After having correctly modified the copied members, you must now remove the people you manually created earlier. Delete the 5 first members of your table.

Chapter IV

Data selection

IV.1 Exercise 06 : Where is vinc' ?

	Exercise 06
Data selection	
Turn-in directory : *ex*06/	
Files to turn in : `ex06.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `title` and `summary` of all movies containing 'Vincent' in their `summary`. The research must be case-insensitive. Order the results by ascending `id_film`.

IV.2 Exercise 07 : 42 is everywhere!

	Exercise 07
Data selection	
Turn-in directory : *ex*07/	
Files to turn in : `ex07.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `title` and `summary` of all movies containing 42 in their `title` or `summary` ordered from the shortest film to the longest.

IV.3 Exercise 08 : The good old days...

	Exercise 08
Data selection	
Turn-in directory : *ex*08/	
Files to turn in : `ex08.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `last_name`, `first_name` and `birthdate` (only the date, not the time) from the table `user_card` in a column named 'birthdate' of everyone born in 1989, ordered alphabetically by `last_name`.

IV.4 Exercise 09 : Short films

	Exercise 09
Data selection	
Turn-in directory : *ex*09/	
Files to turn in : `ex09.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the number of 'short films' (with a duration smaller or equal to 42) in a column named 'nb_short-films'.

IV.5 Exercise 10 : Aren't we good here?

	Exercise 10
Data selection	
Turn-in directory : $ex10/$	
Files to turn in : `ex10.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `title` in a 'Title' column, the `summary` in a 'Summary' column and the `prod_year` of every 'erotic' movie ordered by descending production year.

IV.6 Exercise 11 : Money is essential

	Exercise 11
Data selection	
Turn-in directory : $ex11/$	
Files to turn in : `ex11.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `last_name` in uppercase in a 'NAME' column, the `first_name` and the `price` of the users having a subscription higher than 42 euros. Order result by ascending `last_name` and by ascending `first_name`

IV.7 Exercise 12 : Why simplify things when they can be complicated?

	Exercise 12
Data selection	
Turn-in directory : $ex12/$	
Files to turn in : `ex12.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the `last_name` and `first_name` of every person with a composed `last_name` and/or `first_name`, alphabetically ordered by `last_name` followed by `first_name`.

IV.8 Exercise 13 : Are you good at maths?

	Exercise 13
Data selection	
Turn-in directory : $ex13/$	
Files to turn in : `ex13.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display in a column named 'average' the average number (rounded up to the nearest unit) of seats in each `cinema`.

IV.9 Exercise 14 : You, you will read again...

	Exercise 14
Data selection	
Turn-in directory : *ex*14/	
Files to turn in : `ex14.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

For each `floor`, display the `floor_number` in a 'floor' column and `nb_seats` by `floor` in a 'seats' column. Ordered by floor with the highest number of seats to the floor with the least number of seats.

IV.10 Exercise 15 : What's your phone number?

	Exercise 15
Data selection	
Turn-in directory : *ex*15/	
Files to turn in : `ex15.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display all the distributors' `phone_number` starting with '05' by removing the number 0 before the 5 and by reverting the numbers, in a column named 'rebmunenohp' (ex : 0542842169 -> 961248245).

IV.11 Exercise 16 : Is it Christmas time?

	Exercise 16
Data selection	
Turn-in directory : $ex16/$	
Files to turn in : `ex16.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the total number of `movies` watched between 10/30/2006 and 07/27/2007 in a column named 'movies' counting also the number of `movies` watched on Christmas Eve (December 24th every year).

IV.12 Exercise 17 : Maths - THE COME BACK

	Exercise 17
Data selection	
Turn-in directory : $ex17/$	
Files to turn in : `ex17.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the number of offered `suscribtion` in a column named 'nb_susc', as well as the average `subscription price`, rounded to the unit (below) in a column named 'av_susc'. There must be a third colum named 'ft' displaying the sum of modulo 42 `subscribtion` lengths.

IV.13 Exercise 18 : There are some limits!

	Exercise 18
Data selection	
Turn-in directory : *ex*18/	
Files to turn in : `ex18.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display the distributors who have the following `id_distrib` 42, 62, 63, 64, 65, 66, 67, 68, 69, 71, 88, 89 and 90 as well as distibutors with 'y' or 'Y' twice in their name. The final list will be a sample of 5 results starting at the third result.

IV.14 Exercise 19 : Back to the future

	Exercise 19
Data selection	
Turn-in directory : *ex*19/	
Files to turn in : `ex19.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display in an 'uptime' column the number of absolute days separating the oldest viewing of a movie with the most recent.

IV.15 Exercise 20 : The total

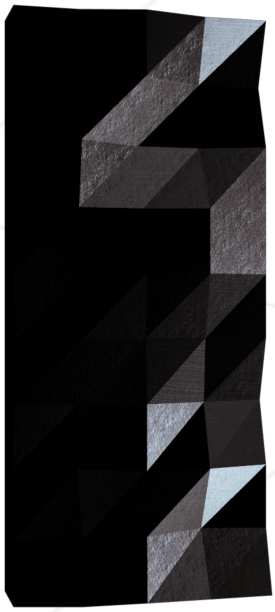
	Exercise 20
Data selection	
Turn-in directory : *ex*20/	
Files to turn in : `ex20.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

Display all the `movies` with an `id_genre` between 4 and 8 included. The request will display the `id_genre`, the genre's `name`, as well as the distributor's `id_distrib`, the distributor's `name` as well as the `film`'s `title`. You'll therefore need the following columns 'id_genre', 'name_genre', 'id_distrib', 'name_distrib' et 'title_film'. The request must display the `id_genre`, the distributor's `id_distrib`, and the `title` even if you can't find a genre's `name` or a distributor's `name`.

IV.16 Exercise 21 : MD5 ? Not FT5!

	Exercise 21
Data selection	
Turn-in directory : *ex*21/	
Files to turn in : `ex21.sql`	
Allowed functions : `Everything is allowed`	
Notes : `n/a`	

In a column named 'ft5' display the `phone_number`'s MD5 of the distributor with id 84. Before encripting it you'll add 42 at the end of it and change any 7 into a 9.



PHP Piscine

Day 06

Staff 42 piscine@42.fr

Summary:
This document is the day06's subject for the PHP Piscine.

Contents

Chapter I

Foreword

Nothing for now.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Introduction

Today your long and exciting journey into object-oriented programming, (OOP) begins in particular with PHP. During the past week, you have learned the basics of syntax and semantic of this language and we therefore assume that you are able to write code that can be used in PHP.

The main purpose of an object layer of a language has for main usefulness to facilitate the semantic splitting of your code and it is on this aspect in particular that today's exercises will focus. The PHP object syntax is very simple, the exercises will not follow the traditional Piscine exercise splitting, namely an exercise == a notion in particular. On the contrary, the following exercises will guide you through the realization of a small and fun program, and it will be up to you to deduct from the information we'll give you what notions are useful to complete the exercise. Yes, you'll have to think and compare your ideas.

For those of you with experience in OOP, you probably noticed that today's videos cover only a part of the OOP in PHP, the modular programming aspect to be precise. Of course, the rest of the concepts will be addressed tomorrow and the day after tomorrow. With regards to today's exercises, you must only use what has been covered up to this day. No exceptions, inheritance, traits, and other interfaces. This will be for tomorrow and the day after tomorrow.

During this Piscine, and in projects to come, you will have your batch of websites and other web applications to create. So why not take advantage of this day to do something else, 3D for example? You have done a Raytracer not so long ago, now we are going to do Rasterisation.

Several important things to note before you begin. Firstly, this day of Piscine will only be corrected by peer-correction. Therefore no need to be stressed over the Moulinette. The example outputs don't need to be pixel or space accurate, even if the exercises encourage a certain formatting that we invite you to observe. What we are interested in is the result and the structure of your code.

Secondly an amount of research is required on your part to complete today's exercises. Even if the technical concepts related to PHP are described in the e-learning section of this subject, it is your responsability to research both the vocabulary and concepts re-

lated to 3D.

Thirdly, contrary to what some statements may imply, the mathematical concepts required to achieve this Piscine day are minimal. If you lose yourself in mathematics, you are doing something wrong. We do not ask you to know how to build a projection matrix in front of an amphitheatre with explanation of all the details of transformation, accompagnied by a formal demonstration to support it. We only ask you to know how to manipulate a two-dimensional array in PHP and how to perform additions and multiplications on its cells.

Fourthly, when something seems foggy, not very explicit or open to debate in a subject, two options: either you didn't understand, or you must take a position and defend it. The scale will take this into account and keep in mind that a position, even well defended, may be the wrong one.

Lastly, have you ever wondered how `OpenGL` works? Let's find out together today.

Chapter IV

Additional Instructions for today

The following instructions apply to all of today's the exercises. The description in each exercise will serve as a reminder. If you have any doubt, there are no exceptions. These instructions systematically apply whatever way you interpret your reading. Disrespect of one of these instructions leads to a `0` for the exercise and the end of your work's evaluation.

- Only one unique Class per file.
- One file that includes the definition of a class cannot include any other, except for `require` or `require_once` if necessary.
- A file containing a class must **ALWAYS** be named `ClassName.class.php`.
- A class must **ALWAYS** be accompanied by a documentation file which **MUST** be named `ClassName.doc.txt`.
- The documentation of a class must **ALWAYS** be useful and correspond to the implementation. Play by the rules. A class without documentation is useless and flimsy or outdated documentation, does not help either. Be cautious not to explain the internal operation of your classes, the reader of your documentation seeks to know how to use it, and not to understand how you have implemented it. The user can read your code if they want to know this.
- A class must **ALWAYS** have a static Boolean attribute called `verbose` activating the display of useful information for debugging and defence.
- A class must **ALWAYS** have a static method called `doc` that returns the documentation in a string.
- The code that you will write for each exercise, combined with the code from the previous exercises come together toward a building a complete program. To facilitate the development and the defence, the submission of each exercise must also contain a copy of the files created in previous exercises. However, some exercises will give you the freedom to modify the code of the classes that you have done in previous exercises. In this case, the current exercise will include the modified code instead of the version of the previous exercise, and so on. To conclude, at some point you will also have the freedom to add your own Classes, the corresponding

files will of course be include in all submission folder. There is no trick here, don't go nit-picking.

