中華民國第 62 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

(鄉土)教材獎

052509

「澱」「資」的奧祕-利用網頁遊戲精進學生對沉澱表之研究

學校名稱：臺中市立臺中第二高級中等學校

作者：	指導老師：
高三 劉佑庭	林姵瓊
高三 林育聖	劉仲函

關鍵詞：學習沉澱表、網頁遊戲、AI 機器學習

摘要

有鑑於高中選修化學相當困難，因此我們製作了一款有助於提升化學能力的遊戲。玩家必須活用沉澱表及化學來通過遊戲。蒐集離子球、並且按照要求合成出沉澱物或是酸鹼，來擊潰敵軍和敵方堡壘。

該遊戲是藉由網頁去宣傳的，因此我們需要一個平台去發布此遊戲，選擇了 netlify 這個平台。其中我們前端使用了三大前端程式，HTML、CSS、JavaScript，後端則利用 Google 試算表搭配 Google APP Script 去收集遊戲數據。最後，我們再利用 AI 演算法中的類神經網路分析，透過交叉比對找到玩家最有可能的成長結果。

壹、前言

一、研究動機

每個理組的學生在高中時必定會經歷過「化學沉澱表」的折磨。不只要硬背起來，還要學會應用，簡直是選化 III 的最大瓶頸。

因此，有程式開發能力的我們，決定開發一款以化學沉澱表為核心的遊戲，再將各種化學特性規則寫入，希望能夠幫助往後的高中學生更容易度過這個難關。

二、研究目的

開發一款能夠讓學生更熟悉化學沉澱表、各種酸鹼的特性、各種陰陽離子特性且好玩的遊戲，並且實測受試學生前後化學觀念是否有進步、進一步預判出學生化學的成長幅度。

貳、研究設備及器材

一、硬體：筆電、PC、手機、選修化學 III 講義

二、軟體：Visual Studio Code（開發）、photocap（製圖）、Google 試算表（後端）、Google APP script、Google form、netlify、Brain.js

參、研究過程

一、思考遊戲規則與方法

二、開始學習程式語言:HTML、CSS、JavaScript

三、遊戲內容製作(詳見第肆大點)

四、美工圖製作

美工圖製作分為幾個部分（照製作順序排列）：離子球（單顆、雙顆、三顆）、沉澱物圖示製作、酸鹼等溶液圖示製作、主畫面（選單）背景製作、遊戲介面背景製作、各類按鈕圖示製作、遊戲規則介紹頁面製作。

五、請化學老師試玩、修正

遊戲完成時，我們把遊戲首先拿給化學老師試玩並幫我們找出問題。

六、找受試者遊玩遊戲

我們請兩班學弟妹來試玩我們的遊戲，並蒐集遊戲和前、後測表單數據。之後又請自然科老師幫忙，向他們所教的其他班級宣傳我們製作的遊戲。

研究過程有三大部分，前測、遊玩過程、後測。分成 ABC 三份考卷，每份考卷有前測與後測，題目共有 XYZ 三個部分。每部分 5 個題型各一題。

	考卷 A	考卷 B	考卷 C
前測	XY	XZ	YZ
後測	Z	Y	X

表一：考卷的題目分布

由上表可以看出題目的分布，前測 10 題，後測 5 題。舉例來說，我們可以利用考卷 B 的答題正確率去推斷考卷 A 同學的 X 部分與 Z 部分的進步情況。

七、整理數據製成報告

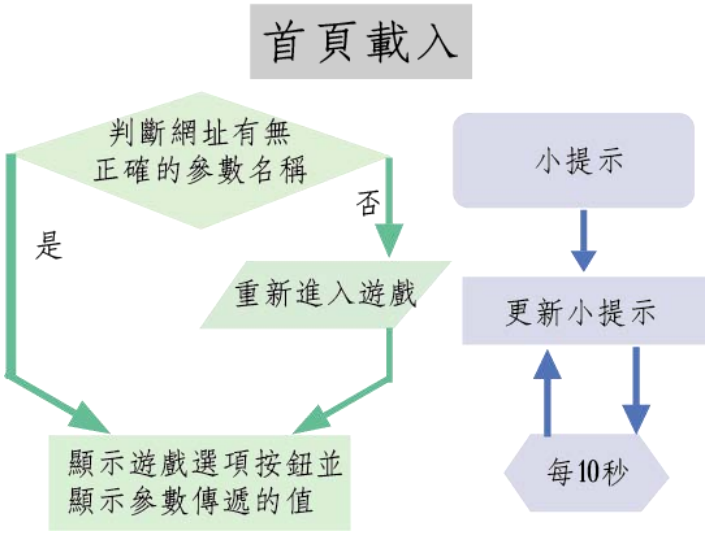
我們將後台數據加入機器學習，得到結論後製成報告。

肆、研究內容

一、遊戲流程圖(製作邏輯介紹)

(一)首頁載入

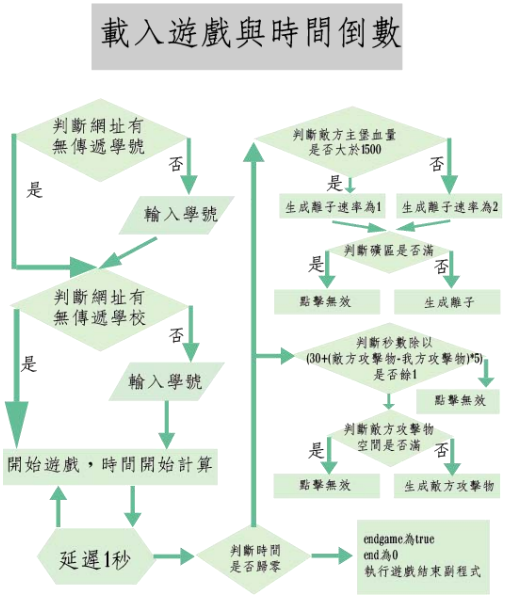
當首頁畫面載入時，會先判斷網址傳遞的參數名稱是否正確，若錯誤則重新導入遊戲網址，反之顯示按紐，並將按鈕的 value 值設為傳遞的參數。遊戲首頁還有另一個部分，小提示，在此要每段時間更新一次，讓資訊量增加，而我們決定以 10 秒為一輪。首先畫面載入後會顯示第一次，之後延遲 10 秒再次執行，則會達到每 10 秒一輪的效果。



圖一：首頁載入流程圖

(二)遊戲載入與計時

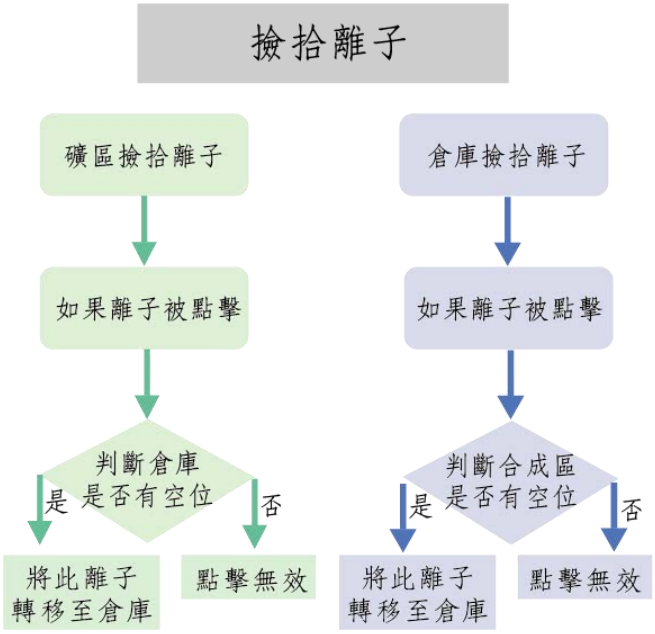
進入遊戲介面後，需要先判斷網址是否有傳遞學號以及學校的參數，若沒有則跳出視窗提醒玩家輸入，獲得兩參數之後，開始進行遊戲，時間開始計算，時間計算會影響到礦物生成、敵方攻擊物生成與遊戲結局，所以在計算時間這個函式中必須執行這三個部分的判斷，第一部分先判斷時間是否歸零，若是，則時間到、遊戲結束，執行 form 函式，若不是則執行生成敵方攻擊物以及礦物的程式，在判斷的程式中，被除數為時間，除數為需要間隔的時間，而當餘數為 0 時執行生成函式，而函式部分會先判斷該生成位置是否額滿，若額滿則不生成。而需要間隔時間在離子生成中，分成 1 秒與 2 秒；在敵方攻擊物生成中，分成 5~55 秒，而這個秒數由敵方與我方攻擊物的差所控制。



圖二：載入遊戲與時間倒數流程圖

(三)撿拾離子

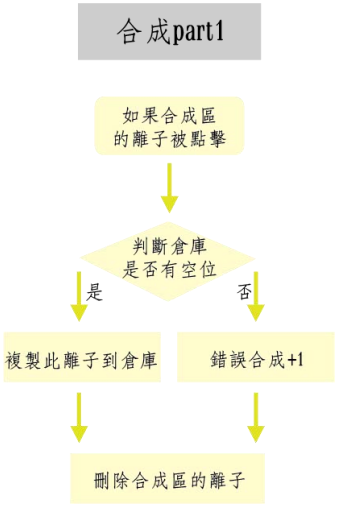
拿取離子分成兩個部分，一個是礦區拿到倉庫，另一個是倉庫拿到合成區，這兩個程式大同小異，皆是利用點擊按鈕去移動離子，當離子被點擊時，會判斷目的地是否有空位，若沒有，則點擊無效，若有，則複製離子到目的地的第一格空格中，且將初始位置的離子移除，達到移動離子的效果。



圖三：撿拾離子流程圖

(四)合成 part1

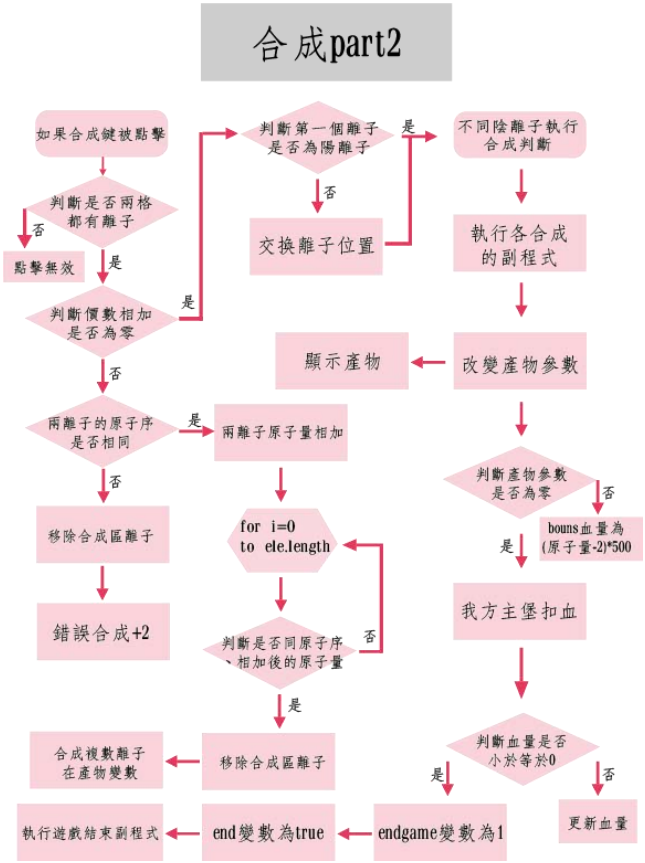
合成離子一共有三個部分可以被點擊，合成鍵被點擊、合成區離子被點擊以及產物被點擊。當合成區的離子被點擊時，會先判斷倉庫是否有空位，若是則將此離子複製到倉庫的第一個空格的位置，並移除此離子原先的位置，若沒有空位則將合成錯誤數+1 以示懲罰，並將合成區的離子移除。



