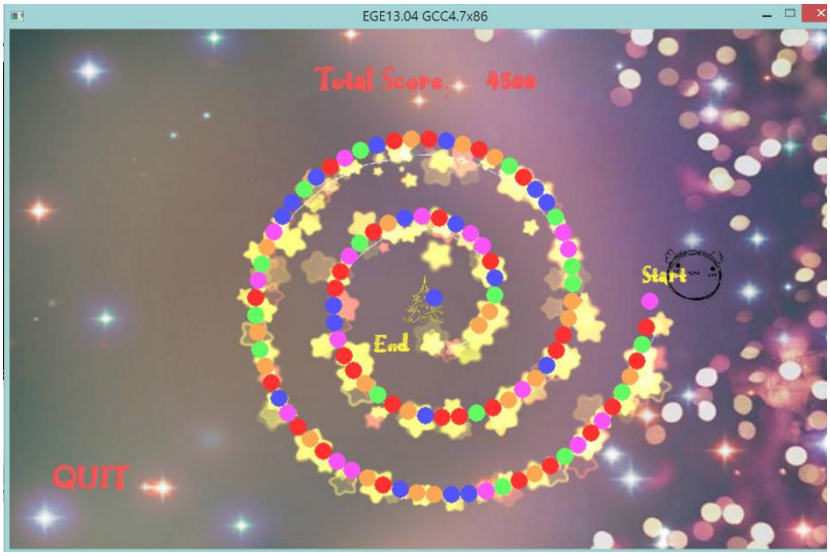
遊戲名稱：Christmas Candy

組員：B03901009 郭笛萱
B03901070 羅啟心
B03901115 吳宇

一、 遊戲介紹：

畫面中間有一隻青蛙，青蛙四周有一軌道，軌道上會有隨機顏色的球不斷從洞口滾出，隨機顏色的糖果會從青蛙口中吐出，玩家要控制青蛙吐糖果的方向，將與糖果相同顏色的三個或以上連續的同色球消除，每當玩家消除 n 顆球，可獲得 n*n*100 的分數，連續消越多，得越多分！

如果玩家來不及消除球，當球滾到出口，玩家就輸了，遊戲也隨之結束。



二、 所使用 third-party 函式庫：EGE

功能：

1. initial 一個 graph 窗口
2. 在 graph 畫出圖片和文字
3. 設置圖片和文字的顏色大小各種參數
4. 動畫中的時間延遲
5. 獲取滑鼠位置和 click 信息

三、 程式操作使用

1. 進入 start 頁面，選擇遊戲 mode 或者遊戲規則說明
2. 開始遊戲：click 軌道上希望小球發出擊中的位置，當射出小球與軌道上小球顏色一致且有 3 個或以上連續的時候會一起消掉，否則發射的小球會插

入軌道。

3. 當任意一小球滾至軌道終點的時候遊戲結束。
4. 顯示最後遊戲得分，click“EXIT”退出遊戲。

四、 各組員分工

1. 郭笛萱：軌道(putFConthetrack, threeormoresame(消球))、發球、class_FrogCandy、printscore、加音樂、debug
2. 羅啟心：軌道((小球螺旋前進)class_candy, cpoint)、“How to play”規則、debug
3. 吳宇：startpage、endpage、中途遊戲 Quit、randcolor、 rh.h、圖片美工、debug

五、　程式說明—小球的軌道設計：

1. struct cpoint

cpoint 是會在軌道上運行的小球。這個 stucture 的 member 主要是控制小球的位置、圖案及顏色、消失與否。

double a

由於軌道是固定的，所以小球的位置由 a(angle)唯一決定。我們的軌道有 800 度，右方為零度，順時針角度增加，因此 a 大約是 0 到 800 的小數。

double R

小球距離中心點(Mid_X,Mid_Y)的距離，由 a 唯一決定。如 picture2。Radius 是 a=0 時的距離，是一個常數。

double x,y

小球的橫坐標和縱座標，由 a 唯一決定。如 picture2。

int round　決定每顆小球滾的圈數，朝過 360 度及滾了一圈，round=1 以此類推

bool draw=false

決定是否畫出小球。

PIMAGE img1　　cpoint()　　color_t color

小球的圖片和顏色，是在 construct 的時候畫出圖片和用 randcolor();決定顏色。

renewxyR()