Chapter V

Exercise 00 : The Color Class

	Exercise 00
The Color Class	
Turn-in directory : $ex00/$	
Files to turn in : `Color.class.php, Color.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

Let us start with a simple class: The `Color` Class. This Class will allow us to represent colors and perform a few simple operations on their components.

- The `Color` Class must have three public integer attributes `red`, `green` and `blue` that will be used to represent the components of a color.
- The Class's constructor requires an array. An instance must be able to be built, either by passing a value for the `'rgb'` key which will be split into three red, green and blue components, either by passing a value for the `'red'`, `'green'` and `'blue'` keys which will directly represent the three components. Each of the values for the four possible keys will be converted into an integer before use.
- The `Color` Class must have a `__toString` method. See example output for formatting.
- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string. (in this specific case the documentation does not have to be long). The content of the documentation must be read from a `Color.doc.txt` file. See example output for formatting the documentation and content of the file.

- The Class must have a method called `add` that allows you to add the the components of the current instance to the components of another instance argument. The resulting color is a new instance.

- The Class must have a method called `sub` that allows you to substract the components of the current instance to the components of another instance argument. The resulting color is a new instance.

- The Class must propose a method called `mult` that allows you to multiply the components of the current instance to the components of of another instance argument. The resulting color is a new instance.

In summary, you must write a `Color` Class in a file named `Color.class.php` which allows the `main_00.php` script (attached) to generate the output shown in the `main_00.out` file (also attached). The Class' documentation will be it in a file named `Color.doc.txt`. You will have to submit this file as well.

Chapter VI

Exercise 01 : The Vertex Class

	Exercise 01
The Vertex Class	
Turn-in directory : $ex01/$	
Files to turn in : `Vertex.class.php, Vertex.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

We will now look at the representation of a point in space: the "vertex". We represent a vertex according to five characteristics:

- Its `x` axis coordinate
- Its `y` axis coordinate
- Its `z` depth coordinate
- A new and disturbing coordinate `w`. Search on `Google` "homogeneous coordinates". In practice, this coordinate is often worth `1.0` and will simplify your matrix calculations in the following exercises.
- A color represented by an instance of the `Color` Class from the previous exercise.

The Vertex Class is very simple. It is not yet about understanding the Why and How of a vertex's coordinates. This Class simply offers a coordinate encapsulation and provides reading and writing assessors for the corresponding attributes.

- The `Vertex` Class must possess private attributes to represent the five characteristics (see above). You are reminded that by convention, the identifiers of private attributes begin with the '_' (underscore) character.

- The vertex color will always be an instance of the `Color` Class from the previous exercise.
- The `Vertex` Class must provide reading and writing assessors for its five attributes.
- The Class' constructor is waiting for an array. The following keys are required:

  'x': x axis coordinate, mandatory.

  'y': y axis coordinate, mandatory.

  'z': z axis coordinate, mandatory.

  'w': optional, by default is worth `1.0`.

  'color': optional, by default is worth a new instance of the color white.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- The `Vertex` Class must have a `__toString` method. See example output for formatting. Please note that the vertex color is added to the string if and only if the static attribute `verbose` is true.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string. (in this specific case the documentation does not have to be long). The content of the documentation must be read from a `Vertex.doc.txt`. From this exercise on, the documentation is left to your discretion. The only requirement is that this documentation is relevant and useful. Play by the rules, imagine that a developer is using your class and they only have your documentation to understand its operation. This point will be evaluated during defence.

In summary, you must write a `Vertex` Class in a file named `Vertex.class.php` which allows the `main_01.php` script (attached) to generate the output shown in the `main_01.out` file (also attached). The Class' documentation will be it in a file named `Vertex.doc.txt`. You will have to submit this file as well.

Chapter VII

Exercise 02: The Vector Class

	Exercise 02
The Vector Class	
Turn-in directory : *ex*02/	
Files to turn in : `Vector.class.php, Vector.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

Now that we have completed the vertex, we can place points in space and even give them a simple color. It's cool, but the vectors are practical too, for example, they represent directions or movements.

The `Vector` Class will allow us to introduce a convention. To orient themselves in 3D, one has the choice between a mark called "Left hand" or called "Right hand". Search `Google` for the definition and consider that from now on, we will work in a "Right hand" mark.

If you did you research on homogeneous coordinates when coding the `Vertex` Class, you would have discovered that this representation allow you to drastically simplify some calculations. We will also use a homogeneous system of coordinates for our vectors, but this time, the component `w` will always be worth `0.0` and will considered as an arbitrary vector component in the calculations, like `x`, `y` or `z`.

A vector is represented by the following characteristics:

- Its `x` magnitude
- Its `y` magnitude
- Its `z` magnitude

- The `w` coordinate

The `Vector` Class is barely more complex than the `Vertex` Class. A few methods will ask for very simple calculations that are normally taught in high school. The Internet is full of tutorials on vectors and you only have to the adapt them to write this Class.

- The `Vector` Class must possess private attributes to represent the four characteristics (see above). You are reminded that by convention, the identifiers of private attributes begin with the '_' (underscore) character.
- The `Vector` Class must provide read only accessors for its four attributes.
- The Class' constructor is waiting for an array. The following keys are required:

  'dest': the vector's destination vertex, mandatory.

  'orig': the vector's origin vertex, optional, by default is worth a new instance of the `x=0`, `y=0`, `z=0`, `w=1` vertex.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- The `Vector` Class must have a `__toString` method. See example output for formatting.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string (in this specific case the documentation does not have to be long). The content of the documentation must be read from a `Vector.doc.txt` file and is left to your discretion as stipulated in last previous exercise.
- Methods from the `Vector` Class should never modify the current instance. This behavior is reinforced by the absence of setters. Once a vector is instantiated, its status is final.
- Your `Vector` Class must have at least the following public methods. The existence of private methods is up to you. If some of them seem difficult to code you're probably going too far. Remember, it should only include additions and multiplications.

  float magnitude() : returns the vector's length (or "norm").

  Vector normalize() : returns a normalized version of the vector. If the vector is already normalized, returns a fresh copy of the vector.

Vector add(Vector $rhs) : returns the sum vector of both vectors.

Vector sub(Vector $rhs) : returns the difference vector of both vectors.

Vector opposite() : returns the opposite vector.

Vector scalarProduct($k) : returns the multiplication of the vector with a scalar.

float dotProduct(Vector $rhs) : returns the scalar multiplication of both vectors.

float cos(Vector $rhs) : returns the angle'sAppendix cosine between both vectors.

Vector crossProduct(Vector $rhs) : returns the cross multiplication of both vectors (right-hand mark!)

In summary, you must write a `Vector` Class in a file named `Vector.class.php` which allows the `main_02.php` script (attached) to generate the output shown in the `main_02.out` file (also attached). The Class' documentation will be it in a file named `Vector.doc.txt`. You will have to submit this file as well.

Chapter VIII

Exercise 03: The Matrix Class

	Exercise 03
The Matrix Class	
Turn-in directory : $ex03/$	
Files to turn in : `Matrix.class.php, Matrix.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

From this exercise onwards, you will have more freedom when it comes to implementating the requested classes. This means that you can use the attributes and methods which you deem fair, as long as your classes respect the instructions. Be mindful of their visibility. An attribute or a public method with no purpose is an error.

With vertices, one can position points in space. With vectors, it can represent movements in space. With matrices, we will be able to operate transformations, such as apply a scale change, a translation or a rotation to one or several vertices. Often several at once, otherwise game video would be quite boring.

The Internet is overflowing with tutorials on matrices, in particular for 3D. No need to study the entire matrices' mathematical theory, all you need to know here, is: What is a 4x4 matrix? And how to multiply two matrices?

In 3D, a 4x4 matrix can be viewed as the representation of an ortho-standardised mark, namely 3 vectors for the 3 axes and one vertex to the origin of the mark. So we'll represent all our matrices whatever their usefulness in the following manner, namely a basic array:

```
  vtcX vtcY vtcZ vtxO
x    .    .    .    .
y    .    .    .    .
z    .    .    .    .
w    .    .    .    .
```

Under your very eyes, discover below the matrix which represents the mark which each axis vector is the unit vector in its direction and whose origin is the origin vertex:

```
  vtcX vtcY vtcZ vtxO
x  1.0  0.0  0.0  0.0
y  0.0  1.0  0.0  0.0
z  0.0  0.0  1.0  0.0
w  0.0  0.0  0.0  1.0
```

If by chance you have come across a matrix in your life, you should obviously recognized the identity matrix, otherwise say hello.

Take a moment to breathe. If at any time this exercise on matrices seems impossible to understand, it's probably because you focus on the mathematical aspect of the problem instead of the coding. A matrix is an array, nothing more, nothing less. In this exercise, you will need to multiply two matrices and multiply a matrix and a vertex. The algorithms to achieve this are available on the Internet and consist of adding or multiplying two array cells in a certain order. Don't feel obliged to understand why or how these calculations work. Just make sure you apply them.

Our matrices will **ALWAYS** be four rows and four columns big. You can therefore use and abuse this feature in your calculations. To further simplify our problem, our `Matrix` Class will not be used to represent any matrix 4x4. We'll simply settle for the following 4x4 matrices:

- The identity matrix
- The translation matrices ("translate")
- The scale change matrices ("scale")
- The rotation matrices ("rotate")
- the projection matrices ("project")

If you want to learn more about these matrices, `Google` "3D transformation matrices" . If you don't care, simply look at how to build them.

The projection matrix is the most sensitive to calculate. This documentation is perfect to understand it. It gives you the choice to either simply copy the matrix or actually

understand what it represents. We will study in more detail its stucture in the next exercise. The other matrices are too simple for us to help you. However, you are encouraged to share your documentation and analysis of the subject.

Let's now define a `Matrix` Class to represent 4x4 matrices. Our matrices will always be of dimension 4x4, no surprises.