圖四：合成 part1 流程圖

(五)合成 part2

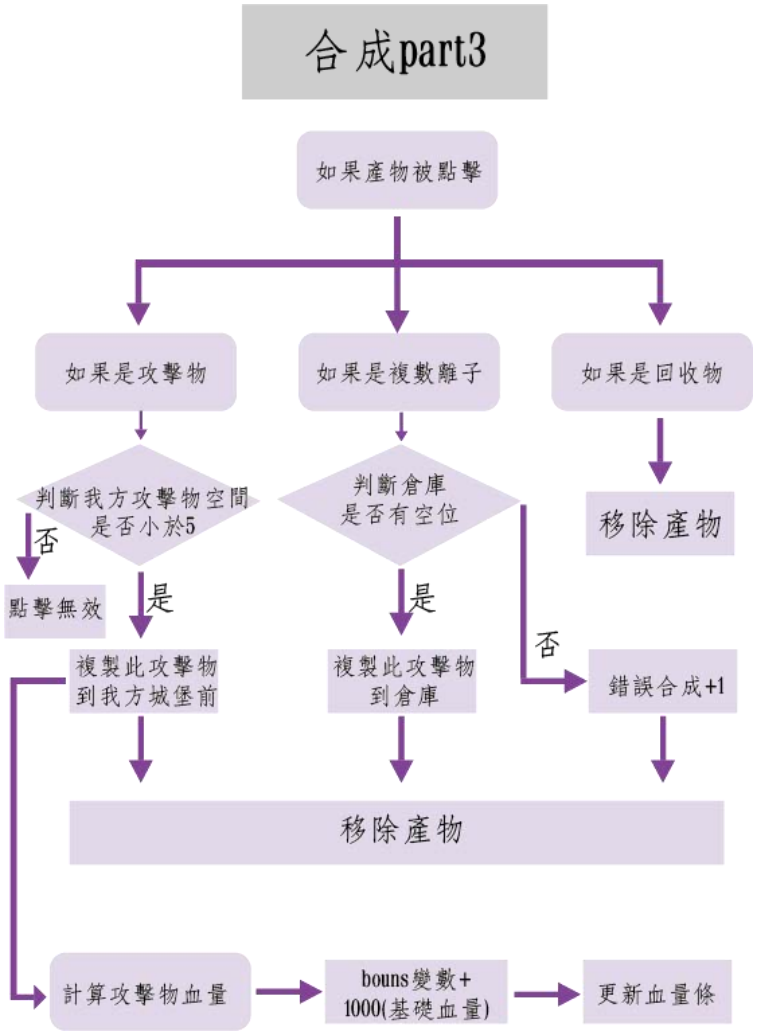
當合成鍵被點擊時，先判斷是否兩格皆有離子，若有則繼續判斷，沒有則點擊無效，再繼續判斷價數相加是否為 0，若是則開始合成，不是則判斷兩個離子的原子序是否相同，若是則將其原子量相加，再用 for 迴圈找出陣列裡符合的複數離子，找不到視為沒有收錄，點擊無效；找得到，則進行合成，移除合成區的離子且生成複數離子產物，產物參數為-1，若不是則視為錯誤合成，錯誤合成數+2（因為有兩個離子），移除合成區的兩個離子。開始合成的第一步，將陽離子交換到第一個位置好讓程式編寫，第二步開始判斷，我們藉由在離子內相對的布林值進行判斷，舉例來說，當陰離子為氯離子，且陽離子的 ClminusBrminusIminus 為 true，那它們合成會變成沉澱物，在經過一系列的判斷過後，產物參數會遭受改變，若沒有改變則是預設值 0（回收物），隨後判斷產物參數是否為-1，如果是顯示複數離子，如果不是判斷參數是否為 0，是的話代表合成出回收物，我方主堡扣血，隨後判斷我方主堡血量是否小於等於 0，若是則 endgame 變數為 1、end 變數為 true，以及執行遊戲結束副程式，若不是則更新主堡血量，若參數不為 0 則代表合成出了攻擊物，我們要在此計算其 bonus 血量，普通離子含有兩顆原子，基礎血量為 1000 點，因此多一顆離子代表 500 點的 bonus 血量，所以 bonus 的計算為（原子量-2）*500。



圖五：合成 part2 流程圖

(六)合成 part3

當產物被點擊時，有三種狀況，狀況一、複數離子被點擊，需要回到倉庫，與移回離子一樣，先判斷倉庫是否有空間，若沒有則錯誤合成+1，若有則移回倉庫的第一個空位。狀況二、回收物被點擊，則移除此回收物。狀況三、攻擊物被點擊時，需要將其移動到我方主堡前，計算其攻擊物的血量，為 1000 基礎血量（沉澱物）加上 bonus 額外血量，除了移動至攻擊區，血條也要更新數值。

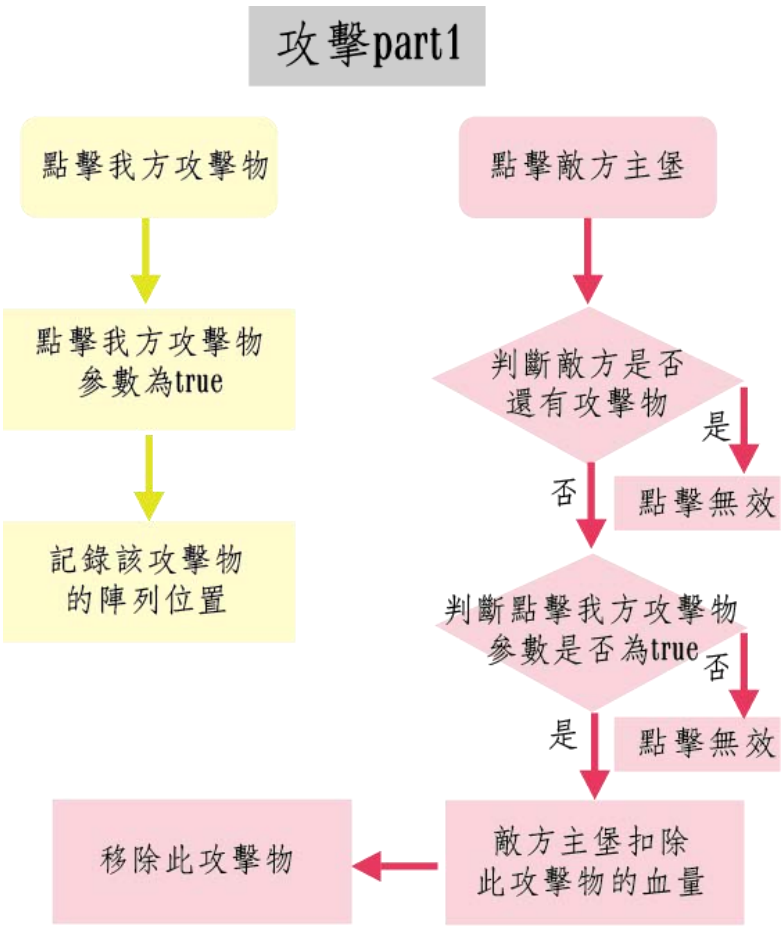


圖六：合成 part3 流程圖

(七)攻擊 part1

當點擊我方攻擊物時，將點擊我方攻擊物參數 check 設為 true，以及記錄此攻擊物陣列位置。

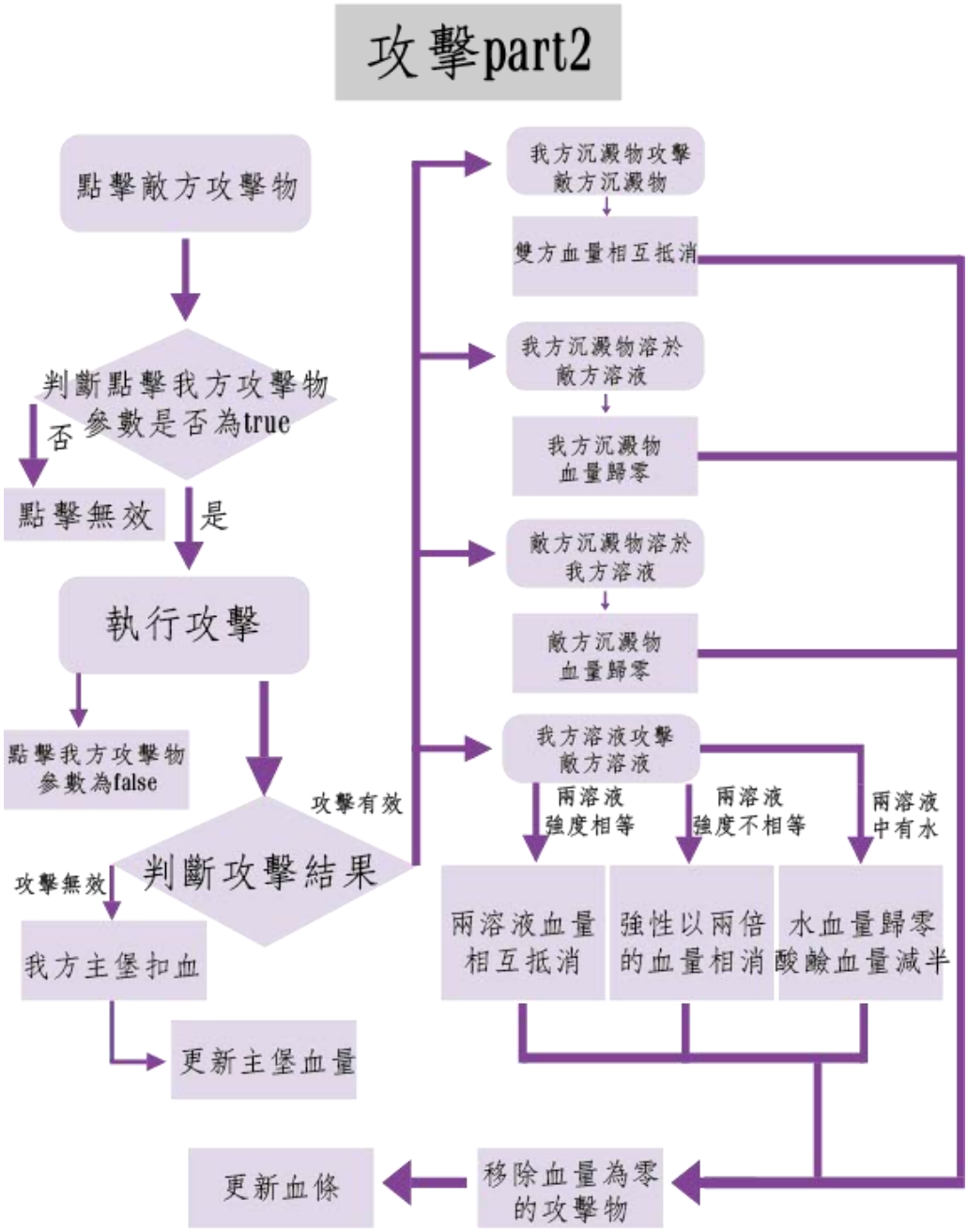
當點擊敵方主堡時，先判斷敵方是否還有攻擊物，若是點擊無效，若不是，再判斷 check 參數是否為 true，若是則敵方主堡扣除我方攻擊物的血量，並此攻擊物移除。舉例：敵方主堡為 5000 點，我方攻擊物 500 點，對戰結果則是我方攻擊物遭移除，敵方主堡剩下 4500 點。



圖七：攻擊 part1 流程圖

(八)攻擊 part2

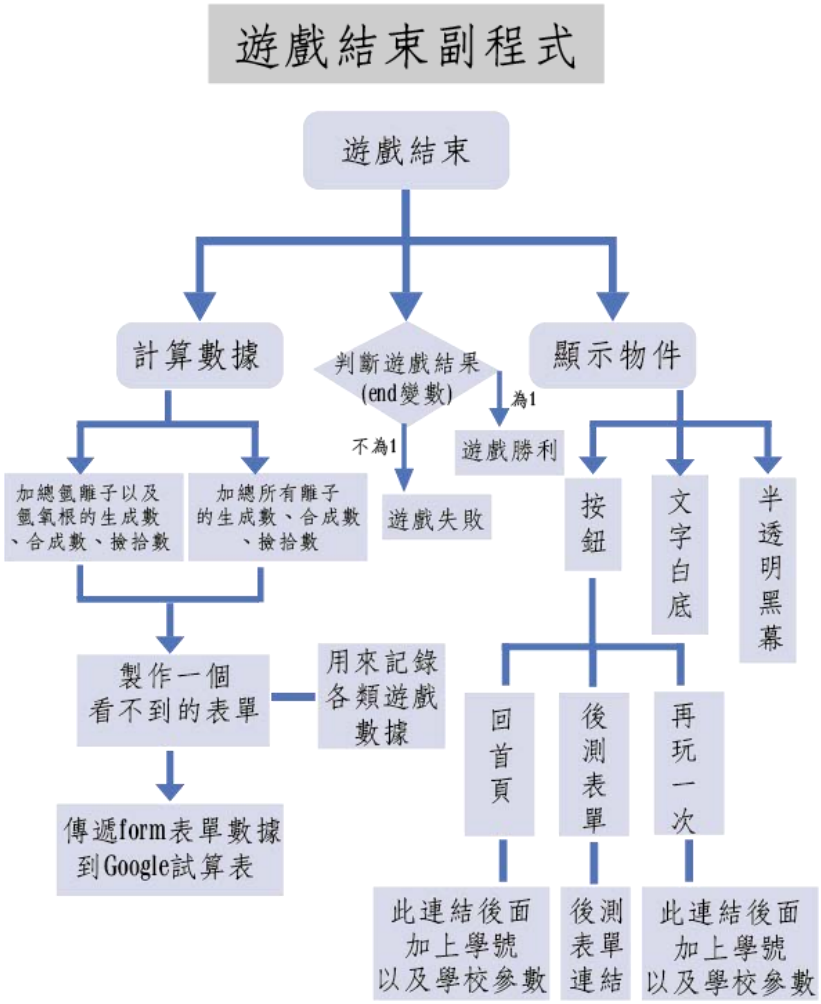
當點擊敵方攻擊物時，先判斷 check 參數是否為 true，若不是則點擊無效，若是則繼續判斷攻擊結果，若攻擊無效，則我方主堡扣除 500 點血量，並更新我方主堡血條數值，若攻擊有效，更新對戰結果，此部分會有三種結果，一、我方沉澱物攻擊敵方溶液並且此沉澱物溶於此溶液，對戰結果：我方沉澱物被移除。二、我方溶液攻擊敵方沉澱物並且此沉澱物溶於此溶液，對戰結果：敵方沉澱物被移除。三、我方沉澱物攻擊敵方沉澱物，對戰結果：雙方血條相互減少，血量歸零的一方遭移除。四、我方溶液攻擊敵方溶液，如果其中溶液沒有水，則是酸鹼中和，而弱酸鹼在此的戰鬥力降至一半，對戰結果：雙方相互攻擊，弱性攻擊強性，弱性扣除強性弱性一半攻擊力；強性攻擊強性或是弱性攻擊弱性，雙方攻擊力相消；強性攻擊弱性，強性的攻擊力變至兩倍。舉例來說：我方 250 點的強酸攻擊敵方 1000 點的弱鹼，對戰結果則是我方強酸攻擊物遭移除、敵方弱酸攻擊物血量剩下 500 點。



