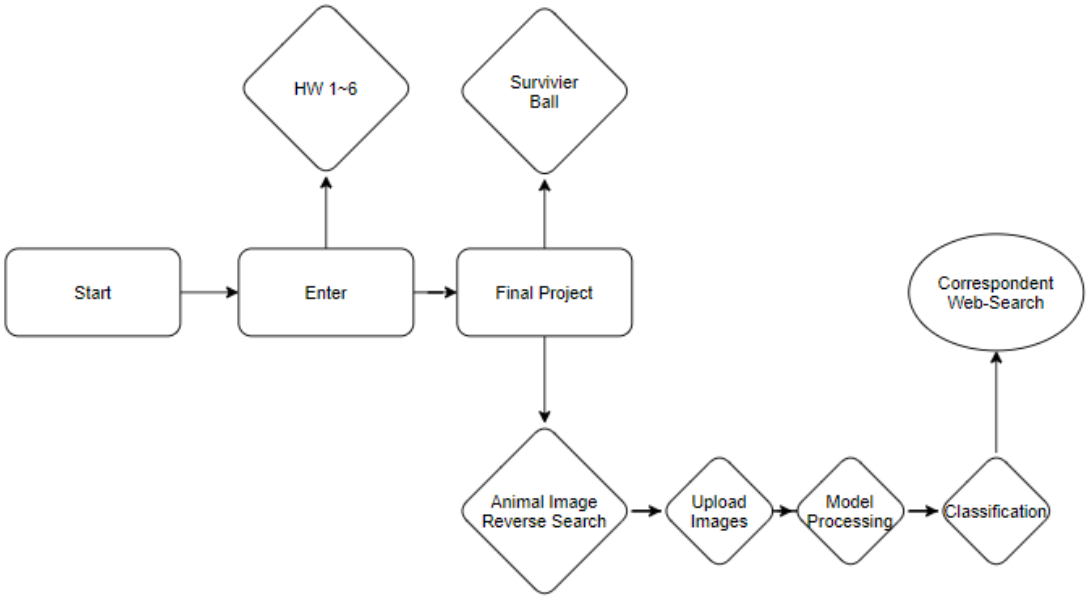
# JavaScript Web Programming Final Project Report

0612005 林政誠 0612020 吳峻陞

## 系統功能

創建具有多向性檔案連結的網站，主要新建有不同玩法且設計完善的遊戲 Survivor Ball 以及可以辨認圖片並協助查找的圖片搜尋引擎 Animal Image Reverse Search

## Processing Flow




## 系統開發

➔ Interface: Wampserver, Visual Studio Code, Notepad++, NodeJS, OS

➔ Code: JavaScript, html, css, php, json

## 程式列表與功能簡述：

1.start.html (index.php)/css ：以遮罩製造高級感

2. enter.html/css/js ：首頁，可跳轉至各網頁

3&4. hw1 ~ hw6.html/css/js ：這學期做的作業，加了相關網頁的超連結

5. game.html/css/js ：小遊戲，以滑鼠或方向鍵控制 Our ball 移動躲避 Enemy ball

6. search.php/css/js、test,js ：類似 google 已經停用的功能”以圖搜圖”，利用已訓練的 mobilenet model 來辨認所輸入圖片，並將分類輸出在網頁搜尋上。

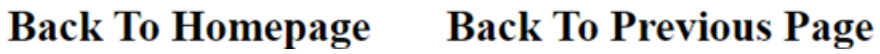
7. 伺服器架設 ： 使用 wampserver 依前面所述檔案和架構築成一個 server，並開放對外連線使用架構內的功能