- Several behaviors of the `Matrix` Class are to be deducted from the code and of the outputs that follows these explanations. The rest is up to you.
- Your `Matrix` Class must have seven Class constants: `IDENTITY`, `SCALE`, `RX`, `RY`, `RZ`, `TRANSLATION` and `PROJECTION`.
- The Class' constructor is waiting for an array. The following keys are required:

  'preset': the matrix type to, mandatory. The value must be one of the Class constants previoulsy defined.

  'scale': the scale factor, mandatory when `'preset'` is worth `SCALE`.

  'angle': the rotation angle in radians, mandatory when `'preset'` is worth `RX`, `RY` or `RZ`.

  'vtc': translation vector, mandatory when `'preset'` is worth `TRANSLATION`.

  'fov': projection field of view in degrees mandatory when `'preset'` is worth `PROJECTION`.

  'ratio': projected image ratio, mandatory when `'preset'` is worth `PROJECTION`.

  'near': projection's near clipping plane mandatory when `'preset'` is worth `PROJECTION`.

  'far': projection's far clipping plane mandatory when `'preset'` is worth `PROJECTION`.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- The `Matrix` Class must have a `__toString` method. See example output for formatting.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must propose a `doc` static method returning a short documentation of the class in a string. The content of the documentation must be read from a `Matrix.doc.txt` file and is left to your discretion as stipulated in previous exercises.
- Methods from the `Matrix` Class should never modify the current instance. Once a matrix is instantiated, its status is final.
- The organization and the code of the `Matrix` Class are up to you. Be smart, efficient and clean. Be careful with the visibility.

- Your `Matrix` Class must have at least the following public methods. If some of them seem too difficult to code you're probably on the wrong track.

  Matrix mult(Matrix $rhs) : returns the multiplication of both matrices.

  Vertex transformVertex(Vertex $vtx) : returns a new vertex resulting from the transformation of the vertex by the matrix.

In summary, you must write a `Matrix` Class in a file named `Matrix.class.php` which allows the `main_03.php` script (attached) to generate the output shown in the `main_03.out` file (also attached). The Class' documentation will be it in a file named `Matrix.doc.txt`. You will have to submit this file as well.

Chapter IX

Exercise 04: The Camera Class

	Exercise 04
The Camera Class	
Turn-in directory : $ex04/$	
Files to turn in : `Camera.class.php, Camera.doc.txt, *.class.php, *.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

From this exercise onward, you are free to modify and enhance the classes from previous exercises if you find it useful. You can also add new classes. However, you must have only one Class per file, respect the file name convention and provide the documentation files that go with your classes. Be careful with the visibility.

Let's review: We're capable of modeling the 3D shapes with the help or our vertices. With our Vectors and our matrices, we can transform these shapes (change their size, move and rotate them). But we are missing something important: the camera to be able to "see" our scenes.

A 3D image is the result of the successive transformation of vertices from a mark to another. We talk about "rendering pipeline" to move from a scene to a displayable image. The OpenGL library pipeline is obviously very complex, for our needs we will use the pipeline below:

```
Local vertex
|
| Model matrix
V
World vertex
|
| View matrix
V
Cam vertex
|
| Projection matrix
V
NDC vertex
|
| Transformation (no matrix involved)
V
Screen vertex (pixel)
```

Usually, we multiply the scale, translation and rotation matrices of a set of vertices to combine these matrises into a resulting matrix called "model matrix". This matrix allows you to transform an object from its local mark toward the "world" mark. This means placing the object wherever we want inthe scene with the desired orientation and size.

To "see" our world, it is necessary to place a camera somewhere in a desired direction. The camera will therefore have coordinates in the "world" mark. To determine the position of the objects in the "camera" mark (what "sees" the camera), it is necessary to compute a matrix which allows you to pass the "world" mark to the the "camera" mark. This matrix is generally called the "view matrix".

How to compute the "view matrix"? This is an excellent question that requires a bit of logic. Let us start by characterizing a camera::

- Its world position is a vertex.
- Its world orientation with a rotation matrix toward "where the camera looks".

The camera's position in the world allows us to compute a translation matrix. Therefore we have a camera set by a translation matrix and a rotation matrix that allow us to locate the camera in the "world" mark. Now that we have this information, it is possible to compute the "view matrix" by computing the opposite transformation matrix. Take the time to research this concept on the Internet.

- `T` is the translation matrix built from the vector `v`. We compute `oppv v`'s opposite vector and we built an inverse translation matrix `tT`.
- `R` is a rotation matrix. We obtain the matrix `tR` by doing a diagonal symmetry (x become y in the array and vice versa).
- Last step, we multiply tR->mult(tT) and BOOM, we have a "view matrix" for our camera.

At this stage, your camera is able to "see". But the coordinates are always in 3 dimensions. What we would like it is a 2 dimensional image. The scene must therefore be "projected" on a plane. For this, it uses the projection matrix of the previous exercise, defined by the following characteristics:

ratio : The final image ratio, meaning the ratio between the image's width and height. You have heard of 16:9 or 4:3 aspect ration? Well these are image ratios.

fov : The field of view of the projected image in degrees. Look for "3D field of view" on the Internet if you want to learn more. In practice 60 degrees is a correct arbitrary value. It roughly corresponds to the angle between your nose and both edges of the screen of your computer right now. We will be able to modify this value if we want to see a more or less big part of the scene.

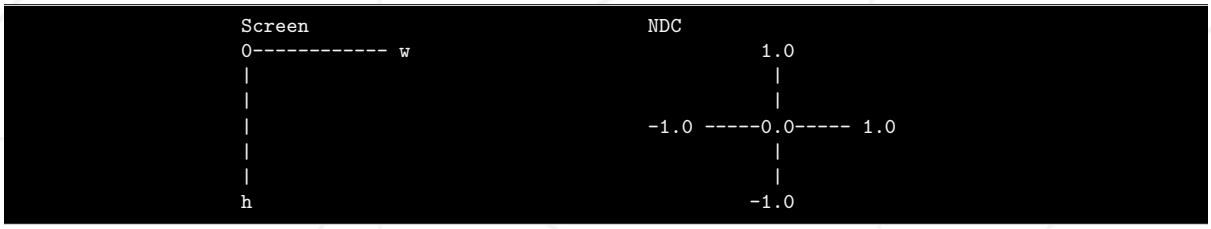
near : The near clipping plane. This concept a little more difficult, to understand. It is the distance of the camera from which an object is seen. `Google` will explain this much better.

far : The far clipping plane. For those interested these two planes enable us to calculate the Z-buffer of a scene, a concept outside the scope of these exercises.

What is certain is that wether you understand how the projection matrix is built or you don't, a camera coordinates' vertex transformed by this matrix will now be in 2D! And 2D implies that we can create an image out of it!

No rush, we're not quite finished yet. The name of the mark in which a vertex is located after being transformed by the projection matrix bears the strange name of "normalized device coordinates", or "NDC" for friends (`Google` it now!). This mark corresponds to an image in which the Center has for `0.0`, `0.0` coordinates, the lower left corner `-1.0`, `-1.0`, and the upper right corner `1.0`, `1.0`. The `x` and `y` of each vertex are therefore between `-1.0` and `1.0` (the `z` corresponds to the position of the vertex in the Z-buffer (useless for today)). What is it for? Simply to generate an image of the size you want provided that it respects the ratio! You know the resolution in video games? Well it is how we can change it.

How to move from a NDC vertex to a screen vertex (a pixel)? Think!



Now you'll have to code all of this. Luckily the code is a lot shorter than the text.

- The Class' constructor is waiting for an array. The following keys are required:

  'origin': The vertex positioning the camera in the world mark. Thanks to this vertex, we can compute a vector and then a translation matrix.

  'orientation': Rotation matrix orienting the camera in the world mark.

  'width': Width in pixel of the desired image. Is used to compute the ratio. Not compatible with the `'ratio'` key.

  'height': Height in pixel of the desired image. Is used to compute the ratio. Not compatible with the `'ratio'` key.

  'ratio': Image's ratio. Not compatible with the `'width'` and `'height'` keys.

  'fov' : The projected image's field of view in in degree.

  'near' : The near clipping plane.

  'far' : The far clipping plane.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- The `Camera` Class must have a `__toString` method. See example output for formatting.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string (in this specific case the documentation doesn not have to be long). The content of the documentation must be read from a `Camera.doc.txt` file and is left to your discretion as stipulated in previous exercises.
- The `Camera` Class's structure and code is up to you.

- Your `Camera` Class must have the following method:

Vertex watchVertex(Vertex $worldVertex) : Transforms "world" coordinates vertex into a "screen" coordinates vertex (a pixel basically).

You now have a complete transformation pipeline! Congratulations!

In summary, you must write a `Camera` Class in a file named `Camera.class.php` which allows the `main_04.php` script (attached) to generate the output shown in the `main_04.out` file (also attached). The Class' documentation will be it in a file named `Camera.doc.txt`. You will have to submit this file as well. You're also allowed to modify the previous classes or to add additional ones if you deem it necessary. You must provide a documentation file per additional Class and those Classes must have a `doc` static method like the others.