圖八：攻擊 part2 流程圖

(九)遊戲結束副程式

執行遊戲結束副程式將結算此遊戲結果，除了要告知玩家遊戲結果為何還需要傳遞數據到後端的數據庫以便我們剖析數據研究以及調整。因此此副程式，要執行三大部分，一、計算數據計算所有離子加總生成數、合成數、撿拾數以及重要離子氫離子與氫氧根，之後製作一個看不見的 form 表單讓我們可以記錄各項 value 傳遞回 Google 試算表。二、判斷遊戲結果，此判斷以 end 為基準，判斷文字需顯示何者。三、顯示物件，按鈕、文字白底、與半透明黑幕，而按鈕包含了再玩一次、後測表單以及回首頁，其中再玩一次與回首頁會將剛剛輸入的學號以及學校加在網址後這樣就不用再重新輸入。



圖九：遊戲結束副程式流程圖

二、程式開發

(一)攻擊物物件陣列

此部分為攻擊物的物件陣列，Mode 為 class 名稱，number、state、compound、attackaction 為其變數，其中 number 為編號，方便編碼圖片的檔名、state 為攻擊物狀態，2 為液態、1 為固態，以便攻擊時判斷攻擊結果，而 compound 與 attackaction 為記錄後端數據所用，初始值為 0。

```
//number,state,compound,attackaction
//標碼]、狀態(2=液態,1=固態)、合成次數、攻擊次數
attack[0]=new Mode(1,2,0,0);
attack[1]=new Mode(2,2,0,0);
attack[2]=new Mode(3,2,0,0);
attack[3]=new Mode(4,2,0,0);
attack[4]=new Mode(5,1,0,0);
attack[5]=new Mode(6,1,0,0);
attack[6]=new Mode(7,1,0,0);
attack[7]=new Mode(8,1,0,0);
attack[8]=new Mode(9,2,0,0);
attack[9]=new Mode(10,2,0,0);
attack[10]=new Mode(11,1,0,0);
attack[11]=new Mode(12,1,0,0);
```

圖十：生成礦物 JavaScript 程式碼

(二)離子物件陣列

此部分為離子的物件陣列，分別記錄了 number 編號、serialnumber 序號、pick 被撿次數、compound 合成次數、plusminus 正負值、atomicnumber 原子數、src 圖片位置、ion 離子名稱與判斷合成的布林值，ClminusBrminusIminus 氯溴碘離子、SO42minus 硫酸根、CrO42minus 鉻酸根、S2minus 硫離子、OHminus 氫氧根、C2O42minus 草酸根、CO32minusSO32minusPO43minus 碳酸根亞硫酸根磷酸根，而這些布林值可以幫助我們在編寫時更加順利。當 ClminusBrminusIminus、SO42minus、CrO42minus 為 true 時視為難溶，當 S2minus、OHminus、C2O42minus、CO32minusSO38minusPO43minus 為 true 時視為可溶。

```
//number, serialnumber, pick, compound, plusminus,atomicnumber, src, ion, ClminusBrminusIminus, SO42minus, CrO42minus, S2minus, OHminus, C2O42minus, CO32minusSO38minusPO43minus
//陣列位置, 序號, 被撿幾次, 合成次數,正負多少, 原子數, 圖片位置, 離子名稱, Cl-Br-I-, SO42-, CrO42-, S2-, OH-, Cr-, CO32-, SO32-, PO43-前三大多可溶,後四大多不可溶
ele[0]=new Element(0,0,0,0,1,1,"picture/+/Li+.png","Li+",false,false,false,true,true,true,true,false,false);
ele[1]=new Element(1,1,0,0,1,1,"picture/+/Na+.png","Na+",false,false,false,true,true,true,true,false,false);
ele[2]=new Element(2,2,0,0,1,1,"picture/+/K+.png","K+",false,false,false,true,true,true,true,false,false);
```

圖十一：離子物件陣列 JavaScript 程式碼

(三)遊戲計時

在計時器中需要計算總秒數、秒數、分鐘數以便控制遊戲結束、生產離子、敵方生產攻擊物。首先，判斷遊戲是否結束，如果結束則計時器不再計時，如果時間歸零且遊戲未結束則遊戲結束，並更改遊戲結局變數、執行遊戲結束副程式。

當礦區滿時，礦物重生時間重製。若不滿且未結束遊戲則將秒數+1。當未結束遊戲則將敵方生成攻擊物秒數+1。

在下一個部分為生成敵方攻擊物，在此，為了不要讓玩家囤積強酸專門攻擊特定沉澱物，且使敵方攻擊物生成速率愈來愈慢，讓敵方攻擊物就算太多玩家也有機會可以反敗為勝，因此將生成速率設為變數，此機制為 30 秒基礎秒數再加上五倍雙方攻擊物數量差，生成速率成 5-55 秒的等差數列。

最後一部分為生成礦物的區域，當敵方血量大於 1500 點時，生成速率為 1，若非，則為 2，此機制的目的為讓玩家可以越打越難，讓遊戲更有變化。之後隨機生成離子且隨機生成位置，使畫面更加有變化，但是為了避免都產生陰離子或是陽離子，讓畫面可以更佳的平衡，所以當陰離子或是陽離子離子池已經有 5 個時就不再生成，利用迴圈再使它生成其他離子，但如果是有特定狀況會強制生成其他離子，如：當強酸出現，我方場上沒有氫氧根離子，這樣會強制生成氫氧根離子。

```
function startCountDown(duration, element) {

    let min = 0;
    let sec = 0;

    let countInterval = setInterval(function (c) {
        gametimeRemaining++;
        if(activecheck==true){
            min = parseInt(secondsRemaining / 60);
            sec = parseInt(secondsRemaining % 60);
            if(gamemode==2||gamemode==4){
                if(secondsRemaining>0&&endgame==false)secondsRemaining = secondsRemaining - 1;
                else if(secondsRemaining==0&&endgame==false){
                    endgame=true;
                    end=0;
                    form();
                }
            }else{
                if(endgame==false)secondsRemaining = secondsRemaining + 1;
                else if(secondsRemaining==0&&endgame==false){
                    endgame=true;
                    end=0;
                    form();
                }
            }

            if(minerspace==9)minersec=0;
            else if(endgame==false)minersec++;
            if(endgame==false)fightsec++;
            element.textContent = `時間計時:${paddedFormat(min)}:${paddedFormat(sec)}`;
            //document.getElementById(`chunk2`).innerHTML=minerspace+","+backpackspace;
            if(fightsec%(30+(enemyspace-myteamspace)*5)==1&&enemyspace<5&&endgame==false)produceenemy();
            //if(fightsec%(2)==1&&enemyspace<5&&endgame==false)produceenemy();
            //console.log(parseInt(2999/eltowerblood))
            var producev=0;
            if(gamemode==0||gamemode==5){
                if(eltowerblood>=1500)producev=1;
                else producev=2;
            }else if(gamemode==1||gamemode==3){
                if(eltowerblood>=2500)producev=1;
                else producev=2;
            }else if(gamemode==2||gamemode==4){
                if(eltowerblood<=2500)producev=1;
                else producev=2;
            }
            if(minersec%(producev)==0&&endgame==false){
                var n= Math.floor(Math.random()*ele.length);
                var m= Math.floor(Math.random()*9);
                var puls=0;
                var minus=0;
                while(ele[n].atomicnumber!=1)
                    n= Math.floor(Math.random()*ele.length);
                while(miner[m].space==true){
                    c=0;

                    m= Math.floor(Math.random()*9);
                    for(var i=0;i<9;i++){
                        if(miner[i].space==true){
                            c++;
                        }
                    }
                    if(c==9)break;
                }
                for(var i=0;i<9;i++){
                    if(miner[i].space==true){
                        if(miner[i].ion.plusminus>0)puls++;
                        else minus++;
                    }
                }
                if(c!=9){
                    if(puls>=5){
                        while(ele[n].plusminus>0||ele[n].atomicnumber!=1)
                            n= Math.floor(Math.random()*ele.length);
                            //console.log(puls);
                    }
                    if(minus>=5){
                        while(ele[n].plusminus<0||ele[n].atomicnumber!=1)
                            n= Math.floor(Math.random()*ele.length);
                            //console.log(minus);
                    }
                }
                if(Hcheck==true){
                    n=8;
                }else if(OHcheck==true){
                    n=28;
                }else if(NH3check==true){
                    n=12;
                }

                ele[n].show_element(n,m);
            }
        }

    }, 1000);
}
```

圖十二：遊戲計時 JavaScript 程式碼

(四)生成礦物

此副程式位於礦物 class 中，功能為生成礦物，若礦區沒有滿則可以執行此程式，傳遞變數 n 為陣列的位置，m 為離子的陣列位置，隨後出現離子按鈕至礦區的該空格中，在最後當特定離子生成時，會觸發任務。

```
show_element(n,m){//n為陣列第幾個的元素 m為第幾個礦區
    if(minerspace<9){

        document.getElementById(`box`+(m+1)).innerHTML=`<button type="button" id="minerbutton`+(m+1)+`" class="minerbutton elementbutton"
        onmousedown="getElement(`+m+`,`+n+`);"> <img src="`+ele[n].src+`" class="minerimage elementimage"></button>`;
        miner[m].space=true;
        miner[m].ion=ele[n];
        ele[n].produce++;
        minerspace++;
        if(n==8){
            Hcheck=false;
        }else if(n==28){
            OHcheck=false;
        }else if(n==12){
            NH3check=false;
        }
        if(n==22||n==23||n==24){//22 23 24 Br-Cl-I-
            task(1);
        }else if(n==36){//36 SO42-
            task(2);
        }else if(n==37){//37 CrO42-
            task(3);
        }else if(ele[n].transition==true){
            task(4);
        }else if(ele[n].twonature==true){
            task(5);
        }

    }
}
```

圖十三：生成礦物 JavaScript 程式碼

(五)離子從礦區拿到倉庫

此程式為 getElement 的第一部分，這裡主要是編寫礦物從礦區拿到倉庫的動作，傳遞值 m 為判斷初始位置的變數，當 m>=0 代表它是礦區陣列的位置，之後當倉庫滿時不能獲得離子，因此要先判斷倉庫有沒有空格，若有則繼續進行，找出倉庫空格第一格的位置，隨後將礦區的離子顯示在倉庫，並將礦區陣列的物件移至倉庫陣列內，最後整理倉庫以及礦區的空間數變數，執行 delete_element 函式刪除礦區的離子。

```
function getElement(m,n){
    if(m>=0){
        if(backpackspace<10){
            var cc=0;
            backpacklocation=0;
            while(backpack[backpacklocation].space==true){
                backpacklocation++;
                backpacklocation=backpacklocation%10;
                cc++;
                if(cc==9)break;
            }
            document.getElementById(`backpack`+(backpacklocation+1)).innerHTML=
            `<button type="button" id="backpackbutton`+(backpacklocation+1)+`" class="backpackbutton elementbutton"
            onclick="getCompoundElement(`+(backpacklocation)+`);"> <img src="`+miner[m].ion.src+`" class="backpackimage elementimage">
            </button>`;
            backpack[backpacklocation].ion=ele[n];
            backpack[backpacklocation].space=true;
            ele[n].pick++;
            miner[m].ion=null;
            miner[m].space=false;
            backpackspace++;
            if(minerspace==9)minersec=0;
            minerspace--;
            delete_element(m);

        }
```

圖十四：離子從礦區拿到倉庫 JavaScript 程式碼

(六)離子從產物拿到倉庫

此程式為 getElement 的第二部分，這裡主要是編寫礦物從合成區的產物拿到倉庫的動作，當玩家合成後，可能會產出複數離子，因此要設置一個功能是將複數離子放置倉庫，給予玩家再度合成。與第一部分相同，先判斷倉庫有無空位，若有則找尋倉庫第一格空格的位置，將產物的物件移至倉庫的該陣列位置中。

```
}else if(m==-1){//拿合成出的複數離子
    if(backpackspace<10){
        var cc=0;
        backpacklocation=0;
        while(backpack[backpacklocation].space==true){
            backpacklocation++;
            backpacklocation=backpacklocation%10;
            cc++;
            if(cc==9)break;
        }
        document.getElementById(`backpack`+(backpacklocation+1)).innerHTML=`<button type="button"
        id="backpackbutton`+(backpacklocation+1)+`" class="backpackbutton elementbutton"
        onclick="getCompoundElement(`+(backpacklocation)+`);"> <img src="`+product.product.src+`
        " class="backpackimage elementimage"></button>`;
        backpack[backpacklocation].ion=product.product;
        backpack[backpacklocation].space=true;
        backpackspace++;
    }
```

圖十五：離子從產物拿到倉庫 JavaScript 程式碼

(七)離子從合成區拿到倉庫

此程式為 getElement 的第三部分，這裡主要是編寫礦物從合成區拿到倉庫的動作，當 m 值為-2 時，從合成區的第一格拿取至倉庫；當 m 值為-3 時，從合成區的第二格拿取至倉庫。跟第一部份一樣，執行動作前必須先判斷倉庫的空間數有沒有滿，若有則找尋倉庫第一個空格的位置，最後顯示離子到倉庫，並刪除合成區搬移的離子。

```
else if(m==-2){//拿合成區第一格離子
    if(backpackspace<10){
        var cc=0;
        backpacklocation=0;
        while(backpack[backpacklocation].space==true){
            backpacklocation++;
            backpacklocation=backpacklocation%10;
            cc++;
            if(cc==9)break;
        }
        document.getElementById(`backpack`+(backpacklocation+1)).innerHTML=
        `<button type="button" id="backpackbutton`+(backpacklocation+1)+`" class="backpackbutton elementbutton"
        onclick="getCompoundElement(`+(backpacklocation)+`);"> <img src="`+compound[0].ion.src+`"
        class="backpackimage elementimage"></button>`;
        backpack[backpacklocation].ion=compound[0].ion;
        backpack[backpacklocation].space=true;
        backpackspace++;
    }
    compounddelete_element(true,false);
else if(m==-3){//拿合成區第二格離子
    if(backpackspace<10){
        var cc=0;
        while(backpack[backpacklocation].space==true){
            backpacklocation++;
            backpacklocation=backpacklocation%10;
            cc++;
            if(cc==9)break;
        }
        document.getElementById(`backpack`+(backpacklocation+1)).innerHTML=
        `<button type="button" id="backpackbutton`+(backpacklocation+1)+`" class="backpackbutton elementbutton"
        onclick="getCompoundElement(`+(backpacklocation)+`);"> <img src="`+compound[1].ion.src+`"
        class="backpackimage elementimage"></button>`;
        backpack[backpacklocation].ion=compound[1].ion;
        backpack[backpacklocation].space=true;
        backpackspace++;
    }
    compounddelete_element(false,true);
```