程式詳述：

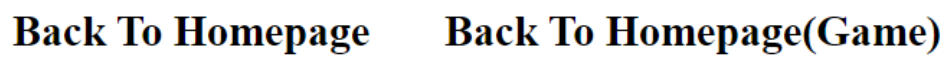
**0. Universal set**

1. html set：

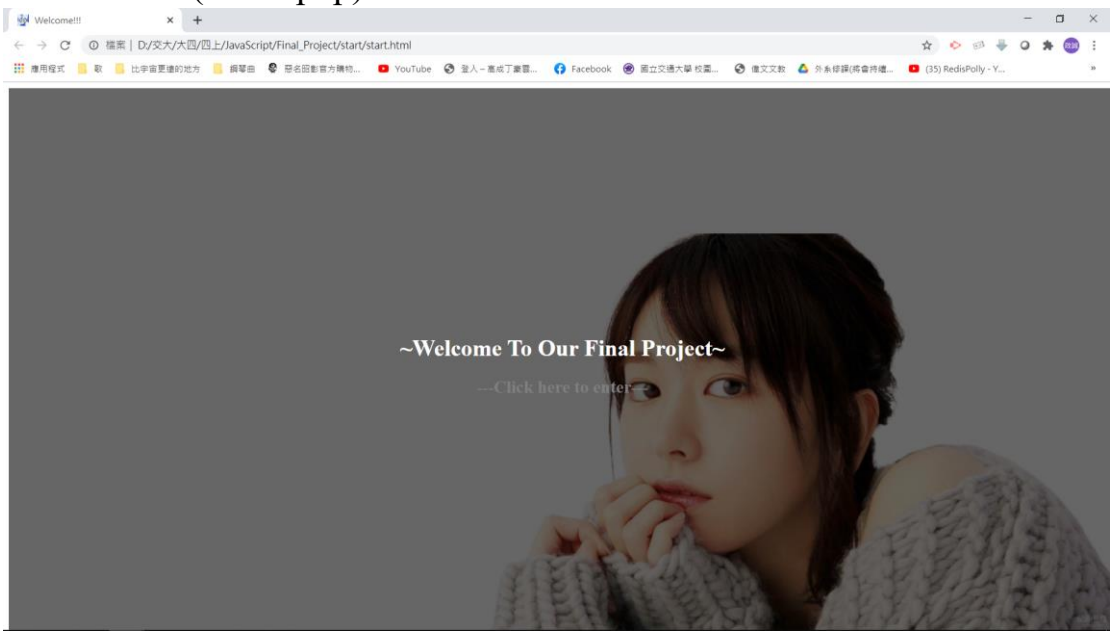
- footer: button[超連結至首頁(enter.html)或上一頁(history.back())]




- footer: button[超連結至首頁(enter.html)或遊戲首頁(game.html)]