Chapter X

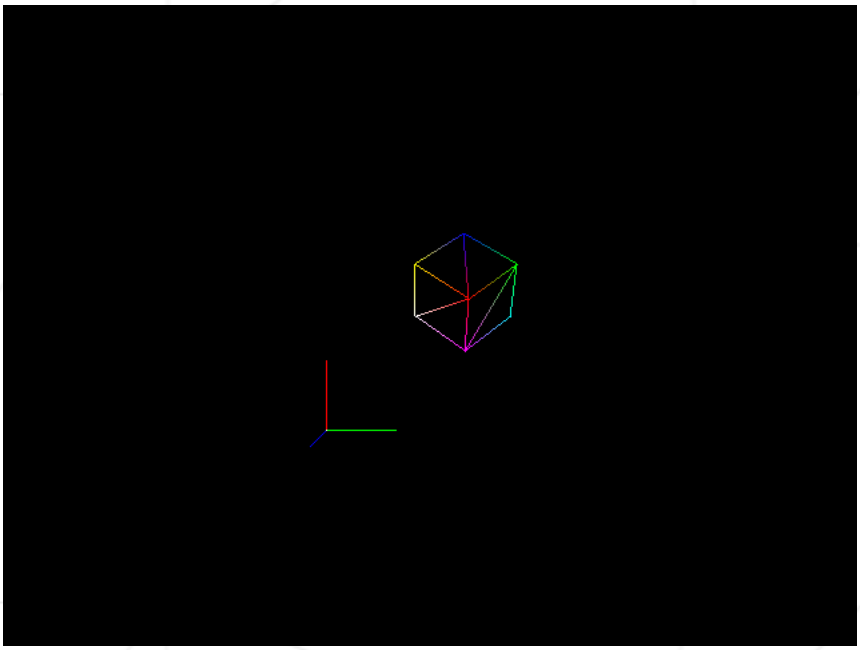
Exercise 05: The Triangle and render classes

	Exercise 05
The Triangle and render classes	
Turn-in directory : *ex*05/	
Files to turn in : `Render.class.php, Render.doc.txt, Triangle.class.php, Triangle.doc.txt, *.class.php, *.doc.txt`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library and the GD library.`	
Notes : `n/a`	

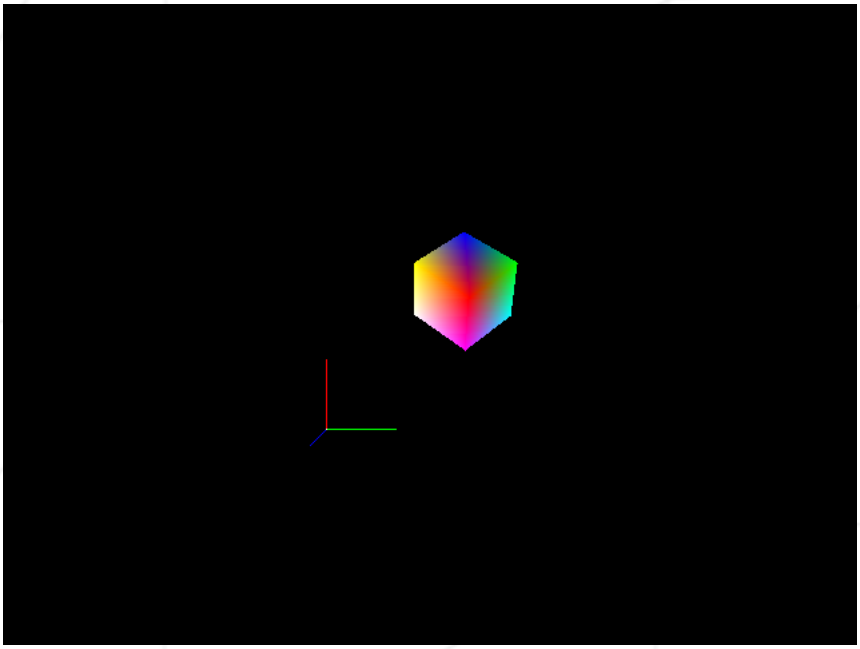
In this exercise, we are finally going to generate an image of our scene. The objective is to be able to render according to 3 modes:

- Vertex :

- Edge :



- Raster :



The generated image must be in `png` format. For this, all the `GD` library of `PHP` is at your disposal.

The implementation of this exercise will be extremely free. You must write a `Triangle` Class as well as `Render` Class. You can add new classes and/or change the classes of the previous exercises. You have to respect the following instructions.

Regarding the `Triangle` Class:

- The `Triangle` Class' constructor is waiting for an array. The following keys are required:

  'A': Vertex of the first point of the triangle, mandatory.

  'B': Vertex of the second point of the triangle, mandatory.

  'C': Vertex of the third point of the triangle, mandatory.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string (in this specific case the documentation doesn not have to be long). The content of the documentation must be read from a `Triangle.doc.txt` file and is left to your discretion as stipulated in previous exercises.
- There are no mandatory methods for your `Triangle` Class. However writing a few of them can be useful. For example I'm thinking about being able to iterate or map the vertices of the triangle, being able to iterate or map the edges of the triangle, be able to sort the vertices or the edges under certain conditions, etc..

Regarding the `Render` Class:

- The `Render` Class' constructor is waiting for an array. The following keys are required:

  'width': The generated image's width, which is mandatory.

  'height': The generated image's height which is mandatory.

  'filename': Filename in which the `png` image created will be saved, which is mandatory.

- Your `Render` Class must propose three Class constants: `VERTEX`, `EDGE` and `RASTERIZE` that will be used to select the rendering mode.

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.

- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.

- The Class must have a static method called `doc` that returns the documentation of the class in a string (in this specific case the documentation doesn not have to be long). The content of the documentation must be read from a `Render.doc.txt` file and is left to your discretion as stipulated in previous exercises.

- Your `Render` Class must have the following methods:

  void renderVertex(Vertex $screenVertex) : Displays a "screen" coordinates vertex in the generated image (a pixel basically).

  void renderTriangle(Triangle $triangle, $mode) : Displays a "screen" coordinates triangle in the generated image according to the selected mode. The mode must be one of the previoulsy defined Class constants.

  void develop() : Save the `png` generated image on the hard drive using the filename given to the constructor.

To simplify your work, I advise you to add some very simple features to some of the classes of the previous exercises or to add your own classes. This is not mandatory of course. Here are a few ideas without being exhaustive:

- A `toPngColor` method for your `Color` Class for which pseudo code would be the following using `GD` library:

```
color = getColorAlreadyAllocatedInPNGImage( img, r, g, b )
IF color == -1
    IF numberOfColorsInPNGImage( img ) >= 255
        color = getPNGImageClosestColor( img, r, g, b )
    ELSE
        color = allocateNewColorInPNGImage( img, r, g, b )
RETURN color
```

- Triangle implementation in the `Matrix` and `Camera` Class.

- An additional `bool isVisible( Triangle $tri )` method in the `Camera` Class to determine if a triangle is visible or not to avoid displaying the rear faces of an object. You can use the z-buffer or a BSP if you want but there is a much shorter and simpler algorithm you could implement. Search "backface culling".

- An additional `Mesh` Class to represent a 3D model composed of triangles to allow an easy manipulation of vertices, edges and triangles of the model. Of course, adding the support of this method to the `Matrix` and `Camera` Classes would be extremely useful!

- ...

You will find an example of the use of this class in the file `main_05.php` (attached) used to generate the three previous images.

Chapter XI

Bonus exercise 06: The Texture class

	Exercise 06
The Texture class	
Turn-in directory : $ex06/$	
Files to turn in : `Texture.class.php, Texture.doc.txt, *.class.php, *.doc.txt, vos fichiers de textures`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library and the GD library.`	
Notes : `n/a`	

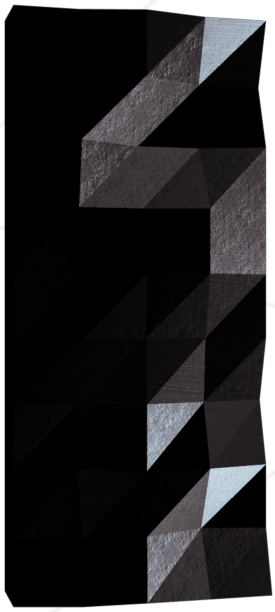
This exercise if for the strenuous coders. For those of you who wants to score more than 100 for this project. You must implement textures to your rasterizer. You can add and modify everything you want.

Please respect the following instructions:

- The Class must include a Boolean static attribute called `verbose` to control the displays related to the use of the Class. This attribute is initially `False`.
- If and only if the static attribute `verbose` is true, then the Class constructor and destructor will produce an output. See example output for formatting.
- The Class must have a static method called `doc` that returns the documentation of the class in a string (in this specific case the documentation doesn not have to be long). The content of the documentation must be read from a `Texture.doc.txt` file and is left to your discretion as stipulated in previous exercises.

To help you in your research, the useful concepts here are the "`u` and `v` vertex' coordinates" as well as barycentric coordinates of triangle ...

Coffee is on me for the first one that succeeds. - Thor



PHP Piscine

Day 07

Staff 42 piscine@42.fr

Summary:
This document is the day07's subject for the PHP Piscine.

Contents

Chapter I

Foreword

"Never forget what you are, the rest of the world will not. Wear it like armor and it can never be used to hurt you."

Tyrion Lannister

"Thanks to Game of Thrones, I have learned about inheritance in PHP and to make love with my sister. Thank you 42!"

Sylvain Dermy

This day will be easier if you have read the 5 volumes of the serie A Song of Ice and Fire, but in any case, this day is easy.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : Short and proud

	Exercise 00
Turn-in directory : *ex*00/	
Files to turn in : `Tyrion.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

Houses `Lannister` has always been populated with proud, blond people of reasonable height. That was until the arrival of `Tyrion Lannister,` who remained of the height of a child, even when he reached the age of an adult .

Create the `Tyrion` Class, inheriting from `Lannister` and containing only one method, so that the `test.php` file (attached) will produce the following output:

It must be possible to modify the strings in `test.php`. The modifications must update the output without modifying the `Tyrion` Class.

```
$> php -f test.php
A Lannister is born !
My name is Tyrion
Short
Hear me roar!
$>
```

Chapter IV

Exercise 01 : Words of honor

	Exercise 01
Turn-in directory : $ex01/$	
Files to turn in : `Greyjoy.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

Each great House has a motto. Using that motto freely would be a dreadful decision. The consequences could be tragic.

Write the `Greyjoy` Class so that the `test1.php` (attached) produces the following output:

```
$> php -f test1.php
We do not sow
$>
```

... and that the `test2.php` file (also attached) produces a fatal error.

Chapter V

Exercise 02 : Fireproofing

	Exercise 02
Turn-in directory : $ex02/$	
Files to turn in : `Targaryen.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

The `Targaryens`, contrary to the legend, are not immune to fire. Except, apparently, one ...

Write the `Targaryen` Class so that the `test.php` file (attached) produces the following output:

```
$> php -f test.php
Viserys burns alive
Daenerys emerges naked but unharmed
$>
```

Chapter VI

Exercise 03 : Playing house

	Exercise 03
Turn-in directory : *ex*03/	
Files to turn in : `House.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

The great Houses of `Westeros` present themselves in the same way. But they each have a motto (or heraldry) that is unique.