圖十六：離子從合成區拿到倉庫 JavaScript 程式碼

(八)離子從倉庫移到合成區

因為當合成區滿的無法將倉庫的離子移到合成區，因此執行函數的一開始要先判斷合成區的數量是否小於 2，如果是則繼續執行，之後找尋有空格的第一格位置，並將倉庫那格離子複製到此空格陣列，且顯示按鈕在此空格中，再將倉庫陣列清空，最後整理倉庫以及合成區的空間數變數。

```
function getCompoundElement(m){//從背包拿離子到合成區
    if(compoundspace<2){
        var cc=0;
        while(compound[compoundlocation].space==true){
            compoundlocation++;
            compoundlocation=compoundlocation%2;
            cc++;
            if(cc==2)break;
        }
        if(compoundlocation==0){
            document.getElementById(`element1`).innerHTML=
            `<button type="button" id="elementbutton1" class="compoundbutton elementbutton"
            onclick="getElement(-2,-2)"> <img src="`+backpack[m].ion.src+`"
            class="compoundimage elementimage"></button>`;
        }else{
            document.getElementById(`element2`).innerHTML=
            `<button type="button" id="elementbutton2" class="compoundbutton elementbutton"
            onclick="getElement(-3,-3)"> <img src="`+backpack[m].ion.src+`"
            class="compoundimage elementimage"></button>`;
        }
        backpack[m].ion.compound++;
        compound[compoundlocation].ion=backpack[m].ion;
        compound[compoundlocation].space=true;
        backpack[m].ion=null;
        backpack[m].space=false;

        backpackspace--;
        compoundspace++;
        backpackdelete_element(m);
    }
```

圖十七：從背包拿離子到合成區 JavaScript 程式碼

(九)小提示

第一個判斷出當小提示要述說其他句話時，就中斷該句話的執行。 第二個判斷出當小提示還沒講完話且沒有要講其他話，則繼續執行遞迴讓小提示說話，每次重複執行都要延遲 25 毫秒，產生出動畫的效果。

```
function hinttextarea(m){
    if(m>0&&(hinttextnumber==0||hintcheck!=m)){
        document.getElementById("text").innerText="";
    }
    if(hinttext[m][hinttextnumber-1]!=" "&&m==hintcheck){
        document.getElementById("text").innerText=hinttext[m].substring(0,hinttextnumber);
        hinttextnumber++;
        setTimeout(`hinttextarea(${m})`,25);
    }else{
        hinttextnumber=0;
    }
}
```

圖十八：小提示程式碼

(十)新手試煉

新手試煉運用 step 這個變數去控制進度走向，圖中判斷的部分是生成離子，依序是生成氫離子、氫離子、硫酸根。而當每次執行完該動作 step 都會隨之加一。

```
if(step==0){
    ele[8].show_element(8,0);
    step++;
    stepcheck=0;
    steptextarea(0);
}else if(step==1){
    ele[8].show_element(8,1);
    step++;
}else if(step==2){
    ele[36].show_element(36,2);
    step++;
}
```

圖十九：新手試煉模式進度程式碼

(十一)小任務與獎勵

在 task 副程式中，當特定離子生成後會依照不同離子的特性去安排不同的任務和隨機獎勵，舉例來說當氯離子生成後，任務會安排合成出含氯離子的沉澱物，而每個任務完成後都會獎勵，執行 present 副程式，利用 if 判斷獎勵序號給予相對應的回饋。

```
function task(m){
    if(taskcheck==false){
        q=m;
        tasktotal++;
        document.getElementById("Qtd").innerHTML=`<img src="./picture/problem/Q/Q${m}.png" id="Qimg" class="QP">`
        p = Math.floor(Math.random()*3+1);
        document.getElementById("Ptd").innerHTML=`<img src="./picture/problem/P/P${p}.png" id="Pimg" class="QP">`
        taskcheck=true;
    }
```

圖二十：小任務程式碼 1

```
function present(m){
    if(m==1){
        taskcomplete++;
        for(var i=0;i<5;i++){
            if(myteam[i].space==true){
                document.getElementById(`myteamblood${i+1}`).innerHTML=`<label class="change2">`+myteam[i].blood+`+200</label>`;
                myteam[i].blood=myteam[i].blood+200;
                if(attack[product.mode-1].state==1){
                    bonus=200;
                }else{
                    bonus=400;
                }

                setTimeout(`document.getElementById("myteamblood${i+1}").innerHTML=${myteam[i].blood}`,500);
            }
        }
    }else if(m==2){
        for(var i=0;i<5;i++){
            if(enemy[i].space==true){
                document.getElementById(`enemyblood${i+1}`).innerHTML=`<label class="change">`+enemy[i].blood+`-${enemy[i].blood/2}</label>`;
                enemy[i].blood=Math.round(enemy[i].blood/2);
                setTimeout(`document.getElementById("enemyblood${i+1}").innerHTML=${enemy[i].blood}`,500);
            }
        }
    }else if(m==3){
        if(fightsec>=7)fightsec=fightsec-5;
    }else if(m==4){

    }else if(m==5){

    }
    document.getElementById("Qtd").innerHTML=`<img src="./picture/problem/Q/Q0.png" id="Qimg" class="QP">`
    document.getElementById("Ptd").innerHTML=`<img src="./picture/problem/P/P0.png" id="Pimg" class="QP">`
    p=0;
    q=0;
    taskcheck=false;
}
```

圖二十一：小任務程式碼 2

伍、研究結果與討論

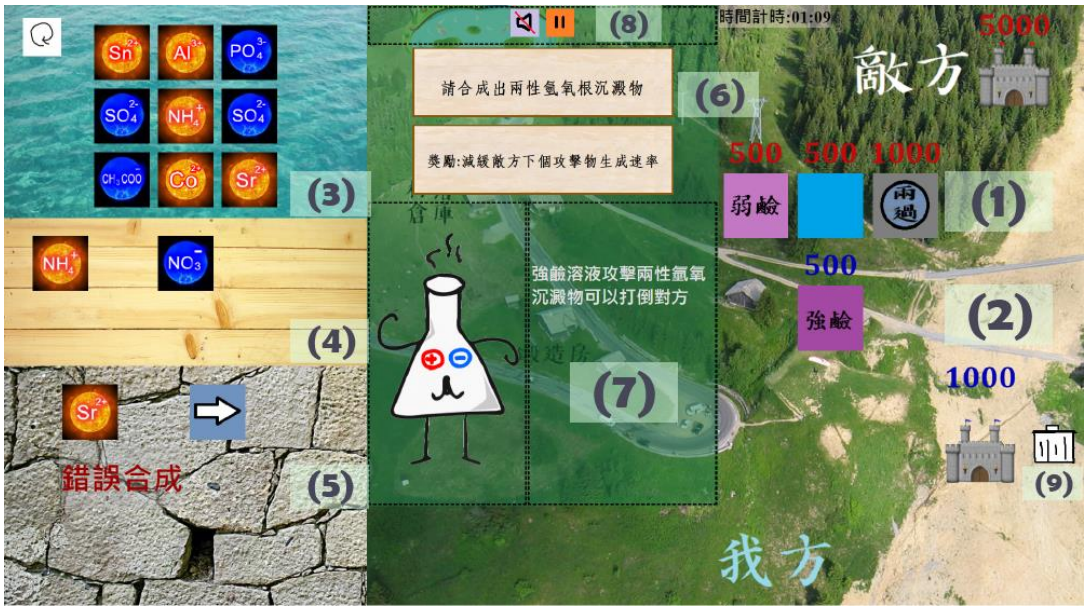
一、遊戲成果

(一)遊戲簡介

化學大戰爭是一款以策略型攻防遊戲。玩家必須活用化學沉澱表及其他化學知識，來進攻敵方。

(二)畫面介紹

1.戰鬥畫面



圖二十二：化學大戰爭戰鬥畫面

(1) **主要地圖的敵方陣營**。玩家可以在此處看到敵方派出的士兵和敵方的堡壘，並且對其施以攻擊。

(2) **主要地圖的我方陣營**。玩家能在此處看到自己派出的士兵，並指揮其攻擊敵方。

(3) **離子池**。用於蒐集離子球。此處每過約莫一兩秒就會自動生成離子球，點擊即可蒐集至我方村落倉庫。

(4) **村落倉庫**。用於存放被蒐集的離子球。點擊即可送至鍛造房。

(5) **鍛造房**。用於合成沉澱物或是酸鹼溶液。合成後點擊產物即可派上戰場。

(6) **小任務**。玩家若有能力在遊玩時順便完成小任務，即可獲得隨機獎勵。(小任務可做可不做)

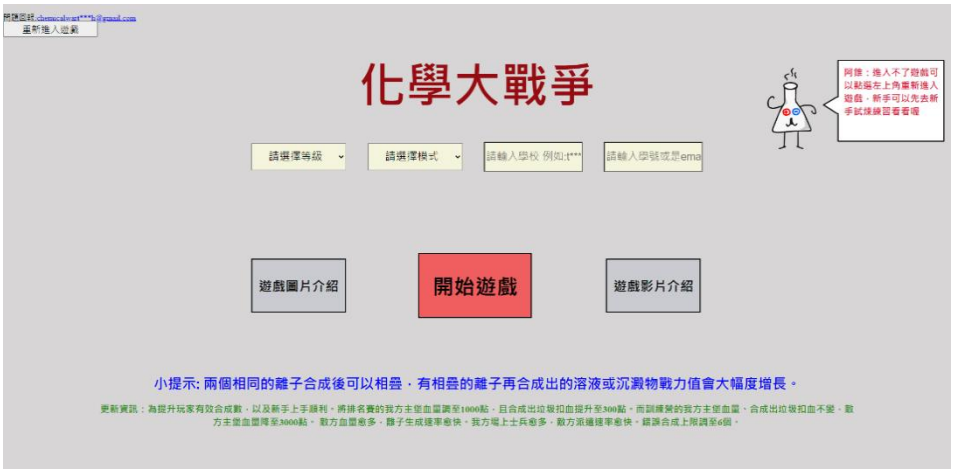
(7) **小提示**。系統會依據目前場上的布局狀況來對玩家提出最有利的提示，以方便遊戲進行。

(8) **遊戲暫停鍵**。點擊此處遊戲暫停，也可以順便查看暫停畫面附的化學沉澱表。

(9) **丟棄我方場上士兵**。點擊指定我方士兵後再點下此鍵，可以刪除我方士兵。

2.主畫面

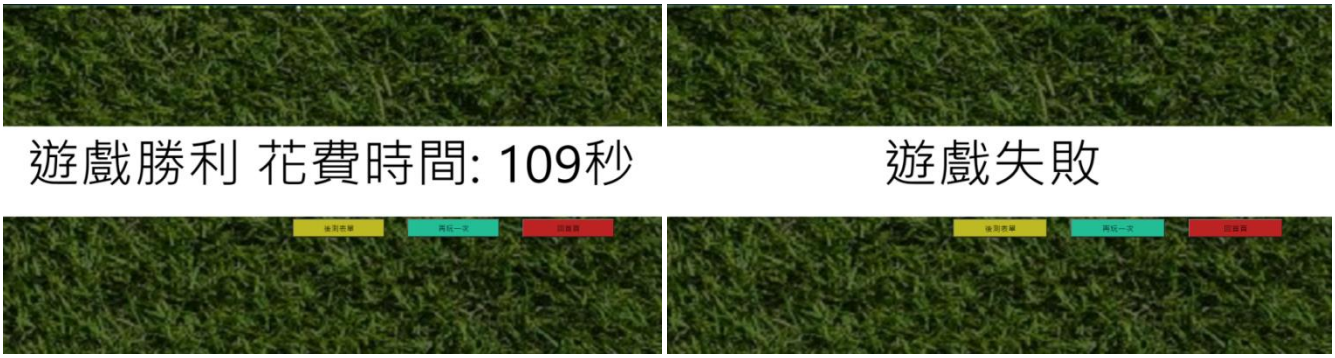
此為主畫面。可以在選單選擇要遊玩的遊戲模式，並輸入完學號和學校代碼後，即可進行遊戲。畫面亦有更新資訊，以及規則介紹等按鈕。



圖二十三：遊戲主畫面

3.結束畫面

此為結束畫面。分為勝利及失敗。

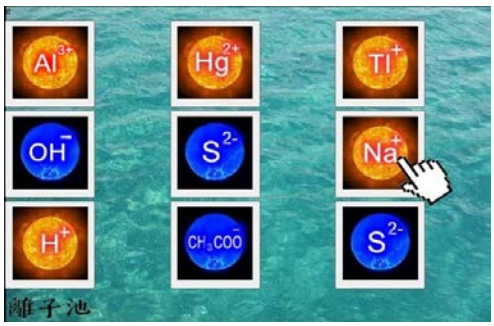


圖二十四：計時排名賽勝利畫面圖　　圖二十五：遊戲失敗畫面

(三)規則介紹(遊戲分為計時和計分兩種)