在 a 被改動之後，根據 a 更新 x y R 的值。

newcpoint()

initialize a x y R draw，並畫出圖片。是在畫面上要從洞口發射新的小球到軌道上時會呼叫的 function。

↓ picture1 (in file rh.h)

→picture2 (in file class_candy.cpp)

```
58    struct cpoint
59    {
60        double a;
61
62
63        double x, y;
64        double R;
65        int round;//記錄轉了幾圈
66
67        PIMAGE img1;
68        color_t color;
69
70        bool draw=false;//whethe
71
72        cpoint();
73
74        void renewxyR();
75        void newcpoint();
76    };
```

```
4   cpoint::cpoint()
5   {
6       color=randcolor();//color of ball
7       img1 = newimage();
8       setfillcolor (color);
9       setcolor(color);
10      fillellipse (10, 10, 10, 10);
11      getimage (img1,0,0,20,20);
12  }
13
14  void cpoint::renewxyR()
15  {
16      R=Radius-0.25*a;
17      x=Mid_X+R*cos(a*PI/180);
18      y=Mid_Y+R*sin(a*PI/180);
19
20  }
21
22  void cpoint::newcpoint()
23  {
24      a=0;
25      x=Mid_X;
26      y=Mid_Y;
27      R=Radius;
28      draw=true;
29      renewxyR();
30      putimage_transparent (NULL, img1, x, y, BLACK);
31  }
```

是控制小球在軌道上行為的 class，因為軌道上的小球在很多情況下必須一起考慮(例如相消、撞前面的小球)，所以要有一個 cpoint 的 array 然後整個 array 一起考慮。

int num

已發射的小球個數，draw==true 和 draw==false 和後來塞入軌道內的小球都算。

cpoint C[MaxNum]

儲存所有在畫面上的小球和即將發射的小球的陣列。

void draw()

把所有 draw==true 且已發射的小球在畫面上畫出來。

double freeangle(int j)

如果 C[j].draw==ture，那麼用餘弦定理計算 C[j]這個小球在畫面上大約最多可以前進多少度而不超越前面的球且與前面球無交集。用於 void plusa()裡面。

ndc[j]　ndc2[j]

在畫面上在 C[j]這個小球(ndc[j]:前面)(ndc2[j]:後面)且離它最近的球的 index。如果 C[j]前面沒有小球，則 ndc[j] == - 1。

void renewndc()　void renewndc2()

找到及更新 ndc[j]和 ndc2[j]。

↓ picture3 (in file class_candy.cpp)
→picture4 (in file class_candy.cpp)

```
79  class candy
80  {
81  public:
82      int num=0;
83      cpoint C[MaxNum];
84      int shoot=5;
85
86      double ctem[MaxNum];//cpoi
87      int ndc[MaxNum];
88      int ndc2[MaxNum];
89      double freeangle(int j);
90      void renewndc();
91      void renewndc2();
92      double needspace[MaxNum];
93
94      ~candy (){closegraph ();}
95      void draw ();
96      void nocollision();
97      void plusa();
98      void update ();
99
100     bool gamecont=true;
101 };
```

```
36  void candy::renewndc()
37  {
38      for(int j=num; j>0; j--) //determine ndc
39      {
40              ndc[j]=-1;//ndc:nearest candy wi
41              for(int t=j-1; t>=0; t--) //find
42              {
43                  if(C[t].draw==true)
44                  {
45                      ndc[j]=t;
46                      break;
47                  }
48              }
49      }
50      ndc[0]=-1;
51  }
52  void candy::renewndc2()
53  {
54       for(int j=0; j<num; j++)
55      {
56              if(C[j].draw==true)
57              {
58                  ndc2[j]=-1;//ndc2:nearest ca
59                  for(int t=j+1; t<=num; t++)
60                  {
61                      if(C[t].draw==true)
62                      {
63                      ndc2[j]=t;
64                      break;
65                      }
66                  }
67              }
68      }
```

```cpp
void candy::draw ()
{
    int i = 0;

    while (i<num)
    {

        if ( C[i].x>0 && C[i].x<2*Mid_X && C[i].y>0 && C[i].y<2*Mid_Y&&C[i].draw==true )
            putimage_transparent (NULL, C[i].img1, (int)C[i].x, (int)C[i].y, BLACK);
        i++;
    }

}

double candy::freeangle(int j)
{
if((C[j].draw==true)&&(ndc[j]!=-1))

    {
        double distancesquare=(C[j].x-C[ndc[j]].x)*(C[j].x-C[ndc[j]].x)+(C[j].y-C[ndc[j]].y)*(C[j].y-C[ndc[j]].y);
        double angle1=acos(((C[j].R)*(C[j].R)+(C[ndc[j]].R)*(C[ndc[j]].R)-distancesquare)/(2*(C[j].R)*(C[ndc[j]].R)));
        double angle2=acos(((C[j].R)*(C[j].R)+(C[ndc[j]].R)*(C[ndc[j]].R)-400)/(2*(C[j].R)*(C[ndc[j]].R)));
        return angle1-angle2;
    }
else return 0;
}
```