Write the `House` Class so that the `test1.php` (attached) produces the following output:

```
$> php -f test1.php
House Stark of Winterfell : "Winter is coming"
House Martell of Sunspear : "Unbowed, Unbent, Unbroken"
$>
```

... but that the `test2.php` (also attached) produces a fatal error. (The guy who wrote that test was clearly on the wrong show!)

Chapter VII

Exercise 04 : His sister ? Seriously ?

	Exercise 04
Turn-in directory : *ex*04/	
Files to turn in : `Lannister.class.php, Jaime.class.php, Tyrion.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

Contrary to what `Sylvain Dermy` might think, to be in love with one's own sister is not something highly recommended. It's funny because `Jaime Lannister`, also thinks that his sister is hot, contrary to his "little" brother `Tyrion`.

So write the `Lannister`, `Jaime` and `Tyrion` Class so that the `test.php` file (attached) produces the following output:

```
$> php -f test.php
Not even if I'm drunk !
Let's do this.
With pleasure, but only in a tower in Winterfell, then.
Not even if I'm drunk !
Let's do this.
Not even if I'm drunk !
$>
```

Chapter VIII

Exercise 05 : Winter is coming

	Exercise 05
Turn-in directory : *ex*05/	
Files to turn in : `NightsWatch.class.php, IFighter.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

The `Night Watch` is the last Wall that protects Westeros from annihilation. Even if no one seems to remember that part...

The situation is what it is. The `Night Watch` is forced to recruit whomever they can attract, whether the rare ones who see it as an honor, or the criminals who are looking for a way out.

However, there would be no logic in using non fighters in battle!

You need to write the `IFighter` as well as the `NightsWatch` Class, so that the `test1.php` file (attached) produces the following output:

```
$> php -f test1.php
* looses his wolf on the enemy, and charges with courage *
* flees, finds a girl, grows a spine, and defends her to the bitter end *
$>
```

And naturally, the `test2.php` (also attached) produces a fatal error.

Chapter IX

Exercise 06 : The wrong kind of pact

	Exercise 06
Turn-in directory : *ex*06/	
Files to turn in : `UnholyFactory.class.php, Fighter.class.php`	
Allowed functions : `Everything learned since the beginning of the Piscine as well as the entire standard PHP library.`	
Notes : `n/a`	

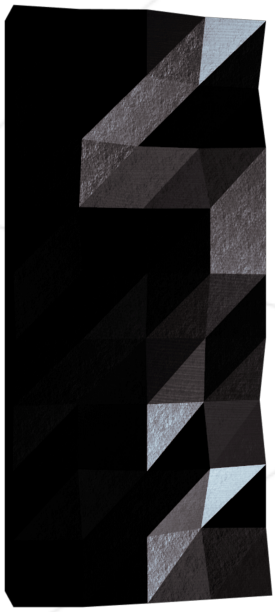
After many unsuccessful efforts to build an army, `Stannis Baratheon` is facing the abyss. No army. No throne. And without a throne he is nothing.

Similar circumstances have pushed some men into making deals with powers that they do not completely control and its precisely what he did, otherwise he would have never attacked `King's Landing` with an army that seemed to be never-ending...

You need to write the `Fighter` and `UnholyFactory` Classes, so that the `test1.php` file (attached) produces the following output:

```
$> php -f test1.php
(Factory absorbed a fighter of type foot soldier)
(Factory already absorbed a fighter of type foot soldier)
(Factory absorbed a fighter of type archer)
(Factory absorbed a fighter of type assassin)
(Factory can't absorb this, it's not a fighter)
(Factory fabricates a fighter of type foot soldier)
(Factory hasn't absorbed any fighter of type llama)
(Factory fabricates a fighter of type foot soldier)
(Factory fabricates a fighter of type archer)
(Factory fabricates a fighter of type foot soldier)
(Factory fabricates a fighter of type assassin)
(Factory fabricates a fighter of type foot soldier)
(Factory fabricates a fighter of type archer)
* draws his sword and runs towards the Hound *
* draws his sword and runs towards Tyrion *
* draws his sword and runs towards Podrick *
* draws his sword and runs towards the Hound *
* draws his sword and runs towards Tyrion *
* draws his sword and runs towards Podrick *
* shoots arrows at the Hound *
* shoots arrows at Tyrion *
* shoots arrows at Podrick *
* draws his sword and runs towards the Hound *
* draws his sword and runs towards Tyrion *
* draws his sword and runs towards Podrick *
* creeps behind the Hound and stabs at it *
* creeps behind Tyrion and stabs at it *
* creeps behind Podrick and stabs at it *
* draws his sword and runs towards the Hound *
* draws his sword and runs towards Tyrion *
* draws his sword and runs towards Podrick *
* shoots arrows at the Hound *
* shoots arrows at Tyrion *
* shoots arrows at Podrick *
$>
```

And naturally, the `test2.php` file (also attached) produces a fatal error.



PHP Piscine

Day 08

Staff 42 piscine@42.fr

Summary:
This document is the day08's subject for the PHP Piscine.

Contents

Chapter I

Foreword

Discover Warhammer 40000.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject ...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Awesome Starships Battles In The Dark Grim Future Of The Grim Dark 41st Millenium

	Exercise 00
Awesome Starships Battles In The Dark Grim Future Of The Grim Dark 41st Millenium	
Turn-in directory : $ex00/$	
Files to turn in : `*`	
Allowed functions : `Everything. Yes everything. Yes even Javascript if you know how to use it. yes even your framework that knows how to brew coffee.`	
Notes : `n/a`	

III.1 No, you are not dreaming

With today's videos, you have just about gone round the whole object syntax of `PHP`. But in reality, you have had to write only about 15 classes that are dueling without any real purpose since the exercises were dictating to you the classes to create and what to put in them. Today, you will have to decide for yourself what classes to create and how to make them interact.

We are not trying to assess today wethere you are able to write a class, an interface or a trait. Nope. Today, we want to determine if you are able to implement what you have learned. Which is why instead of having the usual daily tests, you will have only one exercise for which you will be free to develop and implement your program however you see fit.

Why authorize everything? Because `OOP` isn't a technology. It's a way of coding and you must be able to make it your own and code no matter what the technology is. If you are drowning in one or more extremely complex frameworks for today, or in the labyrinth of JavaScript, it is your problem. We want a program that works with a relevant `OOP` use. The rest isn't our problem today.

III.2 Instructions

- This is a PHP project. Don't give us Python, Ruby or anything else since you think you are allowed "everything". Not only, it isn't funny and it will be worth 0.
- Your program needs to work on Chrome with the version installed on the iMacs.
- A minimal effort is required on the aesthetic front. Use CSS and some images.
- The one or multiple url of your application are up to you.
- No matter which technology you choose, it is your responsibility that during your defence everything required is available on your repository and on your computer and you will not be allowed to push or install 30Gb of framework.
- Each d06, d07 and d08 video presents a concept. Each of these concepts **MUST** be presented at least once in your submitted project, with the exception of reflection API that you may choose to use or not. On top of which the use of each of these concept **MUST** be relevant. Your corrector could refuse to give you any points for a concept that you used badly or that is useless. The syntax of object of JavaScript obviously **does not** count in this account.
- One unique Class per file only.
- One file that contains the definition of a class cannot contain any other, except for `require` or `require_once` if necessary.
- A file containing a class must **ALWAYS** be named `ClassName.class.php`.
- A class must **ALWAYS** be accompanied by a documentation file whose name **MUST** be `ClassName.doc.txt`. It will not be a mistake to copy/paste d06.
- A Class documentation must **ALWAYS** be useful and match the implementation.
- A class must **ALWAYS** have a static method called `doc` returns the documentation of a class into a string.
- An attribute or public method that wasn't needed will result in you being graded 0 for the day. Be clever with the visibility and prove that you know how to use it.
- If we cannot say tha one of your child Classes "is a" parent Class in an (`extends`) inheritance, your concept is wrong. You will then be graded 0 for the day. Be meticulous.
- This is more an advice than a recommendation: big and incomplete is bad. Small and complete is better.

III.3 The subject!

You will have to write a turn by turn space combat game using the `Warhammer 40000` universe (SciFi Universe invented by `Games Workshop`). The rules are simple but not trivial to have minimal fun. For those who know, the rules will be freely inspired by the game `Dreadfleet` from the same company. Actually all rights reserved to `Games Workshop` etc.

`Awesome Battleships Battles` is a 2 players game that allow them tocontrol a fleet of awesome spaceships that fight it out for a very good reason that you are free to invent. The goal is to eliminate the adversary fleet first.

III.3.1 The dice

- The game uses ordinary 6 faces dice that we will refer as `D6` to make it short from now on.
- To throw one die is shortened with `1D6`. Throwing a 3 is `3D6`. Etc.
- Obtaining `3`, `4`, `5` or `6` on `1D6` is shortened with `3+`. Etc.

III.3.2 The game zone

- The game zone is a grid of 150 by 100 cells.
- Both enemy fleets start from opposite corners and all the spaceships are stationary.
- There must be a few obstacles (asteroids or space stations for example). Up to you which obstacles you include. The goal is to block the movements of the spaceships and to break the lines of fire to encourage maneuvers. The board cannot be filled with obstacles. 5 or 6 obstacles of about 10 cells each will do. You can of course adapt these values to your liking.
- The position and the size of the obstacles do not have to be random if you do not wish to spend more time on it.
- A spaceship that is out of bounds or that bumps into an obstacle for any reason is eliminated.

III.3.3 The turns

- Each turn, players play with one of their ships one after the other until all ships have played.
- When one plays a ships it's called "activated".
- We can only activate one ship once per turn.
- Activation is irreversible.
- An active ship must realize three mandatory phases and always in this order:
  - Order phase
  - Movement phase
  - Shoot phase
- Once all the spaceships of both players have been activated, a new turn starts.

III.3.4 Spaceships

- A spaceship is defined by a few features

  Name : The name of the vessels **MUST** be badass, like "Wrath Of The Righteous" "Rightful Vengeance" or "Smite Of Terra" for the imperial ones, "Megacrusha" for Orks, or even "Bane Of All Hope" for Chaos for example.

  Size : Define a width and lengths in cells of a Spaceship. A Spaceship 3x10 is a very big admiral ship. A ship 1x2 is a small light one, a "scout". An average one will be 1x4 cells.