1.通用規則

(1)玩家首先要在離子池裡選取要用到的離子球，即可送到村落倉庫。



圖二十六之一：離子池

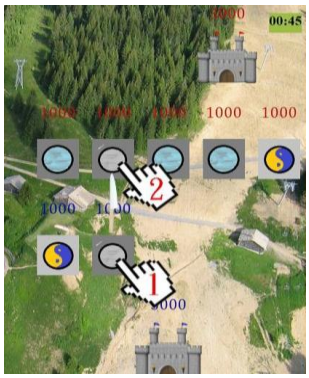
(2)接著再從村落倉庫裡選取兩顆離子球，搬送到鍛造房進行合成。

圖二十六之二：村落倉庫

(3)按下箭頭即可合成鍛造房裡的離子球。再點擊生成物將其派至戰場上。

圖二十六之三：鍛造房

(4)先點擊指派自己的士兵，再點擊欲攻擊的敵兵，即可完成一次攻擊，士兵的數值會互相抵消(普遍狀況下)。



圖二十六之四：指揮我方士兵進行攻擊

敵方會時不時派出士兵進攻。若玩家想傷害到敵方主堡，則得先將敵方式兵全部剷除，才能對主堡造成傷害。

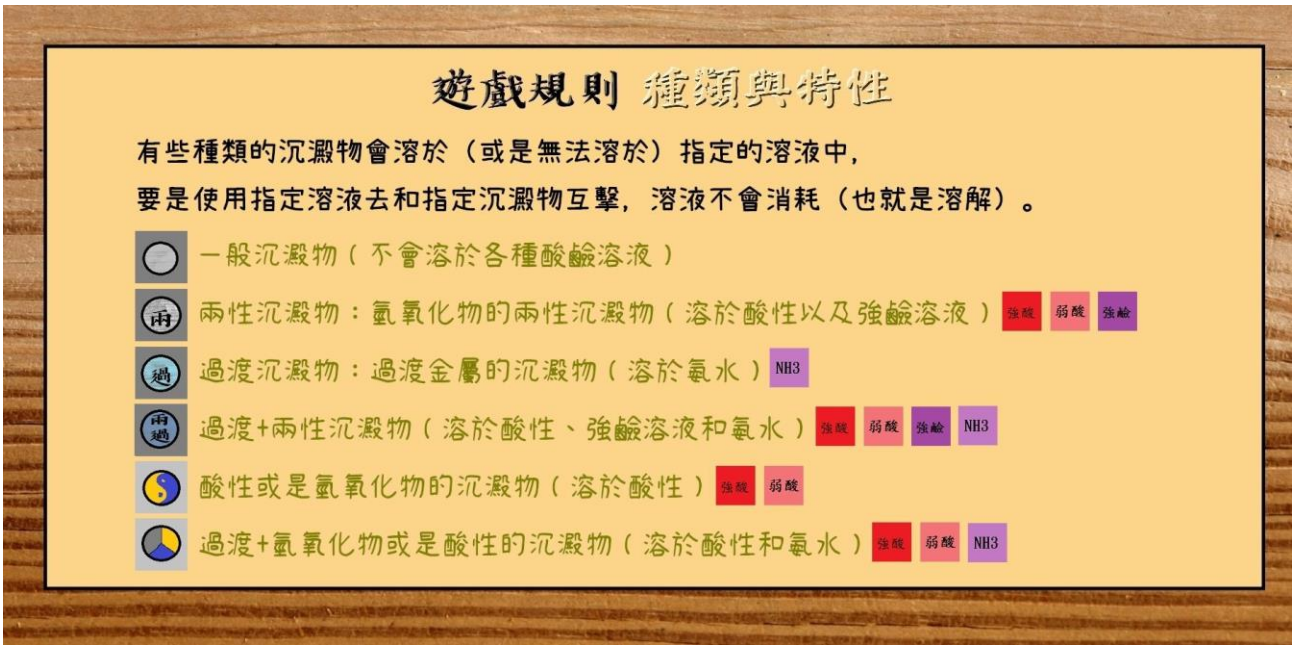
(5)合成：兩個相同的離子合成後可以相疊，相疊的離子再合成出的產物戰力更高；兩個無法合成的離子若是被合成，則會產出垃圾，每次合成扣自家堡壘 200 分。

(6)一般情況下，各類基礎沉澱物戰力值為 1000；溶液則為 500。各類沉澱物互相攻擊則會互相抵銷。沉澱物無法攻擊溶液。

(7)酸鹼能攻擊部分沉澱物。酸不能打酸、鹼不能打鹼。同強度酸鹼互相攻擊會抵消。

弱酸攻擊強酸鹼時，僅能消耗酸鹼一半的戰力值。酸鹼被水攻擊時，只需消耗酸鹼一半的戰力值即可擊潰水。水可以攻擊酸鹼(只能消耗酸鹼一半的戰力值)，但不能攻擊水或是各類沉澱物。氨水在攻擊溶液類時，視為弱鹼，也可以攻擊部分沉澱物。

(8)進階模式特殊規則(包括兩性元素和過渡元素)



圖二十七：進階規則

2.計時模式

玩家必須想盡辦法在最短的時間內打擊敵方並使其城堡血量歸零。在敵方堡壘血量歸零之前，敵方陣營會不斷地派出兵力。此模式為我們首先製作的遊戲模式，也是主打的遊戲模式。

3.計分模式

計時兩分鐘內，敵方陣營會不斷地派出兵力。玩家必須在時間內想盡辦法打擊敵方堡壘，敵方堡壘受到的傷害會從數值 0 慢慢往上累計，最後時間截止所攻擊堡壘的傷害量即為分數。此遊戲沒有獲勝，但有失敗（我方堡壘基礎血量為 1000，亂合成會導致自家堡壘扣血）。

4.基礎/進階賽

有鑑於每人化學能力不同，因此我們特別設計基礎模式與進階模式。其中基礎模式的沉澱物只有一種：一般沉澱物，但進階賽沉澱物則細分多種(見遊戲規則)，多種沉澱物則又多了其他規則。另外，進階模式的敵方派兵速度也比較快。

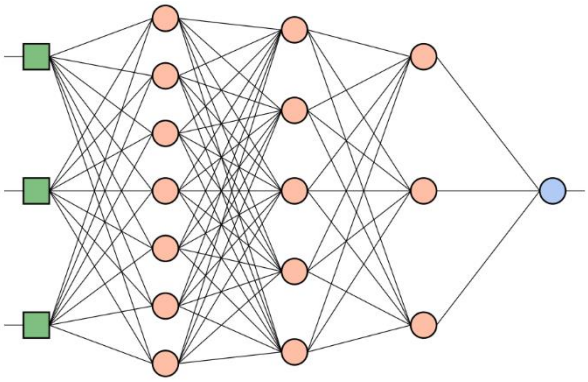
5.新手試煉模式

在做數據收集時，發現玩家也懶得點開遊戲規則介紹圖一一閱讀的狀況，因此我們回家設計出了新的新手模式。在這個模式裡，會有吉祥物帶領玩家一步步介紹遊戲操作，並且請玩家練習操作。做完基本的遊戲和操作介紹即可通關。

二、數據分析

(一) 簡介

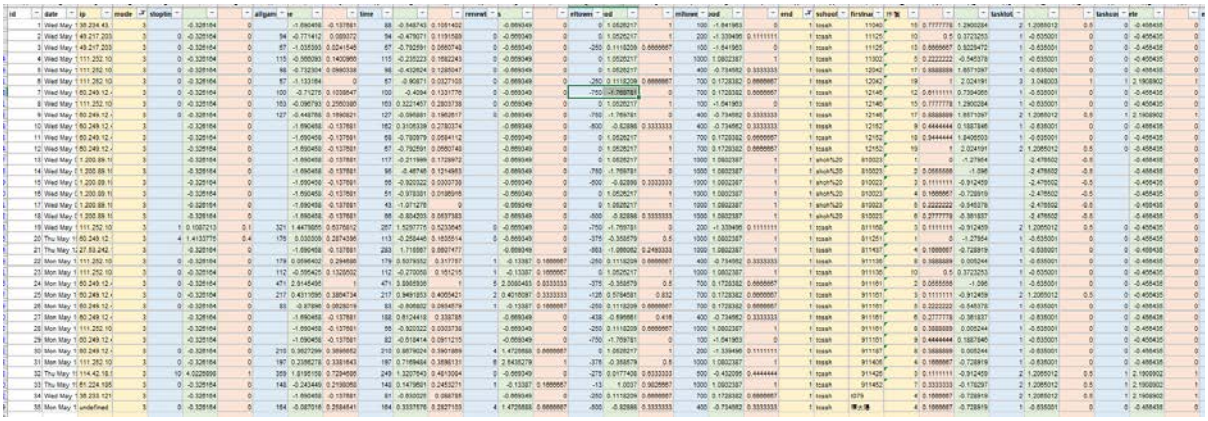
在研究的過程，我們找了兩個班級的學弟妹幫忙測試前測、遊玩遊戲、後測整個流程，在這之中，獲得了大量的統計數據。之後，我們藉由這些數據放入 Brain.js 進行分析，去發揮 AI 的預測能力，進而讓之後的學生可以預測進步幅度。



圖二十八：類神經網路示意圖

(二)研究步驟

1.步驟一



圖二十九：數據規模展示

從資料庫中，找出合適的資料，並將其正規化。

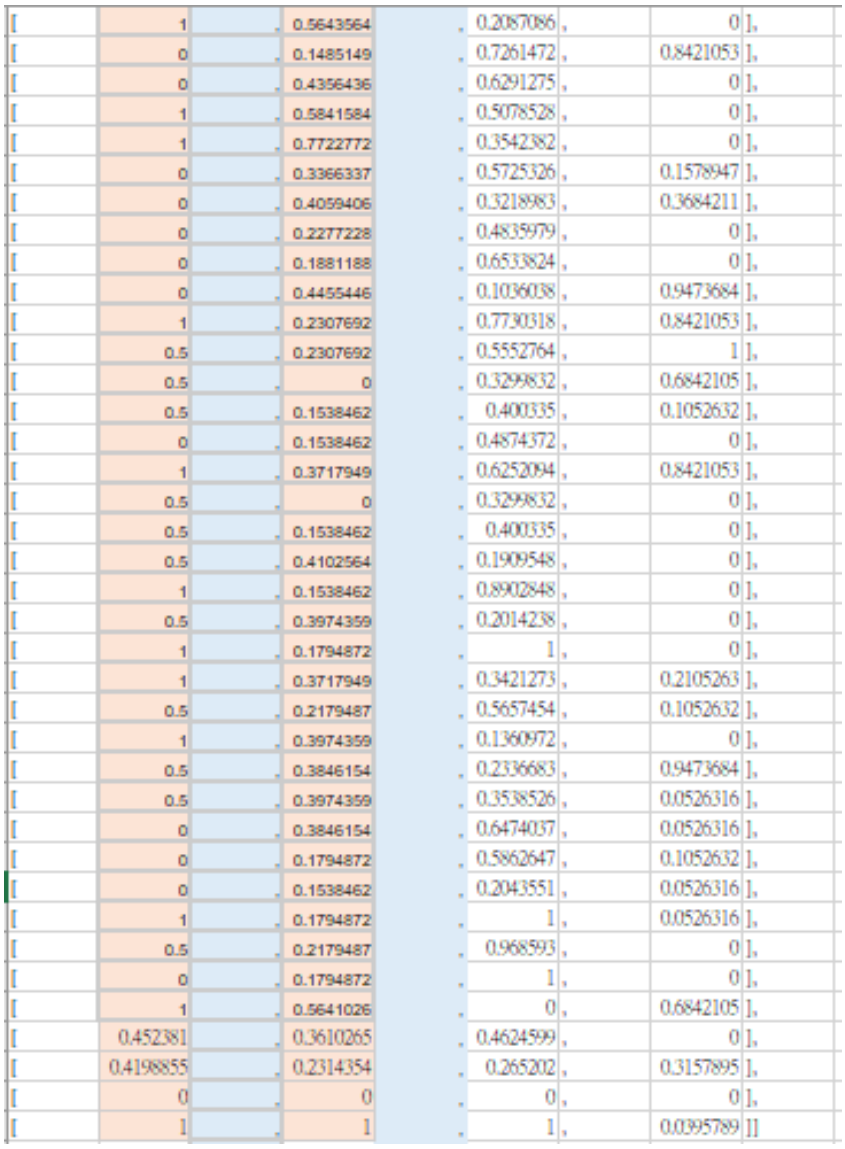
Input1	Input2	Input3	Output1
該玩家遊戲場次序號	前測分數	班上化學排名	該場次遊戲