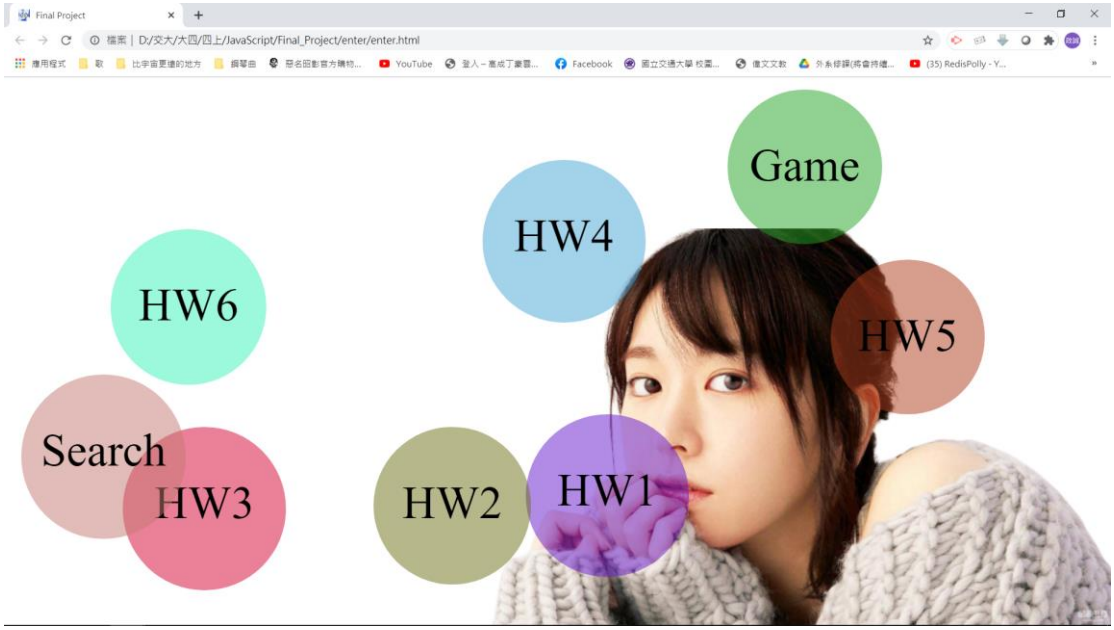



1. start.html (index.php) /css




=> 藉由遮罩裝飾初始頁並用 href 連接 enter.html 到其他的檔案

2. enter.html/css/js

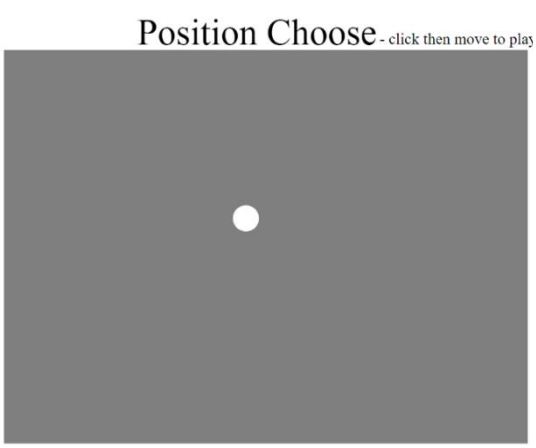


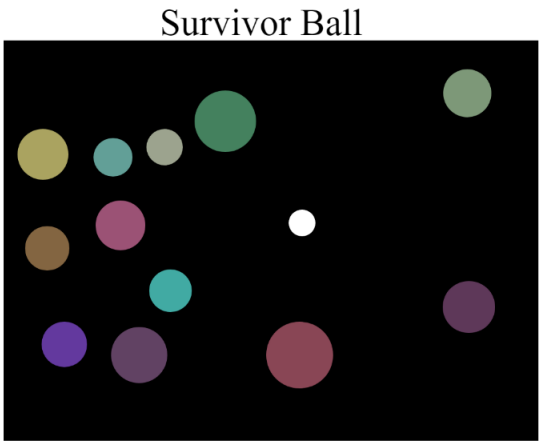

=> 藉由 click event 偵測，根據當時的滑鼠相對瀏覽器位置(e.clientX, e.clientY)與各圓心的距離和其半徑得比較進行超連結跳轉判斷。

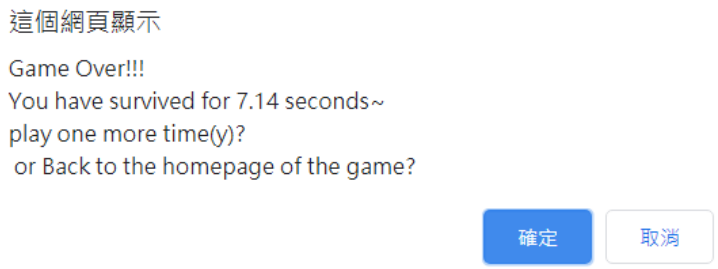
5. game.html/css/js


Game - Survivor Ball :以滑鼠或方向鍵控制 Our ball 移動以躲避 Enemy ball

1. 前圖為遊戲主頁，可點選按鈕選擇操作模式(滑鼠操縱 or 方向鍵操縱)。選擇 Mouse Control，會進入 Our ball 初始位置選擇頁面，藉由點選霧面區域範圍內位置決定遊戲初始位置，並以滑鼠鼠標為 Ourball 圓心位置進行遊戲。選擇 Keyboard Control，直接開始遊戲，以方向鍵控制 Ourball 位置進行遊戲。Game Over 後，顯示存活時間以及回到首頁(enter.html)或是遊戲首頁(game.html)的選擇。




=> Game over 示意圖

2. 創造 2 種物件:
   => Ball()(Enemy ball)[需躲避的球], push to aBall[]
   => PlayBall()(Our ball)[控制的球] , push to bBall[]
3. 初始位置:
   - Mouse Control:
     => Our ball: 藉 **mousemove** event 由 e.clientX, e.clientY, radius 決定
     => Enemy ball: Our ball 生成後，隨機在 container 中生成。
   - Keyboard Control:
     => Our ball: 控制.coor.x, .coor.y，固定生成於 container 右下角
     => Enemy ball: 同 Mouse Control
4. 移動模式:

◆ Enemy ball:
   - 碰撞模式:
     => 與邊界的距離決定是否反彈
     => Enemy ball 間的距離決定是否碰撞。

◆ Our ball:

- Mouse Control:
  => 以 mousemove event 決定球座標，根據與邊界的距離決定是否停住(滑鼠在遊戲區域外，Ourball 在停在裡面)
- Keyboard Control:
  => 鍵盤左右上下鍵決定移動方向
  => 根據與邊界的距離決定是否停住

5. 難度提升
   0. 藉由 var 之特性讓 balls 可以在 function 中改變
   1. 藉由計算 setinterval 執行的次數，每 5 秒增加一次難度。
   2. 增加一顆 enemyball
   3. 增加速度:對所有 Enemy ball, Our ball 隨機增加.speed.x, .speed.y

6. Game over 條件
   1.將 Our ball 的與所有 Enemy ball 的距離做比較，若 Our ball 和 Enemy ball 有交集，即 Game Over
   2. confirm(): 彈出式視窗，顯示 point = point.toFixed(2)值(存活時間)，根據選擇跳轉到不同頁面
   => Ok : game(house).html/game(keyboard).html[重新開始遊戲]
   => Cancel : game.html[回到遊戲首頁]

7.**程式設計巧思**

A. 遊戲開始時，先生成 Ourball，再一顆一顆隨機生成 Enemyball，把新生成的 Enemyball 對先前生成的 Enemyball 以及 Ourball 進行距離、半徑對比得出是否交集的結論，若有交集就改變對應 bool 值，並包在由 bool 值決定是否迴圈的 **while** 中，這樣就可以讓新生成的 Enemyball 絕對不會與任何球有交集。

B. Enemyball 碰撞判定: 舉例來說，若是我現在螢幕上有 7 顆 Enemyball，從第 0 顆執行到第 6 顆，當在進行第 n 顆的迴圈時，先更新它的 coor (coor += speed)，它只與第 n+1 顆~第 6 顆球進行是否交集的比對，若交集，則把第 n 顆球的位置回歸(coor -= speed)，再根據碰撞公式，對兩顆球的 speed.x, y 進行改變。這樣做會有誤差，因此我讓 **move()執行的週期從原本的 100ms 降到 10ms**，使每次移動的距離變成原本的 1/10(speed 變成 1/10 倍)，讓該誤差變成肉眼辨識不出的誤差。

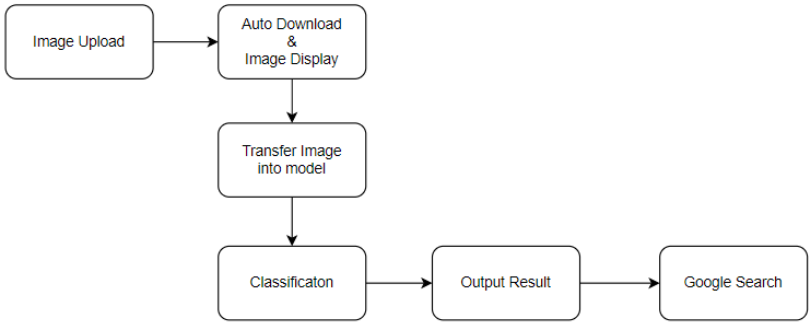
C. 同 B.，將 move()執行的週期從原本的 100ms 降到 10ms，可以讓 Ourball 視覺上可以跑得更順(原先會有頻閃現象)

D. 控制方向鍵移動 Ourball 時，由於是基於 keydown event 來觸發移動的，當長按方向鍵時，會因為 keypress 的原因使 Ourball 有開始

移動的延遲。解決方法：**在 keydown 時執行 setInterval 並在 keyup 時執行 clearInterval 來避免 keypress delay**。這樣做會造成同時按多個方向鍵時，由於啟動時間不一樣，會導致 Ourball 無法以完美斜直線運動。**當同時按複數個鍵時，把在執行的鍵停止(clearInterval)，再把複數個鍵一起開始(setInterval)。(各方向移動時間幾乎相等的話就可以讓移動軌跡為直線)**。這樣做可支援同時按複數個方向鍵的情況(上下左右都按也行)。

## 6. Search.html/css/js、test.js

Flow




a. 創造上傳連結

```
<input type="file" accept="image/*" name="output" onchange="loadFile(event)"><br>
<img id='output'>
```

➔ 使用 accept 接受上傳的檔案並用 onchange 用來 trigger loadFile(event)