  Sprite or equivalent : The representation of the ship on the grid.

  Hull points : life points of the ship. If those fall to `0`, the ship is destroyed. 5 points is a good average for a mid-sized ship.

  Engine power : The engine power gives the ship a number of points that the players will be able to attribute to different actions when activating a ship depending on situations. Those will be "power points" shortened with `PP`. Those points can be spent to make the ship go faster, strengthen its shields or use its weapons. This attribution is done during the order phase which will be detailed later. 10`PP` represents an average for a basic ship and will be the most common value. The biggest vessels can go up to 15`PP`.

  Speed : Maximum number of cells that the ship can move each turn. This specificity can be raised with the support of `PP`. A scout who is faster, can move up to 20 cells. A big ass one can only go 10.

  Handling: Number of cells that a ship that moved on the prior turn needs to travel straight this turn if he wants to stay stationery for the next one (inertia really). It's also the minimum number of cells that a ship can go straight

before he can turn to the right or to the left AND between each turn. A stationary ship can make a free turn before starting to move at the beginning of the movement phase. A light and nimble scout will have a handling of 2 or 3. A big ship will have a handling of 5 for example.

Shield: Number of damage points that a ship can endure before losing his hull points. Worth `0` upon activation of the ship and can receive `PP` points.

Weapons: List of weapons that each ship owns, generally one or two, sometimes more for the really big ships. Each weapon needs `PP` to function. Each `PP` attributed to a weapon will allow it to raise its efficiency shooting factor for that turn. A section will be dedicated to the weapons and their specificities.

Some weapons will be able to have some special bonuses that will modify their specificities or action range.

III.3.5 Les phases

The activation of a spaceship will give access to 3 different phases that must always be played in this order:

The order phase

At the beginning of this phase, all the `PP` spent on the prior turn are back to zero. Which means that the weapons systems, shields and speed will be back to their initial values.

The player will now spend the `PP` of the ship on one or the other of the ship's systems according to the situation. He can spend all its points, some of them or none.

- `1PP` spent on speed allows to move `1D6` more space.
- `1PP` spent on shields give `1` shield point.
- `1PP` spent on weapons gives `1D6` more to shoot with it.

The players can also spend his `PP` to fix his ships. To repair a ship, the ship must be stationary. Each `PP` spent in repair allows to roll `1D6`. On a `6`, a hull point is restored up to the maximum starting value.

The movement phase

During this phase, a ship can move.

- We define a turn as "turning the ship 90 degrees to the right or to the left".
- The ship will rotate around the most central cell that compose that ship.
- A stationary ship can do a free turn before he starts to move.*
- A ship can only move at maximum the number of cells matching its given speed (with the additional `PP` spent on speed for the turn).
- A ship must always move to an equal or superior amount of cells matching its given handling.
- If during the last turn a ship has moved exactly the amount of cells corresponding to its given handling, the ship can stay stationary this turn.
- A stationary ship can stay that way as long as wanted.
- A Stationary ship can move a number of cells inferior to its given handling. It's the only case where it is allowed to move less cells that its given speed. If it does that the ship won't be considered stationary on next turn.
- A ship can make a turn each time it has moved a bigger or equal number of cells that its given handling.
- A ship that hits another one will immediately stop and can no longer move or shoot for that turn. It will be considered stationary at the beginning of the next turn. Furthemore, if it moved a number of boxes strictly greater than its given handling, each ship will receive a number of damage points equal to the number of hull points that the other ship had prior to impact. Those damage points can be absorbed by shield points. In that case we are talking about "buccaneering". It's a risky maneuver, quite desperate really but it can lead to some spectacular situations.
- A ship that hits an obstacle or that goes out of bounds is eliminated and considered destroyed.

Shooting phase

It's the phase where serious stuff can start. Imagine spaceships many kilometers wide, shooting each other with weapons the size of buildings... Special effects guaranteed.

Every weapons have a specific profile defined as such:

Charge: Initially `0` upon activation of the Spaceship. Each `PP` spent on that weapon for that turn adds 1 charge point. Each charge point gives `1D6` for the shooting with that weapon. Some weapons have a number of charge points by default always available that allows to shoot even if no `PP` was used for that weapon for that turn.

Short Range: Number of cells the weapon can reach short range.

Middle Range: Number of cells the weapon can reach middle range.

Long Range: Number of cells the weapon can reach long range. Also maximum range for the weapon to be used..

Effect Zone: Description of the cells on which the weapon can shoot..

To shoot, a ship must have a clear view of its target. Any ship or obstacle can block that target view. To check the target view, we need to be able to trace a line between the shooter and its target without any obstacle. If the shooter has acquired a clear target, the ship will throw the dice equal to the number of charge points the weapons has. Of course the target must be within its effect zone and its weapon range. The dice obtaining at least able specific value are considered "a success". The basic value to obtain to win are as follows:

Short range: `4+`

Middle range: `5+`

Long range: `6`

Each win will provoke a damage point on one or many targets. The damage points are first deducted to the target's shield, then to its hull points.

Each weapon can only shoot once per turn. Of course, a ship can decide not to shoot. Furthemore some weapons can force the ships to be stationary to be able to shoot. Those weapons are usually powerful long range guns. Some weapons are also able to modify the value to obtain from `1D6` to provoke success.

A shoot that reaches its target from the front or the back provokes an "enfilade shoot". An enfilade shoot will pass through a ship lengthwise bringing on catastrophic damages. To represent that case, a ship that shoot an enfilade shoot will reduce by `1` the value required on `1D6` to obtain success.

III.3.6 Weapons examples

- Side laser batteries

  Charge: 0

  Short Range: 1 to 10 cells

  Middle Range: 11 to 20 cells

  Long Range: 21 to 30 cells

  Effect Zone: The ship's width for the first cell, plus 1 width cell more at the front and back of the ship per cell away from the ship. The ship can choose to shoot either from the left or from the right at each use. The drawing will be more clear. The 'X' represent the ship and the '.' the effect zone.

```
..........
........
......
....
XXXX
```

- Nautical lance

  Charges : 0

  Short Range: 1 to 30 cells

  Middle Range: 31 to 60 cells

  Long Range: 61 to 90 cells

  Effect Zone: A straight line or column 1 cell wide that start from the front of the ship.

- Heavy nautical lance

  Charges : 3

  Short Range: 1 to 30 cells

  Middle Range: 31 to 60 cells

  Long Range: 61 to 90 cells

  Effect Zone: A straing line or columns of 1 cell wide that start from the front of the ship.

  Special: The shooter must be stationary to be able to shoot. Furthemore, as long as the shoot destroys its target, the dice can be thrown again to attempt to destroy a target located behind the original one as long as the maximum range of the weapon isn't reached.

- Close range super heavy automatic weapon

  Charges : 5

  Short Range: 1 to 3 cells

  Middle Range: 4 to 7 cells

  Long Range: 8 to 10 cells

  Effect Zone: Any cell within range.

- Macro canon

  Charges : 0

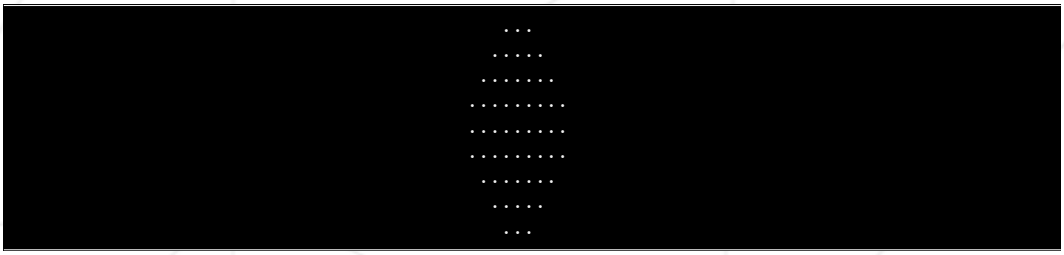
  Short Range: 1 to 10 cells

  Middle Range: 11 to 20 cells

  Long Range: 21 to 30 cells

  Effect Zone: A straing line or columns of 1 cell wide that start from the front of the ship.

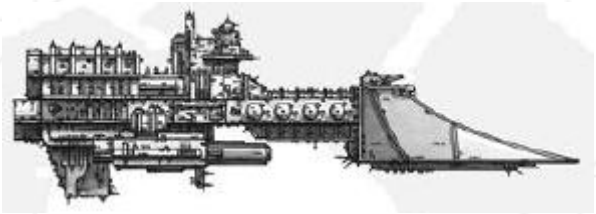
  Special : The explosion of the amo reaches multiple boxes. The center of the explosion is located on the closest target's hit cell from the shooter. The explosion covers "a circle" of 9 cells:

  Each target covered, even partially by the "circle" receives a number of damage points matching the dice throw. Useful to vaporize the fleet of fast little scouts.