表二：game 模型

Input1	Input2	Input3	Output1
遊玩總場次	前測分數	班上化學排名	進步幅度

表三：form 模型

2.步驟二



```
[ 1 , 0.5643564 , 0.2087086 , 0 ],
[ 0 , 0.1485149 , 0.7261472 , 0.8421053 ],
[ 0 , 0.4356436 , 0.6291275 , 0 ],
[ 1 , 0.5841584 , 0.5078528 , 0 ],
[ 1 , 0.7722772 , 0.3542382 , 0 ],
[ 0 , 0.3366337 , 0.5725326 , 0.1578947 ],
[ 0 , 0.4059406 , 0.3218983 , 0.3684211 ],
[ 0 , 0.2277228 , 0.4835979 , 0 ],
[ 0 , 0.1881188 , 0.6533824 , 0 ],
[ 0 , 0.4455446 , 0.1036038 , 0.9473684 ],
[ 1 , 0.2307692 , 0.7730318 , 0.8421053 ],
[ 0.5 , 0.2307692 , 0.5552764 , 1 ],
[ 0.5 , 0 , 0.3299832 , 0.6842105 ],
[ 0.5 , 0.1538462 , 0.400335 , 0.1052632 ],
[ 0 , 0.1538462 , 0.4874372 , 0 ],
[ 1 , 0.3717949 , 0.6252094 , 0.8421053 ],
[ 0.5 , 0 , 0.3299832 , 0 ],
[ 0.5 , 0.1538462 , 0.400335 , 0 ],
[ 0.5 , 0.4102564 , 0.1909548 , 0 ],
[ 1 , 0.1538462 , 0.8902848 , 0 ],
[ 0.5 , 0.3974359 , 0.2014238 , 0 ],
[ 1 , 0.1794872 , 1 , 0 ],
[ 1 , 0.3717949 , 0.3421273 , 0.2105263 ],
[ 0.5 , 0.2179487 , 0.5657454 , 0.1052632 ],
[ 1 , 0.3974359 , 0.1360972 , 0 ],
[ 0.5 , 0.3846154 , 0.2336683 , 0.9473684 ],
[ 0.5 , 0.3974359 , 0.3538526 , 0.0526316 ],
[ 0 , 0.3846154 , 0.6474037 , 0.0526316 ],
[ 0 , 0.1794872 , 0.5862647 , 0.1052632 ],
[ 0 , 0.1538462 , 0.2043551 , 0.0526316 ],
[ 1 , 0.1794872 , 1 , 0.0526316 ],
[ 0.5 , 0.2179487 , 0.968593 , 0 ],
[ 0 , 0.1794872 , 1 , 0 ],
[ 1 , 0.5641026 , 0 , 0.6842105 ],
[ 0.452381 , 0.3610265 , 0.4624599 , 0 ],
[ 0.4198855 , 0.2314354 , 0.265202 , 0.3157895 ],
[ 0 , 0 , 0 , 0 ],
[ 1 , 1 , 1 , 0.0395789 ]]
```

圖三十：數據檔案轉換

在 excel 中編輯成二維陣列轉換成 prn 檔，複製文字檔到程式碼中。

3.步驟三

將資料庫中的三個資料除外當作測試誤差用，再將剩下的資料進行機器學習，機器學習完後把三個資料隨機一個進行測試，最後將模擬出來的數據儲存到後端內。每個模型測試 3000 次當作分析的數據。

4.步驟四

分析每個模型的誤差值，包含 R-square、誤差小於 0.1 的比例、MAE、MSE 等並製作成表格呈現。

(三)程式片段

1.總資料量

```
const data = [[           0      , 0.336633      , 0.180555,       0.192538],
[            0      , 0.554455      , 0.152777,       0.532107],
[            0      , 0.811881      , 0.041666,       0.006584],
[            0      , 0.623762      , 0.138888,       0.475512],
[            1      ,          1    , 0.138888,       0.168283],
[            0      , 0.435643      , 0.291666,       0.612957],
[          0.5      , 0.227722      ,          0,     0.321898],
[            1      ,          1    , 0.166666,       0.329983],
[            1      , 0.564356      , 0.166666,       0.208708],
[            0      , 0.148514      , 0.138888,       0.726147],
[            0      , 0.435643      , 0.291666,       0.629127],
[            1      , 0.584158      , 0.097222,       0.507852],
[            1      , 0.772277      , 0.222222,       0.354238],
[            0      , 0.336633      , 0.083333,       0.572532],
[            0      , 0.405940      , 0.027777,       0.321898],
[            0      , 0.227722      , 0.138888,       0.483597],
[            0      , 0.188118      , 0.180555,       0.653382],
[            0      , 0.445544      , 0.041666,       0.103603],
[            1      , 0.230769      , 0.236111,       0.773031],
[          0.5      , 0.230769      ,      0.125,     0.555276],
[          0.5      ,          0    , 0.055555,       0.329983],
[          0.5      , 0.153846      , 0.194444,       0.400335],
[            0      , 0.153846      ,      0.125,     0.487437],
[            1      , 0.371794      ,       0.25,     0.625209],
[          0.5      ,          0    , 0.166666,       0.329983],
[          0.5      , 0.153846      , 0.152777,       0.400335],
[          0.5      , 0.410256      , 0.180555,       0.190954],
[            1      , 0.153846      , 0.069444,       0.890284],
[          0.5      , 0.397435      , 0.194444,       0.201423],
[            1      , 0.179487      ,          0,             1],
[            1      , 0.371794      ,          0,     0.342127],
[          0.5      , 0.217948      , 0.152777,       0.565745],
[            1      , 0.397435      , 0.111111,       0.136097],
[          0.5      , 0.384615      , 0.027777,       0.233668],
[          0.5      , 0.397435      , 0.111111,       0.353852],
[            0      , 0.384615      ,          1,     0.647403],
[            0      , 0.179487      , 0.097222,       0.586264],
[            0      , 0.153846      , 0.097222,       0.204355],
[            1      , 0.179487      , 0.291666,             1],
[          0.5      , 0.217948      , 0.138888,       0.968593],
[            0      , 0.179487      , 0.097222,             1],
[            1      , 0.564102      , 0.291666,             0]]
```

圖三十一：總示範資料

2.設定模擬資料

```
for(let i=0;i<data.length-3;i++){
    while(a==location||b==location||c==location)
        location=location+1;
    arr2[i]=location;
    location=location+1;
}
```

圖三十二：設定模擬資料程式碼

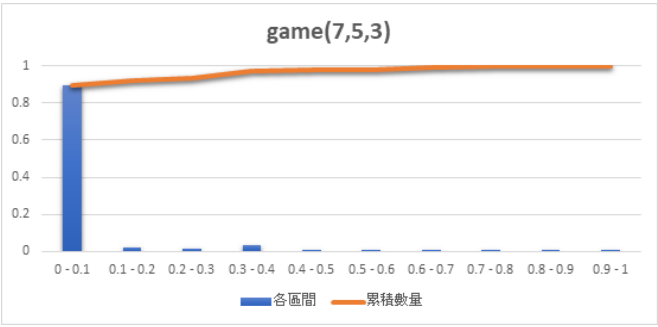
3.回傳模擬值

```
output = net.run([data[a][0],data[a][1],data[a][2]]);
output = net.run([data[b][0],data[b][1],data[b][2]]);
output = net.run([data[c][0],data[c][1],data[c][2]]);
```

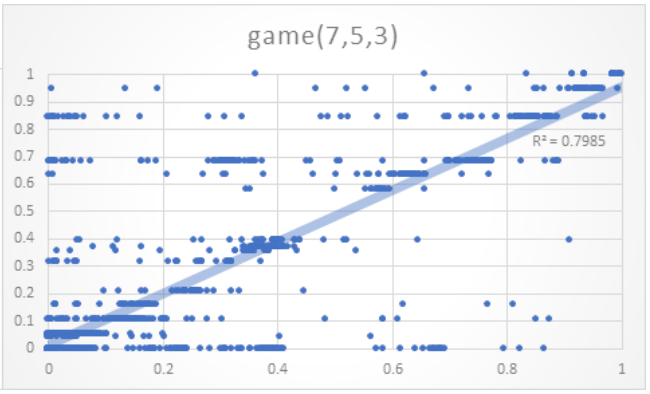
圖三十三：回傳模擬值程式碼

(四)各種 hiddenlayer 執行出的數據分布圖（最差與最好的模型對比）

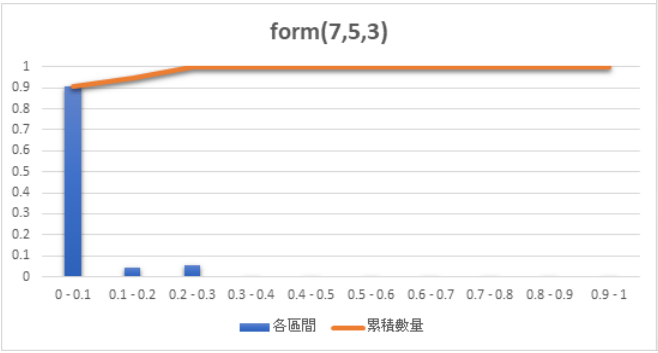
1.較好模型



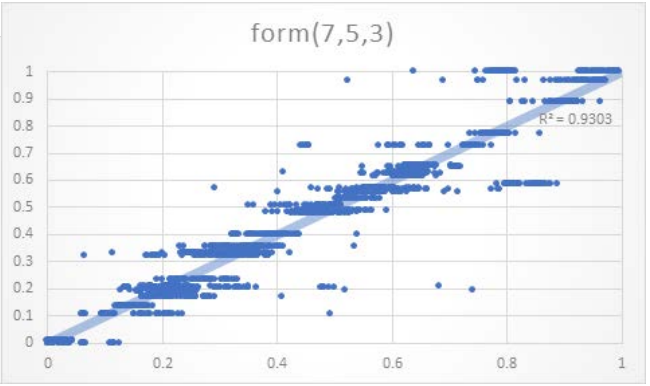
圖三十四：(7,5,3)誤差值各區間數量與累積數量組合圖



圖三十五：(7,5,3)各數據預測與真值分布圖

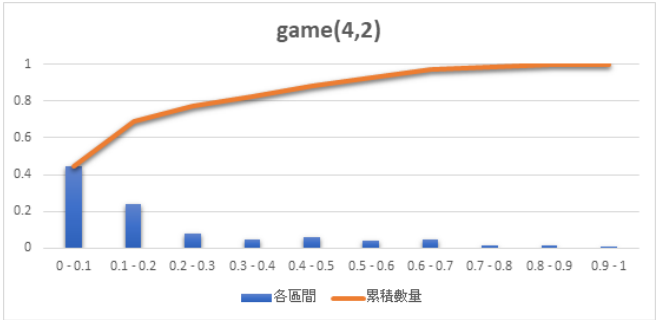


圖三十六：(7,5,3)誤差值各區間數量與累積數量組合圖

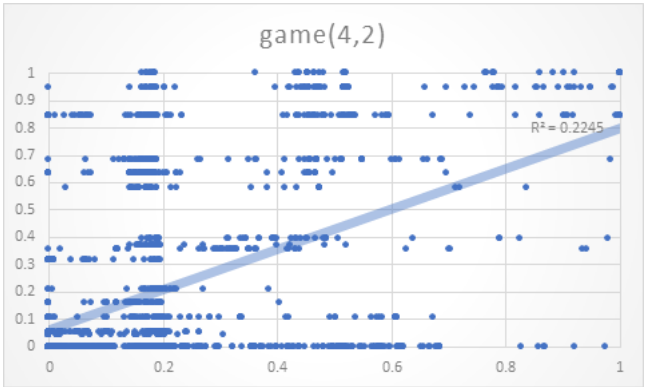


圖三十七：(7,5,3) 各數據預測與真值分布圖

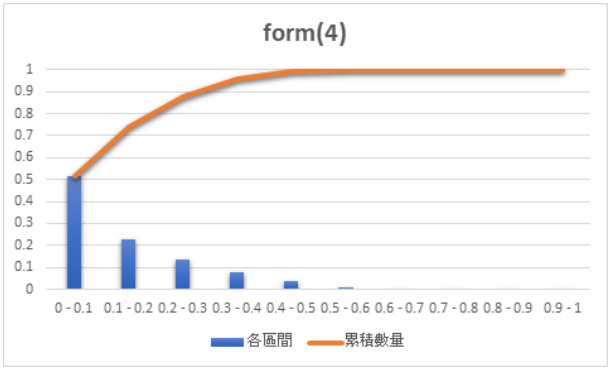
2.較差模型



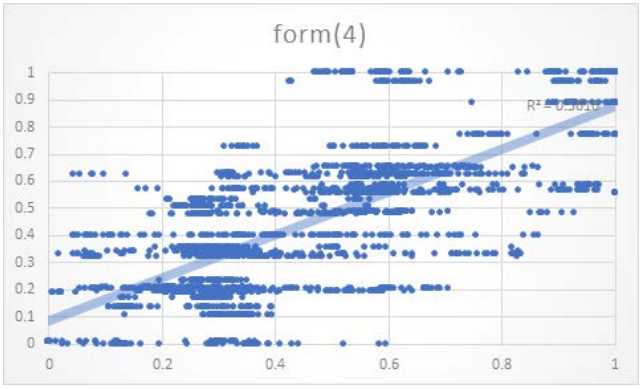
圖三十八：(4,2)誤差值各區間數量與累積數量組合圖



圖三十九：(4,2) 各數據預測與真值分布圖



圖四十：(4)誤差值各區間數量與累積數量組合圖



圖四十一：(4) 各數據預測與真值分布圖

(五)結論

R-square

hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.561638	0.653448	0.857455	0.642232	0.857909	0.918396	0.737597	0.930334
game	0.395313	0.53808	0.738017	0.22446	0.46886	0.765712	0.430023	0.798515

表四：各模型 R-square

誤差小於 0.1 的比例

hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	51.5%	59.8%	83.1%	66.7%	83.2%	87.1%	79.1%	90.4%
game	57.3%	71.0%	81.9%	44.8%	58.2%	83.8%	46.5%	89.6%

表五：各模型誤差小於 0.1 的比例

平均誤差值 MAE

hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.133629	0.115052	0.056018	0.098524	0.055340	0.041869	0.072411	0.039388
game	0.150040	0.116657	0.069305	0.188986	0.143686	0.067659	0.154539	0.052197

表六：各模型平均誤差值 MAE

平均平方誤差 MSE

hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.033607	0.027751	0.010271	0.02556	0.010013	0.005846	0.018453	0.005101
game	0.069008	0.052652	0.027892	0.079752	0.054849	0.024662	0.054017	0.019262

表七：各模型平均平方誤差 MSE

在數據分析中，R 大於 0.7 就算是高度相關了，以 R-square 來看也就是大於 0.5，而表中大多的 R-square 大多呈高度相關，form(7,5,3)的 R-square 甚至可達到 0.93 的數值。