➔ 指派一個 id 跟 name 給上傳的圖片，方便後面預覽及下載

b. 讀取上傳檔案並預覽上傳圖片

```
<script>
    // process the function by onclick
    var loadFile = function (event) {
    // grasp the uploaded image by ID and its info
    var output = document.getElementById('output');
    output.src = URL.createObjectURL(event.target.files[0]);

    // use canvas to display uploaded image
    output.onload = function () {
        var canvas = document.createElement('canvas')
        canvas.width = output.width
```

```
        canvas.height = output.height
        var context = canvas.getContext('2d')
        context.drawImage(output, 0, 0, output.width, output.height)
        };
    };
</script>
```

➔ 利用 canvas 畫出和上傳檔案相同大小的 frame，之後讀取圖片的訊息複製到 canvas 中，就可以顯示出上傳圖片的樣子

c. 利用 button 觸發 server 自動下載上傳的檔案並顯示檔案元素

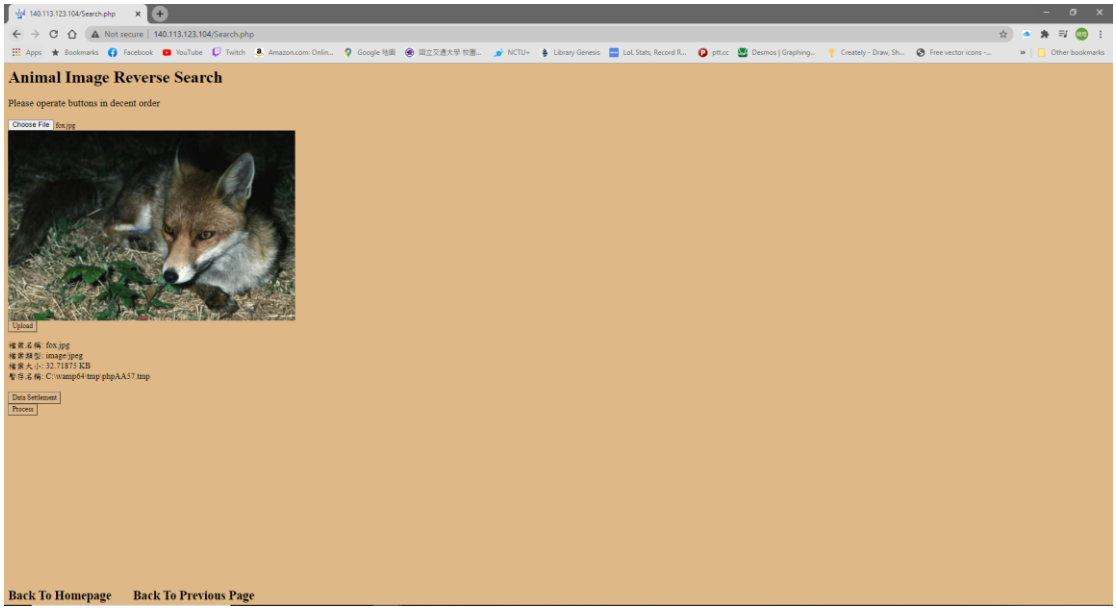
```
<input type="submit" name="upload" class="button" value="Upload"/>
<?php
if(array_key_exists('upload', $_POST)) {
        move();
    }
function move(){
    # Check for uploading
    if ($_FILES['output']['error'] === UPLOAD_ERR_OK){
        echo '檔案名稱: ' . $_FILES['output']['name'] . '<br/>';
        echo '檔案類型: ' . $_FILES['output']['type'] . '<br/>';
        echo '檔案大小: ' . ($_FILES['output']['size'] / 1024) . ' KB<br/>';
        echo '暫存名稱: ' . $_FILES['output']['tmp_name'] . '<br/>';
        # Check for Existence
        if (file_exists('C:/wamp64/www/upload/' . $_FILES['output']['name'])){
            echo '檔案已存在。<br/>';
        } else {
            $file = $_FILES['output']['tmp_name'];
            $dest = 'C:/wamp64/www/upload/' . $_FILES['output']['name'];
            # Move file and rename
            move_uploaded_file($file, $dest);
            rename($dest, "C:/wamp64/www/upload/image.jpg");
        }
    } else {
    echo '錯誤代碼：' . $_FILES['output']['error'] . '<br/>';
    }
}
?>
```