III.3.7 Ships examples

- Imperial Frigate



Name: "Honorable Duty"

Size: 1x4 cases

Hull points: 5

PP: 10

Speed: 15

Handling: 4

Shield: 0

Weapons: Side laser batteries

- Imperial Destroyer

Name: "Sword Of Absolution"

Size: 1x3 cases

Hull points: 4

PP: 10

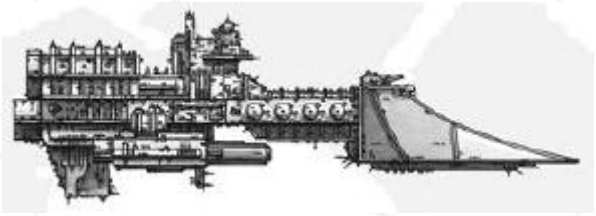
Speed: 18

Handling: 3

Bouclier : 0

Weapons: Side laser batteries

- Imperial Frigate



Name: "Hand Of The Emperor"

Size: 1x4 cases

Hull points: 5

PP: 10

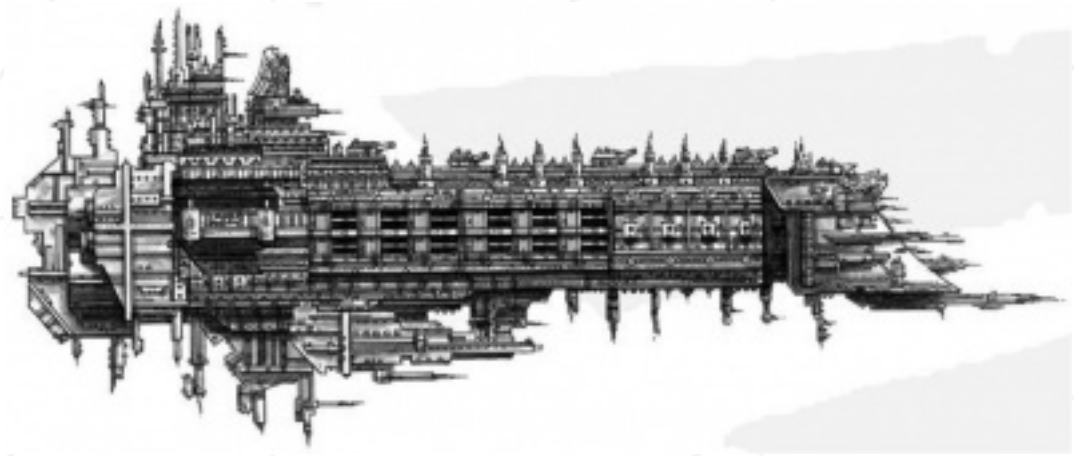
Speed: 15

Handling: 4

Bouclier : 0

Weapons: Nautical Lance

- Imperial Ironclad



Name: "Imperator Deus"

Size: 2x7 cases

Hull points: 8

PP: 12

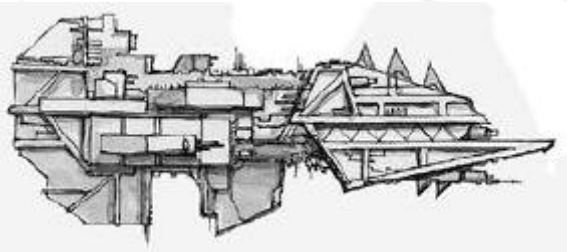
Speed: 10

Handling: 5

Bouclier : 2

Weapons: Two Nautical Lances

- Onslaught Attack Ship



Name: "Orktobre Roug'"

Size: 1x2 cases

Hull points: 4

PP: 10

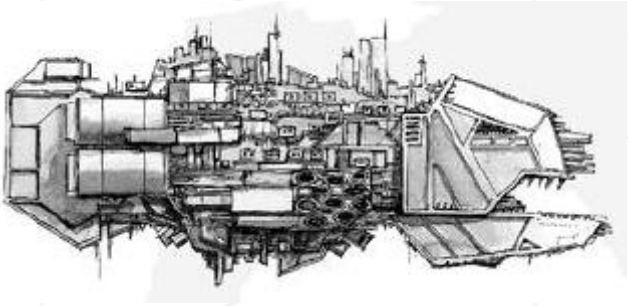
Speed: 19

Handling: 3

Bouclier : 0

Weapons: Side laser batteries

- Terror Ship



Name: "Ork'N'Roll !"

Size: 1x5 cases

Hull points: 6

PP: 10

Speed: 12

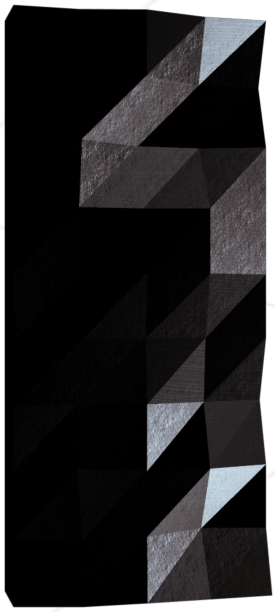
Handling: 4

Bouclier : 0

Weapons: Close range super heavy automatic weapon - Makro Kanon

III.4 Additional comments

- Discuss with your peers to get their opinion of the best way to use this or that notion of OOP in PHP for this exercise. The composition of this game is perfect for it..
- Prioritise a limited – but playable version - of the game rather than being tempted to do it all and finish nothing. Your objective is to practice OOP, not to code an extraordinary game (Even if we'd be happy if you did that).
- To have two identical ships that can move on an empty grid and be shot at with an identical weapon is the strict minimum.
- An average game will see 2 fleets of equal strength composed of 5 to 10 ships each.
- Don't hesitate to post on the forum your weapon and/or vessels proposals. The Warhammer 40000 world has a lot of terrible factions and has even more terrible and numerous weapons. This way you will raise easily that way the level of your game.
- You can modify some parameters of the game as long as they do not modify the nature of the rules and Warhammer 40000's universe.
- The provided illustrations come from http://wh40k.lexicanum.com/. All rights reserved to all concerned etc.
- For the Emperor.



PHP Piscine

Day 09

Staff 42 piscine@42.fr

Summary:
This document is the day09's subject for the PHP Piscine.

Contents

Chapter I

Foreword

Gollum : What has he done? Stupid Hobbit! You are damaging them.

Sam : Damaging what? They‘ve got almost no meat. What we need are potatoes.

Gollum : What are potatoes my precious? What are they?

Sam : Potatoes you idiot! Broiled, mashed, fried, sliced or boiled with sauce. Nice fat sliced and fried in oil to go with fried fish. Even you won’t resist.

Gollum : Oh but we will resist. To waste perfect fish like that... We prefer fish raw and alive, keep your nasty fries.

Sam : You are impossible.

Chapter II

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- Using a library is forbidden and will be considered cheating. Cheaters get `-42`, and this grade is non-negotiable..
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- By Odin, by Thor ! Use your brain !!!

Chapter III

Exercise 00 : Come blow this balloon

	Exercise 00
Come blow this balloon	
Turn-in directory : *ex*00/	
Files to turn in : `balloon.html`	
Allowed functions : `HTML, CSS, JS`	
Notes : `n/a`	

For this exercise, you will blow a balloon. To start, your HTML file will have a basic structure similar to the one seen in the introduction video. No library allowed.

Create in HTML/CSS a `div` `200px` by `200px` with a red background color. The borders will be rounded to create a perfect round shape and not a square anymore. That `div` will then be our balloon.

When a click occurs on the balloon, its size will grow by `10px` while keeping its round shape. With each click its color will go in that order from red to green, then blue and back to red again.

Usually, this type of balloon is rather resistant but if its size becomes greater than `420px` it will explode and return to its original size.

Small additional detail, when the mouse is in the balloon and leaves it, the size of the balloon shrinks by `5px` (the size of the balloon cannot go lower than `200px`) and its color changes in the reverse order than mentioned earlier.

Chapter IV

Exercise 01 : It's over 9000

	Exercise 01
It's over 9000	
Turn-in directory : $ex01/$	
Files to turn in : `calc.html`	
Allowed functions : `HTML, CSS, JS`	
Notes : `n/a`	

For this exercise, we will create a basic calculator. To start, your HTML file will have a basic structure similar to the one seen in the introduction video. No library allowed. The design doesn't matter as long as the exercise is still doable.

First, let's model our calculator. It will be composed as follows:

- A `text input` field that represents the left member of our calculation.
- A `select` field that will contain a list of the following operators as options:(`'+'`, `'-'`, `'*'`, `'/'`, `'%'`).
- A `text input` field that represents the right member of our calculation.
- A `submit input` of value `'Try me!'`.

When we click on the button `'Try me!'`, the calculation is executed and the result appears in an alert message. The result also needs to be displayed on the console of your browzer (log).

Both inputs fields can only contain positive integer values($>= 0$) for the calculation to be executed. Otherwise display an alert message must appear with the following message `'Error :('`.

Division and modulo by zero display an alert message with the following message: `"It's over 9000!"`. The result also needs to be displayed on the console of your browzer (log)

Every 30 seconds, an alert pop-up windows will appear saying `'Please, use me...'`.

Chapter V

Exercise 02 : To do or not to do

	Exercise 02 :
To do or not to do	
Turn-in directory : $ex02$: /	
Files to turn in : `index.html, todo.js`	
Allowed functions : `HTML, CSS, JS`	
Notes : `n/a`	

For this exercise, we will have to create a mini local task management. To start, your HTML file will have a basic structure similar to the one seen in the introduction video. The design does not matter as long as the structure presented below is respected. Be creative but concentrate in priority on features.

The to do list will be represented by a `div` that will have as `id` attribute the value `'ft_list'`. This bloc contains the list of "TO DO". Each TO DO is represented by a div contained in the `'ft_list'` bloc. When a TO DO is created, it is placed at the top of the list. Up to you to create the element and place it in the right spot (DOM manipulation).

There must be creation button named `'New'`. When clicked, it'll open a text window (checkout the promp function) that will allow the user to fill in a new TO DO. Once validated if not empty it must appear at the tol of the list.

To remove a TO DO from the list, all you have to do is click on it. A configuration window must open and ask whether yes or no you want to remove that TO DO. If you confirm, the TO DO must disappear permanently from the DOM, it can't just be hididen.

Small additional implementation, your TO DO list will have to be saved as a cookie. If the list contains some TO DO when you close you browser, this same list must be loaded and displayed in `'ft_list'`. If the cookie(s) do not exist, then the list will be blank.

Chapter VI

Exercise 03 : If jQuery, I'm going too

	Exercise 03 :
If jQuery, I'm going too	
Turn-in directory : $ex03$: /	
Files to turn in : `Same files than respective exercises`	
Allowed functions : `HTML, CSS, jQuery`	
Notes : `n/a`	

It's time to use that tool we love so much. You need to redo the first 3 exercises using lib jQwery. Import this lib via CDN as explained in the elearning section. No need to download the lib in your repository otherwise you will lose some points.

For the repository, create 3 disctinct folders: `ex00bis/`, `ex01bis/` et `ex02bis/`

Only the jQuery lib is allowed for these exercises.

Chapter VII

Exercise 04 : AJAX, stronger than dirt!

	Exercise 04 :
AJAX, stronger than dirt!	
Turn-in directory : $ex04$: /	
Files to turn in : `index.html, todo.js, select.php, insert.php, delete.php, list.csv`	
Allowed functions : `HTML, CSS, jQuery, PHP`	
Notes : `n/a`	

Now that you are a bit more comfortable with jQuery, it's time to throw yourself in the (bleach) bath! The school storage is great but the Cloud is better.