在此圖表可以看出不論是 form 還是 game 的預測模型在誤差小於 0.1 的比例都有近 9

成的準確度，相當可觀。

在平均誤差值中，幾乎每個模型都有在 0.1 以內，且表現最佳的(7,5,3)更是只有 0.04、0.05 的誤差。

在以上圖表分析過後可以發現越多層的模型越好，越多節點的模型越佳，且兩者模型的 hiddenlayer(7,5,3)可以對數據的分析達到有效的預測。

三、討論問題

(一)遊戲設計與平衡類

當初合成出回收物是不會扣血的，但也讓玩家較不會注意此細節，只會繼續合成下一個攻擊物。因此我們新增合成回收物扣血 300 點的規則，主堡血量從 2000 點降至 1000 點。

此外，我們希望玩家可以更有效的合成，調整成在計時賽與訓練營的敵方生成速率(隨著敵方主堡剩餘血量而變化)。

而在對戰的部分，從一開始的固定每 30 秒生成一次敵方士兵，改為當我方攻擊物愈多則敵方攻擊物生成速率愈快；敵方攻擊物愈多，敵方攻擊物生成速率愈慢。

有這些調整的原因如下，第一個調整是為了讓玩家不要囤積酸性溶液等著無損攻擊特定沉澱物，第二個調整是為了讓玩家等到敵方攻擊物愈來愈多時有更大的翻盤機會。

(二)表單類

關於題目是否能代表學生的化學能力，我們特地向請化學老師幫我們的測試題目做調整和修正，將每題的題型、方向、簡易度控制在同一個標準，足以直接測試受試者該單元化學能力。

(三)離子池生成機率問題

原先離子池的生成離子球是隨機的，但由於正離子基本數量較多，很容易有整版面都是正離子的問題。因此我們設定將離子池版面生成出的陰陽離子球固定在 5：4，防止以上問題發生。

(四)溶液攻擊力調整

對學生而言，看到氫離子或氫氧根都能很直覺地合成出酸鹼溶液，但是我們的學習主軸是沉澱表，為了避免這種狀況發生，我們在之後的更新把酸鹼溶液的攻擊力大幅下修，以提升沉澱物被合成的機會。

四、未來展望

這次研究固然有許多缺失，我們希望可以在各方面獲得更多突破。

(一)數據太少

有鑑於這次的研究數據分母太低、知道這個遊戲的人太少，下次發行遊戲希望可以投放到能吸引更多觸及率的管道，例如找學校合作加強宣傳、找圖書館主任老師討論能否在電腦閱覽區張貼通知或網頁釘選連結等等。

(二)前測與後測時間間隔太短

由於發行於大眾的時辰過於急促，下次發行遊戲希望能夠拉長前測與後測間隔的時間，給予較為充足的第二階段回饋，比較能得到更多完整的數據。

(三)遊戲延伸

這類的卡牌對抗遊戲還有很多值得延伸的空間，希望未來可以添加更多離子或者添加更多符合化學特性的規則。

(四)人工智慧學習

希望我們未來有能力利用機器學習，去分析該玩家的弱點，並針對此弱點去設立任務，或是在遊戲結束後顯示建議準備方向。先讓玩家遊玩一場，藉由大家第一場的遊戲數據去分析此玩家的弱點為何，而在之後的遊玩場次中，使用大家進步幅度的數據去分析此玩家是否已達到解決弱點，並加強其對沉澱表的熟悉程度。

陸、結論

透過三個面向的玩家反饋：前測表單和後測表單的結果比較、玩家由玩遊戲的後台數據顯示的進步程度（通關耗費時間變少）、玩家口頭反饋，我們可以得知得有：

(一)這次的遊戲製作確實能使受試者比起死背課本，更享受遊戲同時帶來的樂趣，因此能推斷此遊戲對化學能力的輔助稍有成效。

(二)一大部分數據在多次遊玩後，通關耗費時間下降許多，因此能知道這次遊戲能夠使玩家對離子其合成產物更加熟悉。

(三)在機器學習的部分找出了最佳的 hiddenlayer 層數以及節點數，利用這兩個模型可以去分析出未來玩家的學習狀況以及進步幅度。game 模型可以讓玩家知曉自己玩到第幾場時，應該要獲得如何的成績，而 form 模型可以方便玩家知道自己想進步多少，需要遊玩多少場的訓練。

柒、參考資料及其他

一、采風設計苑（2010）·輕鬆學網路設計三合一·松崗出版社

二、吳宗謀（2006）·多媒體與網頁設計·文魁資訊

三、陳峰棋（2003）·Visual Basic 網路程式設計·全華出版社

四、Kazuhiro FURUHATA（2001）·最新 HTML 語法參考辭典·博碩文化出版社

五、Ryan Lu（2018）·Evaluation Metrics: 迴歸模型·取自

https://medium.com/ai%E5%8F%8D%E6%96%97%E5%9F%8E/evaluation-metrics-%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B-807c04871bb4

六、Brain.js·W3.CSS.·取自

https://www.w3schools.com/ai/ai_brainjs.asp

七、Brain.js·取自

https://brain.js.org/#/

八、CwC 頻道（2021 年 9 月 13 日）·【EP1】上傳「表單」資料到 Google 試算表·取自

http://promooo.info/cwccontent.php?pid=105

九、神 Q 超人（2017 年 10 月 11 日）·[筆記][JavaScript]取得網址後帶的參數(QueryString)·取自

https://ithelp.ithome.com.tw/articles/10190254

十、DelftStack（2021 年 10 月 2 日）·JavaScript 中的倒數計時器·取自

https://www.delftstack.com/zh-tw/howto/javascript/count-down-timer-in-javascript/

【評語】052509

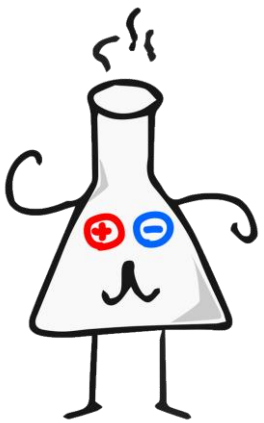
本作品有鑑於高中化學沉澱表學習困難，進而製作了一款網頁遊戲，期望提升沉澱表熟悉度的遊戲。此想法新穎有趣，遊戲過程中的資料也可以作為評估學生學習成效的一種指標。

報告內雖有遊戲方式的說明，但並未詳述為何這樣的遊戲過程可以幫助化學學習。建議實驗的設計要對準研究主題，例如：有玩遊戲與沒玩遊戲的群體，兩組的沉澱表後測是否有差異？抑或是透過遊戲的方式較容易長期記憶？以此方式設計實驗方能透過實驗驗證提出系統的成效。實驗中建議也可以記錄玩家對於此遊戲的主觀意見 (好玩程度、是否願意繼續玩、是否對記憶沉澱表有幫助等)，並當作後續的改善方向。

利用遊戲資料來分析學生學習成效的部分很有趣，建議更深入做實驗與反饋給遊戲 (例如：使用者在某些情況較易出錯，後續的遊戲會自動調整多出一些相關情境協助他快速熟悉)，更能展現此遊戲特別之處。

作品簡報

「澱」「資」的奧祕
利用網頁遊戲精進學生對沉澱表之研究



目錄

01 研究動機

每個理組的學生在高中時必定會經歷過「化學沉澱表」的折磨。不只要硬背起來，還要學會應用，簡直是選化III的最大瓶頸。

因此，有程式開發能力的我們，決定開發一款以化學沉澱表為核心的遊戲，再將各種化學特性規則寫入，希望能夠幫助往後的高中學生更容易度過這個難關。

02 研究目的

開發一款能夠讓學生更熟悉化學沉澱表、各種酸鹼的特性、各種陰陽離子特性且好玩的遊戲，並且實測受試學生前後化學觀念是否有進步、進一步預判出學生化學的成長幅度。

前言

研究過程

測驗方式

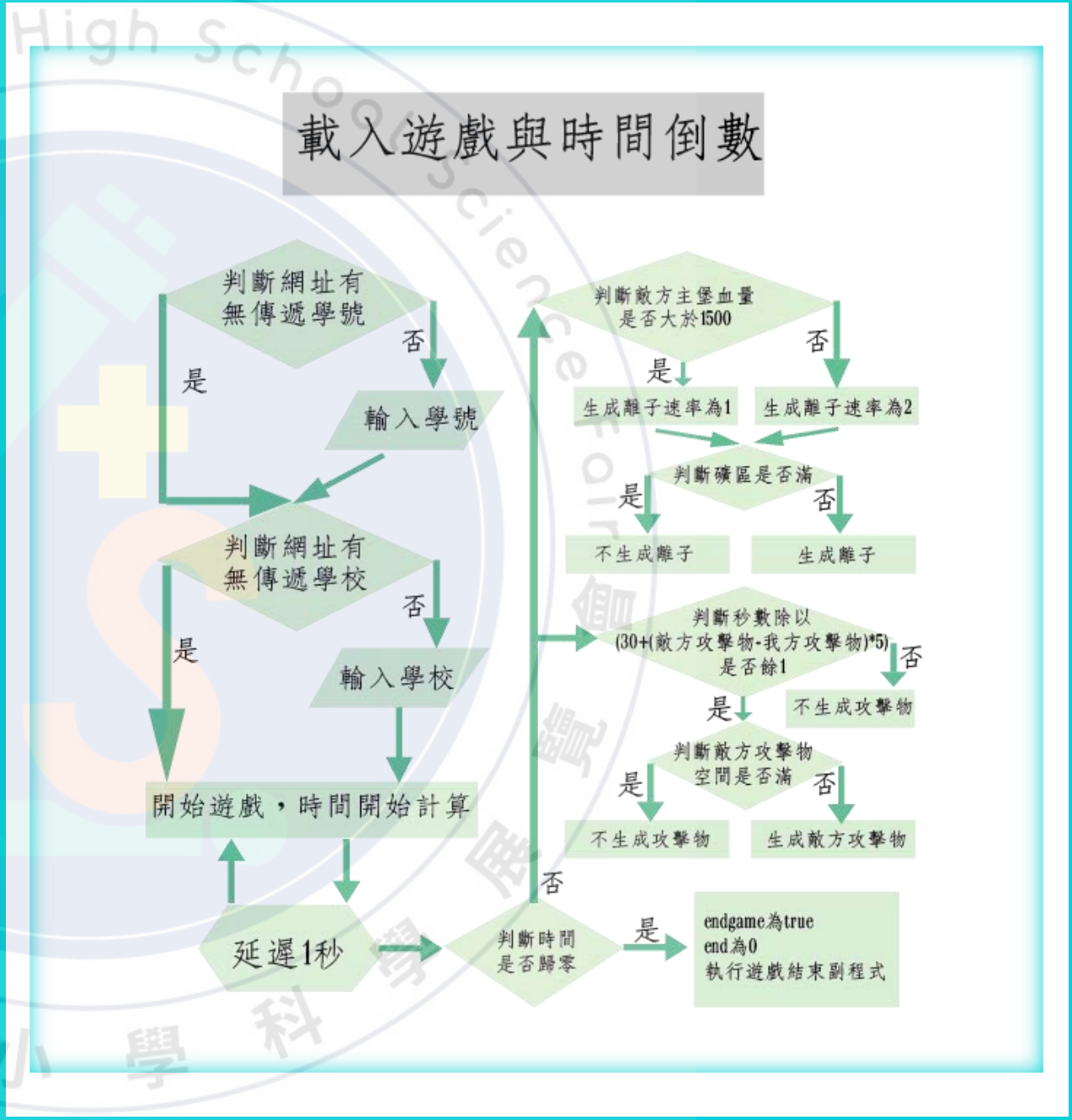
研究過程有三大部分，前測、遊玩過程、後測。分成ABC三份考卷，每份考卷有前測與後測，題目共有XYZ三個部分。每部分5個題型各一題。

	考卷A	考卷B	考卷C
前測	XY	XZ	YZ
後測	Z	Y	X

遊戲介紹

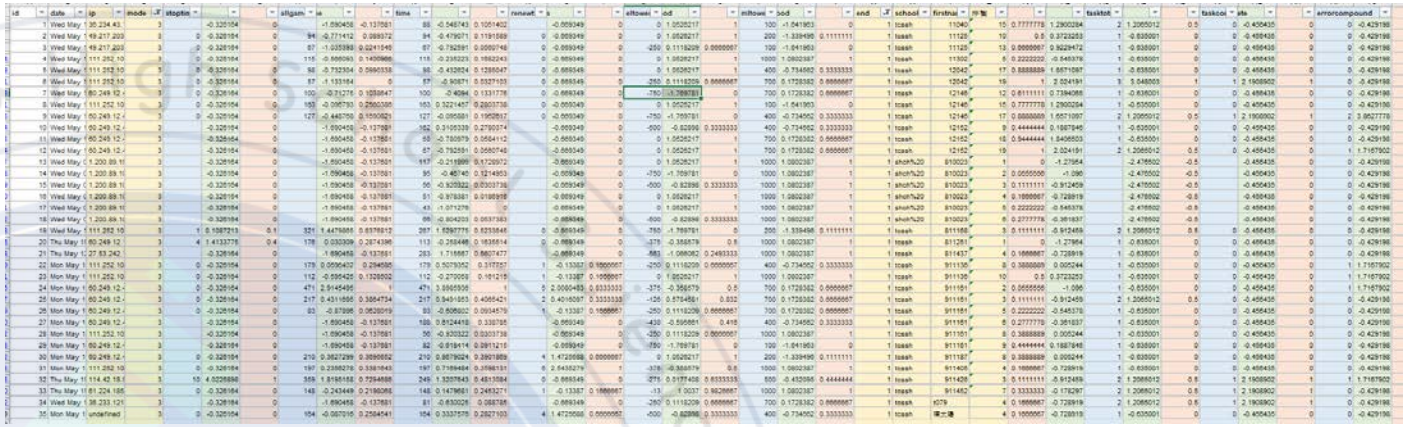
(1)主要地圖的敵方陣營
(2)主要地圖的我方陣營
(3)離子池
(4)村落倉庫
(5)鍛造房
(6)小任務
(7)小提示
(8)遊戲暫停鍵
(9)丟棄我方場上士兵