➔ 指定 button 的 name 為 upload，判斷如果觸發 upload 則執行 move()

➔ 利用$_FILES[‘output’][‘error’]如果上傳檔案沒有問題就輸出 file_elements 顯示操作正常

➔ 設定變數$file 和$dest 並利用這兩個變數將檔案從暫存區存入指定的資料夾

➔ 用 rename()把移動過後的檔案重新命名為 image.jpg 來符合 model 操作格式




d. 利用 button 觸發 data settlement

```
<button onclick="Reload()">Data Settlement</button><br>
<script>
    var Reload = function(){
        location.reload();
    }
</script>
```

➔ Reload 網頁避免資料因不知明原因無法傳入 test.js

e. 利用 button 喚醒 server 的 cmd 並執行 model (nodejs)

```
<button onclick="Process()">Process</button>
<script>
var Process = function(){
    var out = "<?php echo exec("node C:/wamp64/www/test.js"); ?>";
    window.open("…")
};
</script>
```

➔ 使用 php 執行 cmd 並輸入指令

➔ 將 model prediction 傳入 out 之後用 window.open()使用關鍵字 google search

f. 將載下的檔案變成符合 model 的 input (tensor)

```
// read the image and decode .jpg into pixels
const readImage = path => {
  const buf = fs.readFileSync(path)
  const pixels = jpeg.decode(buf, true)
  return pixels
}

// make the pixels into arrays
const imageByteArray = (image, numChannels) => {
  const pixels = image.data
  const numPixels = image.width * image.height;
  const values = new Int32Array(numPixels * numChannels);

  for (let i = 0; i < numPixels; i++) {
    for (let channel = 0; channel < numChannels; ++channel) {
      values[i * numChannels + channel] = pixels[i * 4 + channel];
    }
  }
  return values
}

// make the input image into tensors
const imageToInput = (image, numChannels) => {
  const values = imageByteArray(image, numChannels)
  const outShape = [image.height, image.width, numChannels];
  const input = tf.tensor3d(values, outShape, 'int32');

  return input
}
```

➔ readImage 負責讀取 image.jpg 並把 jpg decode 成 pixels

➔ imageByteArray 將已經變成 pixels 的檔案再一步轉化成 array

➔ 最後 imageToInput 把 array 轉成 tensor 送到 model

g. 將下載的圖片送入 train 好的 model 中執行預測

```
const classify = async (path) => {
  // read image and transfer into suitable input
  const image = readImage(path)
  const input = imageToInput(image, NUMBER_OF_CHANNELS)

  // load pretrained and classify the image
  const  mn_model = await loadModel()
  const predictions = await mn_model.classify(input)

  // output result
  const result = predictions[0].join()
  console.log(result)
```

➔ imageToInput 是利用 fs 將原本的像素點轉換成 tensor array，使 input 符合 model 架構

➔ 為了去掉 string 的”而使用 join()使得 result 出來的結果為純字元

➔ Model 的部分因為考慮前後端之間的資料傳送和執行速度，所以選用

mobilenet v2 作為 training model，train dataset 是使用 imagenet，後來有再把 model 多跑了 200 個 epoch，但準確度沒有比原本的好，所以就還是使用原本的 weight 來預測

| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 32 | 1 | 2 |
| $112^2 \times 32$ | bottleneck | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | bottleneck | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | bottleneck | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | bottleneck | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | bottleneck | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | bottleneck | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | bottleneck | 6 | 320 | 1 | 1 |
| $7^2 \times 320$ | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2 \times 1280$ | avgpool 7x7 | - | - | 1 | - |
| $1 \times 1 \times 1280$ | conv2d 1x1 | - | k | - | |

➔ Predictions 原本的輸出為 dict 顯示，但為了符合 search 的需要，我將格式轉成 list，並選取最有可能的 class 作為輸出

```
topClassesAndProbs = [];
for (i = 0; i < topkIndices.length; i++) {
        topClassesAndProbs.push([
        imagenet_classes_1.IMAGENET_CLASSES[topkIndices[i]]
        ])}
```