And as luck would have it, your Cloud won't be far. It will be on your localhost server that will execute the phpt.

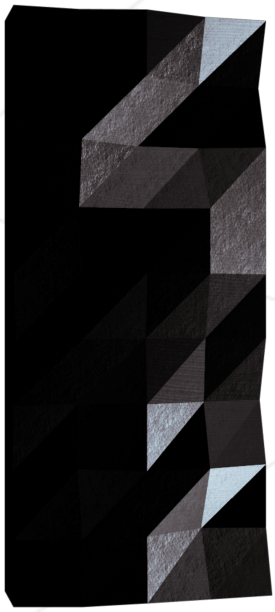
Therefore you will need to replace in the previous exercise the backup system so that everything will be functional with an "online" CSV backup and no longer via cookies. The data formatting inside the CSV must be the following `'id;i am a todo'`.

It is MANDATORY that you use AJAX, your HTML page should never be refreshed.

- the `select.php` file gets the TO DO list from the CSV.
- the `insert.php` file adds a TO DO to the CSV.
- the `delete.php` file removes a TO DO from the CSV.

Any action on the page related to your list (adds, remove) will have to be transfered to the CSV using AJAX calls.

Good Luck !



Rush0 of the PHP Piscine

ft_minishop

Staff 42 piscine@42.fr

Summary: This rush is here to help you set-up a mini e-commerce website and make you practice all the basics seen this week. But before anything, you will need to set up your own data management system.

Contents

Chapter I

Foreword

Roger : All right, I have everything we need.

Roger : Chocolate milk, cheese crisps, and some speed I just bought in the bathroom.

Steve : Wait, Roger, Are your sure it's a good idea?

Roger : I have never been more sure of anything in my life.

Roger : You see, everything is about moderation, now, I am going to have 5 crisps and not one more.

Steve : "One Leg", please, drive on the proper side of the road.

Roger : Why are you calling me “One Leg”, do you have a problem with my legs?

Roger : Is that why there’s a wheelchair in the car?

Roger : What have you done with my legs you fucking Nazi?

Klaus : Steeve, who is he talking to now?

Roger : And you, Garfield, why do you hate Mondays? You don’t even work!

Chapter II

General Instructions

The following recommendations are all part of the defence scale. Be very careful when you apply those as they are sanctioned by a grade of 0 with no appeal.

- In case there was be a doubt, this rush is of course to be created in `PHP`.
- `Javascript`, `HTML` et `CSS` are authorised.
- `HTML` templates are forbidden.
- `Javascript` libraries are authorized only for the bonus part.
- This project will be corrected by humans only.
- You have to handle errors carefully. (ex: no SQL should be possible).
- Object oriented programmation is forbidden (check out mysqli's procedural mode).
- Remember to maintain your own logic in the hierarchy and structure of your project.
- Anything not specifically authorized is therefore forbidden.
- You can ask your questions on the forum, on slack...
- Good luck to all for this rush!

Chapter III

Subject

III.1 Mandatory part

Following your first week of training on web, you will have to -in a weekend– create a mini e-commerce online shop. For those that don't know what an online shop is, please check the list here. Note: It's in french but most of the website listed have an english translation and it's about getting an overview of the design and features.

The tool you will use for your server from now on is PAMP, developed by 42. That one is still in beta, please help us make it better by bringing up the bugs you will encounter by ticket or on the forum.

Of course, we won't ask of you that it is as complete as some CMS generated websites, however, you will have some mandatory modules to develop.

An e-commerce website must be able to manage the following features:

- **Data management:** You need to implement, either your own data management (creation, modification and removal of data) in the format you want (ex: CSV), or using mysql. The data management is a key element to the design of your site.
- **User management:** You must be able to create and delete a user account. The login to your site is essential before one can validate your basket, however, it can be possible to fill it before being identified. To do that, look at PHP sessions.
- **A basket:** It must contain the list of products that your client wants to buy, the price and the quantity of each article, as well as the total cost. It must have a validation button to archive the order.
- **Categories and associated products:** As for any e-commerce website, it requires to be able to organize the products by categories. So, one category can contain multiple products, but one article can belong to multiple categories. It's up to you to conceptualize your data base before your start.
- **A landing page:** That page is the showcase of your website, it must be attractive and intuitive so that your clients will want to buy. For that, the page can contain

the connection section, the account creation, the basket and a preview of a few articles and categories. The organization of your modules will improve the user experience.

- **Administrative section:** In this section, we are asking come up with a way to give an exclusive access to some users. Those users will be able to manage the content of your site (add, modify, remove articles, categories and users)..

In summary, a user must be able to register, connect, add articles in his basket and validate his order. Once validated, the order must be visible from the administration section..

III.2 Bonus part

To allow you to go further in the development of your e-commerce website, we are proposing some bonus features. To take these bonus points into account, we are asking you to have a fully functioninal website and to obtain the grade of 75.

Here are a few ideas of bonuses that can be interesting:

- Stock management
- Itemisation of the articles (color, size, etc.)
- Advanced users info
- Captcha
- API
- Graphics/Statistics generation
- Printable invoice generation
- Registration/connection via Oauth

You can inspire yourself from numerous existing e-commerce websites to find more bonuses to implement. You are particularly free to create as long as the mandatory part is respected.

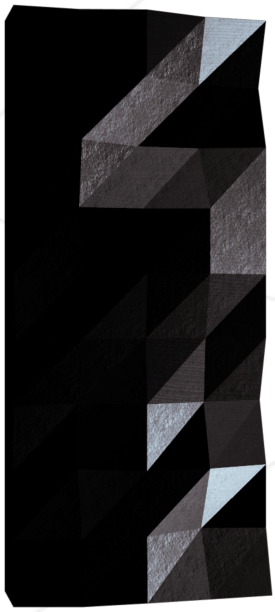
Chapter IV

Submission and peer correction

- You'll have to submit a file called `author` containing your usernames followed by a '\n' at the root of your repository.

```
$>cat -e auteur
xlogin$
ylogin$
$>
```

- Your website must contain at least the following files: `index.php` and `install.php`. The `index.php` file will be the entry point of your site, the `install.php` will create your database and everything you require to initialize your website.
- Only the work on your repository will be graded.
- The size of attached files (images, librairies) with the source files cannot exceed 5Mo.
- Any useless file will be penalized.



PHP Piscine

Rush1

Staff 42 piscine@42.fr

Summary:
This document is the rush01's subject for the PHP Piscine.

Contents

Chapter I

Foreword

What do all these films have in common?

- Terminator 2
- Aliens
- The Godfather Part II
- Mad Max 2
- Evil Dead 2
- The Dark Knight

Same as today's topic, they are all better than the first one.

Chapter II

Consignes

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / the Internet / http://www.php.net / ....`
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- By Odin, by Thor ! Use your brain !!!

Chapter III

Awesome Starships Battles II

III.1 The vengeance's Return

Do you want to create something better than D08? The time has come to show what you are truly capable of..

Scream if you're happy.

III.2 General Instructions

- As the final rush of this `PHP` Piscine, you are naturally allowed to use everything you want to do `PHP` web programming. The server's side must be in `PHP`.
- This is a `PHP` project. Don't give us `Python`, `Ruby` or anything else since you think you are allowed "everything". Not only, is it not funny it will also be worth `0`.
- Your program needs to work on `Chrome` with the version installed on the iMacs.
- Your game must be beautiful. With all the libraries available to you, a minimum of dynamism is required!
- The one or multiple url of your application are up to you.
- No matter which technology you choose, it is your responsibility that during your defence everything required is available on your repository and on your computer and you will not be allowed to push or install `30Gb` of framework.
- Only one unique Class per file.
- One file that contains the definition of a class cannot contain any other, except for `require` or `require_once` if necessary.
- A file containing a class must **ALWAYS** be named `ClassName.class.php`.
- A class must **ALWAYS** be accompanied by a documentation file whose name **MUST** be `ClassName.doc.txt`. It's still not a copy/paste error.

- A Class documentation must **ALWAYS** be useful and match the implementation. Rest assured that this point will be checked during peee2peer correction !
- The Class must **ALWAYS** have a static method called `doc` that returns the documentation of the class in a string.
- An attribute or public method that wasn't needed will result in you being graded `0` for the day. Be clever with the visibility and prove that you know how to use it.
- If we cannot say that one of your child Classes "is a" parent Class in an (`extends`) inheritance, your concept is wrong. You will then be graded `0` for the day. Be meticulous.
- This is more an advice than a recommendation: big and incomplete is bad. Small and complete is better.

III.3 The subject

- Get back to the day08's subject for the rules. Implement them meticulously, the grading will be based upon those.
- The ships must have enough space to maneuver. Keep in mind that on a 150x100 cells map a basic vessel should have a size of 1x4 cells and not one more. A vessel of 3x10 cells is an absolutely colossal ship in a fleet.
- A game will always be more interesting with a handful of obstacles of average size rather than a multitude of small ones. The ships must have the room to maneuver, take them from the back, etc.
- The weapons described in D08 must exist in the game.
- The gamers must be able to create an account and log in to play. Their profile will contain all the usual profile information as well as combat performance.
- There must be a lobby where players can chat and create games the other players can join in.
- Players must be ranked.
- Games must have a melee mode or by team. A game cannot host more than 4 players.
- There are no limits to the number of games than can happen simultaneously think `Battlenet`).
- Each side must be easy to identify in the game arena.
- When creating a game, one must be able to select a fleet "value". This value will be used by each player to select the ships he wishes to use amongst the vessels available for the faction he chose.

- 500 points represent a small fleet. 1500 points an average one and 3000 is a gigantic war armada.
- Therefore, each ship must have a value in points that represents its efficiency.
- A fleet must always belong to a single unique faction. You game must propose at least 3 factions owning each at least 3 ships.
- We will also greatly appreciate the presence of scripted games and campaigns recreating famous battles with legendary spaceships. This will be a bonus.
- We might organize a tournament in the weeks to come using some of the best games of the cohort. If you want your game to be selected, you will need to make us try it in the days that will follow defense. We'll then select one or more to start a tournament.
- Have a blast! For the Emperor!