程式流程圖


載入遊戲與時間倒數
判斷網址有無傳遞學號
否
是
輸入學號
判斷網址有無傳遞學校
否
是
輸入學校
開始遊戲，時間開始計算
延遲1秒
判斷時間是否歸零
否
是
endgame為true
end為0
執行遊戲結束副程式
判斷敵方主堡血量是否大於1500
是
否
生成離子速率為1
生成離子速率為2
判斷礦區是否滿
是
否
不生成離子
生成離子
判斷秒數除以(30+(敵方攻擊物-我方攻擊物)*5)是否餘1
否
是
不生成攻擊物
判斷敵方攻擊物空間是否滿
是
否
不生成攻擊物
生成敵方攻擊物

(一)步驟一

從資料庫中，找出合適的資料，並將其正規化。

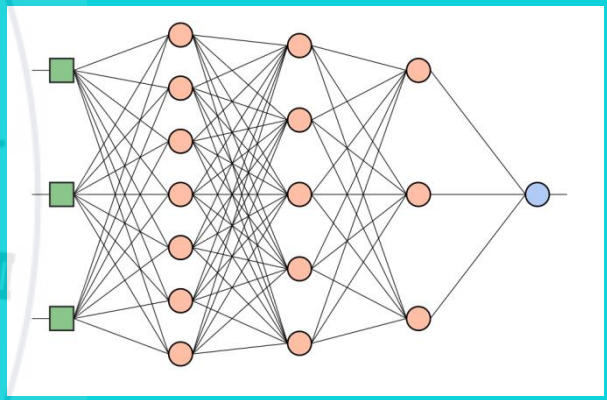


數據規模展示

數據分析

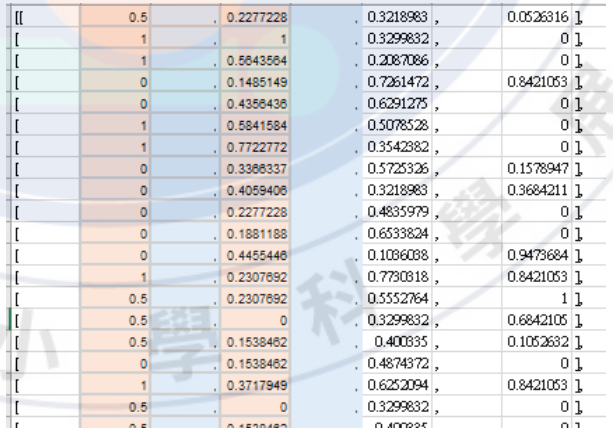
如何製作篇

在研究的過程，我們找了兩個班級的學弟妹幫忙測試前測、遊玩遊戲、後測整個流程，在這之中，獲得了大量的統計數據。之後，我們藉由這些數據放入Brain.js進行分析，去發揮AI的預測能力，進而讓之後的學生可以預測進步幅度。



(二)步驟二

在excel中編輯成二維陣列轉換成prn檔，複製文字檔到程式碼中。



[[0.5 , 0.2277228 , 0.3218983 , 0.0526316],
[1 , 1 , 0.3299832 , 0],
[1 , 0.5643564 , 0.2087086 , 0],
[0 , 0.1485149 , 0.7261472 , 0.8421053],
[0 , 0.4356436 , 0.6291275 , 0],
[1 , 0.5841584 , 0.5078528 , 0],
[1 , 0.7722772 , 0.3542382 , 0],
[0 , 0.3366337 , 0.5725326 , 0.1578947],
[0 , 0.4059406 , 0.3218983 , 0.3684211],
[0 , 0.2277228 , 0.4835979 , 0],
[0 , 0.1881188 , 0.6533824 , 0],
[0 , 0.4455446 , 0.1036088 , 0.9473684],
[1 , 0.2307692 , 0.7730818 , 0.8421053],
[0.5 , 0.2307692 , 0.5552764 , 1],
[0.5 , 0 , 0.3299832 , 0.6842105],
[0.5 , 0.1538462 , 0.400835 , 0.1052632],
[0 , 0.1538462 , 0.4874372 , 0],
[1 , 0.3717949 , 0.6252094 , 0.8421053],
[0.5 , 0 , 0.3299832 , 0],

數據檔案轉換

(三)步驟三

將資料庫中的三個資料除外當作測試誤差用，再將剩下的資料進行機器學習，機器學習完後把三個資料隨機一個進行測試，最後將模擬出來的數據儲存到後端內。每個模型測試3000次當作分析的數據。

step1:隨機三個數
step2:將隨機三個數除外的陣列位置進行分析
step3:將三個數其中的一個數測試模型所產生的output紀錄下

數據分析

如何製作篇

(四)步驟四

分析每個模型的誤差值。

$$R^2 = 1 - \frac{SSE}{SST}$$

$$SSE = \sum_{i=1}^{n} (y_i - y_i)^2$$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

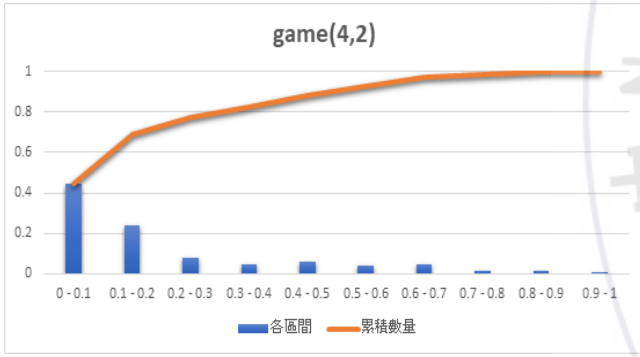
$$MSE = \frac{1}{n} \sum_{i=1}^{n} (f_i - y_i)^2$$

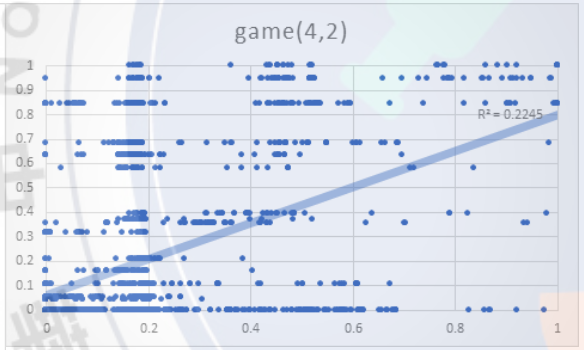
$$MAE = \frac{1}{n} \sum_{i=1}^{n} | f_i - y_i | = \frac{1}{n} \sum_{i=1}^{n} | e_i |$$

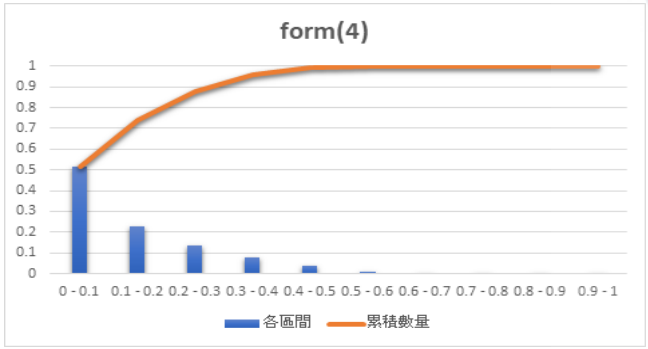
數據分析 如何製作篇

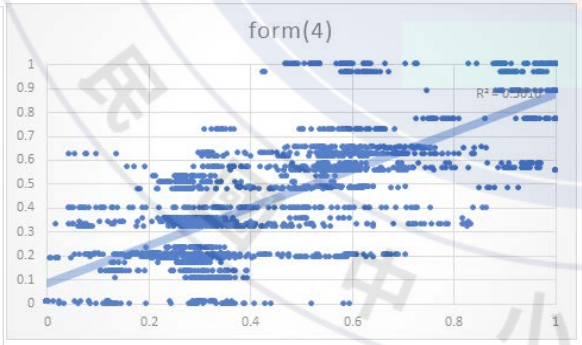
各種hiddenlayer執行出的數據分布圖
（最差與最好的模型對比）

較好模型

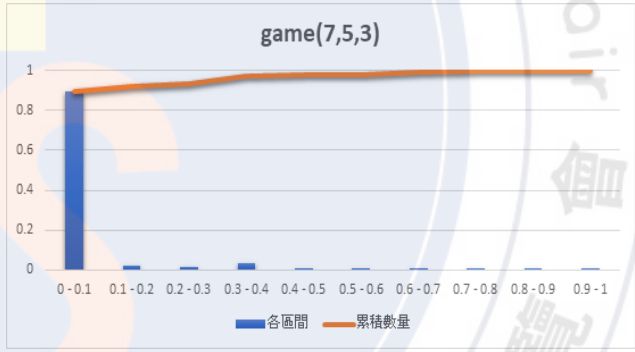


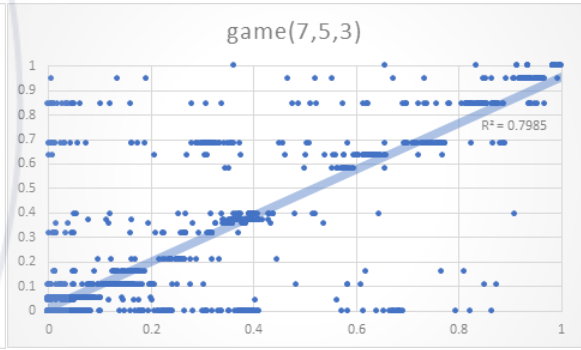


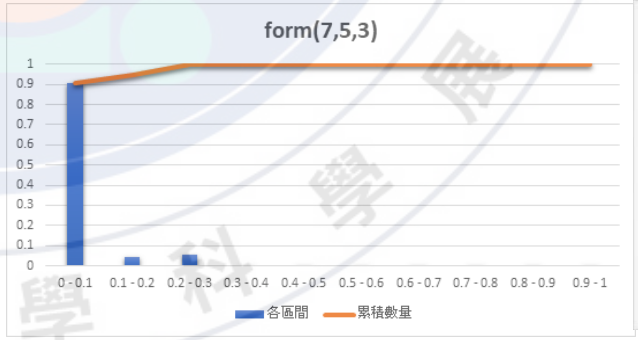


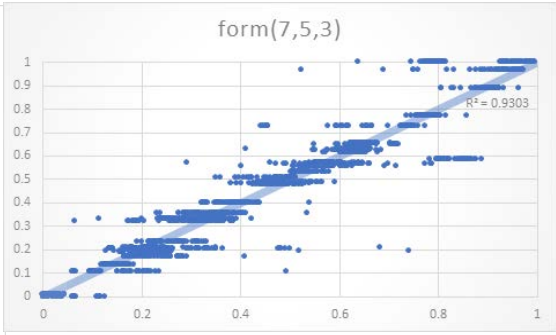


較差模型









數據分析結論篇

R-square								
hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.561638	0.653448	0.857455	0.642232	0.857909	0.918396	0.737597	0.930334
game	0.395313	0.53808	0.738017	0.22446	0.46886	0.765712	0.430023	0.798515

誤差小於0.1的比例								
hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	51.5%	59.8%	83.1%	66.7%	83.2%	87.1%	79.1%	90.4%
game	57.3%	71.0%	81.9%	44.8%	58.2%	83.8%	46.5%	89.6%

平均誤差值MAE								
hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.133629	0.115052	0.056018	0.098524	0.055340	0.041869	0.072411	0.039388
game	0.150040	0.116657	0.069305	0.188986	0.143686	0.067659	0.154539	0.052197

平均平方誤差MSE								
hiddenlayer	4	6	8	4,2	5,3	7,5	5,3,2	7,5,3
form	0.033607	0.027751	0.010271	0.02556	0.010013	0.005846	0.018453	0.005101
game	0.069008	0.052652	0.027892	0.079752	0.054849	0.024662	0.054017	0.019262

在此圖表可以看出不論是form或game的預測模型在誤差小於0.1的比例都有近9成的準確度，相當可觀。

在以上圖表分析過後可以發現越多層的模型越好，越多節點的模型越佳，且兩者模型的hiddenlayer(7,5,3)可以對數據的分析達到有效的預測。

未來展望

1

數據太少

希望可以投放到能吸引更多觸及率的管道，例如找學校合作加強宣傳。

2

前測與後測時間間隔太短

下次發行遊戲希望能夠拉長前測與後測間隔的時間，給予較為充足的第二階段回饋。

3

遊戲延伸

這類的卡牌對抗遊戲還有延伸空間，希望未來可以添加更多符合化學特性的規則。

4

人工智慧學習

希望我們未來有能力利用機器學習，去分析玩家的弱點，並針對此去設立任務。

結論

透過三個面向的玩家反饋：

(1)前測表單和後測表單的結果比較

(2)玩家由玩遊戲的後台數據顯示的進步程度（通關耗費時間變少）

(3)玩家口頭反饋

可玩性 ★★★★★

化學成績進步的同時享受遊戲帶來的樂趣

可學習性 ★★★★★

數據分析下可以發現玩家通關耗費時間下降許多

模擬成長路徑 ★★★★★

game模型可以讓玩家知曉自己玩到第幾場時，可獲得如何的成績

form模型可以方便玩家知道自己想進步多少，需要遊玩多少場的訓練。