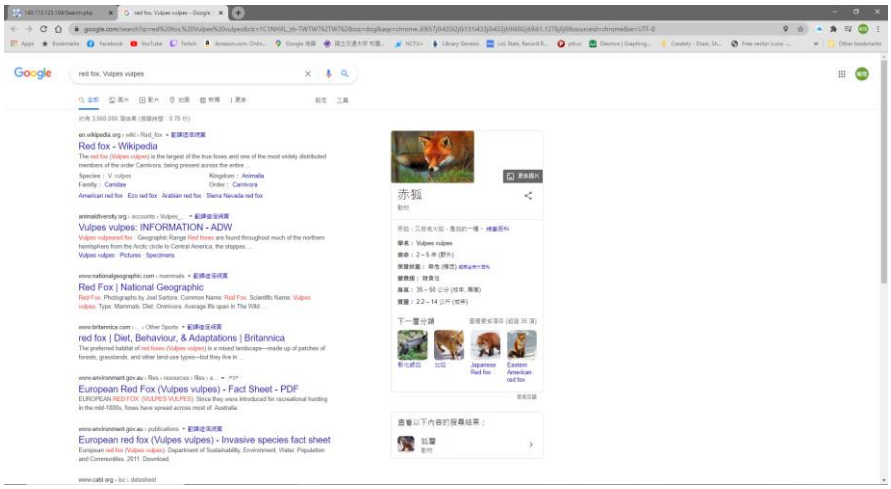
h. 將結果 output 成網頁搜尋並刪除下載圖片避免覆蓋

```
// search the result with google
open("https://www.google.com/search?q=" + result + "&rlz=1C1NHXL

// use fs to remove image for avoid data overlapping
remove_file(path)
}
```

➔ Open()類似 html 的 window.open，remove_file 是用 unlink 將下載圖片給丟棄

i. Search 執行畫面展示



## 7. 伺服器架設

a. 原因 & 功能布置

為了要實現一體化的 Search 功能，我選擇使用架 server 的方式將前面設計的功能包在裡面，因為單純用 html 和 js 僅能調用 local 檔案而無法和 client 互動，此外 js 無法自動開啟 cmd 來執行 nodejs 的指令，因為 model prediction 如果要以 js 實現一定需要 import node library (node_modules)，透過 php 擁有的互動和 server 資料儲存功能，就可以把步驟全部整合在網頁上順利執行

b. 工具 & 設定

➔ 使用 Wampserver 在本機架設對外聯絡網站
➔ 把 require local 改成 require all granted 開放 client 權限訪問網站
➔ 在路由器創建固定 DHCP 後將 distributed IP 使用 port forwarding (port: 80) 建立對外 IP 到連接裝置 (PC->Server)的 bridge
➔ 將原剛開始的檔案 start.html 更改成 index.php 建立訪問起始位置
➔ 將必要使用 php 的.html 檔更改檔名成.php 並進行對應 code 更改

c. 網站地址

➔ **http://114.32.242.51**

Demo:

➔ https://ppt.cc/fksnbx

分工:

林政誠: Coverage (start)主要編寫及設計、enter 主要編寫及設計、主要視窗架構和附加功能設計、網頁流程主要設計、Survivor Ball 主要編寫及設計、視覺優化(css)

吳峻陞: Search 功能和畫面設計、model 模擬和訓練及其餘相關參數設定、nodejs 相關程式編寫及環境設定、主機和網頁互動設計、Server 架設與相關 code 設計、.php 檔設計、視覺優化(css)

★ 雖然有主要負責的部分，但全部的排版及設計都有經過兩個人的討論

心得

林政誠:

Final Project 讓我使用不斷使用 event 來實現與使用者端的連動，算是一種很新奇的體驗。除了基本的網頁設計(html)、美工(css)外，在 js 程式裡也使用許多物件的觀念搭配上寫好的網頁設計做出遊戲，以前寫程式總是拿來做計算或做邏輯設計，這堂課讓我看到自己的程式顯示在眼前，也是一次很好的體驗

吳峻陞:

這次作業選擇做了一個比較大的題目，當初想說可能單靠網頁語言+JS 可能無法實現全部的東西，也沒想到要架 server，結果在碰壁的第 379 次成功的利用 html+php+js 把整個架構都實現了，雖然還有進步的空間(美觀的部分)，但這次的 project 帶給我很多的收穫，之前都是碰像 C、python 這種跟電腦對話的語言，課堂上學到的視覺化(html)跟互動式(php, js)語言真的第一次嘗試去熟悉，能把這個 project 從無到有生出來真的讓人感到相當大的成就感!