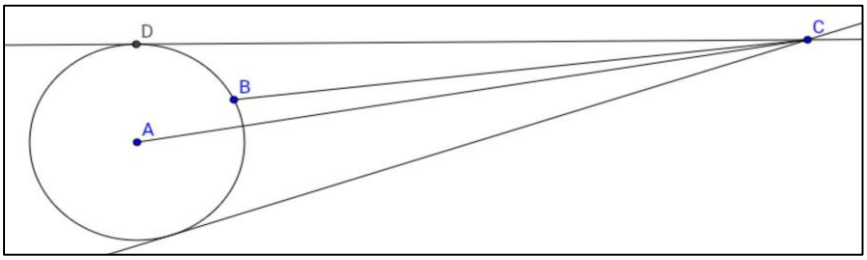
2. nocollision()

↑ picture5 (in file class_candy.cpp)

是一個讓畫面上的球看起來會相撞的 funtion。目的是讓軌道上的球不要有交集。對於每一個畫面上小球 C[j]，計算 C[ndc[j]].a 大約最少要比 C[j].a 多幾度才不會有交集，記這個角度為 needspace[j]。在這裡我們不是算出準確的值，而是做了一個估計。這個估計算出來的 needspace[j]會比真正 need 的 space 大(如 picture 6)，但是由於在我們的軌道中 C[j].R 通常比小球直徑大很多，所以畫面上小球看起來大約是相切。

如果 C[ndc[j]].a>C[j].a+ needspac[j]，那就維持 C[ndc[j]].a。

如果 C[ndc[j]].a<=C[j].a+ needspac[j]，那就把 C[ndc[j]].a 調到 C[j].a+ needspac[j]使兩球看起來相切。



←picture6

↓ picture7

(in file class_candy.cpp)

```cpp
void candy::nocollision()
{
    renewndc();

    for(int j=num; j>0; j--) // 調整順序及使小球無交集
    {
        if(C[j].draw==true)
        {

            if(ndc[j]!=-1)
            {
               needspace[j]=asin(20/C[j].R)*180/PI;
               C[ndc[j]].a=((needspace[j]+C[j].a-C[ndc[j]].a)>0?(needspace[j]+C[j].a):(C[ndc[j]].a));
               C[ndc[j]].renewxyR();
            }
        }
    }
}
```

3. plusa()

這個 function 的主要目的是控制軌道上小球前進(nocollision 也會造成小球前進，但其影響是次要的)，這個 function 表現出來的結果是：軌道上的第一顆球，其速度是角度 a 的函數，但是如果後面沒有球撞上來，前面的球就會停下來等後面的球，直到後面的球撞上來才會推動前面的球前進。

其中 ctem[j]會儲存控制權交給這個 function 時，C[j].a 的值。

```
 99    void candy::plusa()
100    {
101        for(int j=0; j<=num; j++)//store temporary value of angle
102        {ctem[j]=C[j].a;}
103
104        renewndc();
105        renewndc2();
106
107           for(int j=0; j<=num; j++)
108           {
109               if(C[j].draw==true)
110               {
111                     if(ndc2[j]!=-1)
112                     {
113                         if((ctem[j]-ctem[ndc2[j]])>6)
114                         {C[j].a+=0;}
115                         else if(ndc[j]==-1)
116                         {C[j].a+=1/(1+100*C[j].a);}
117                         else
118                         {C[j].a+=min((ctem[ndc[j]]-ctem[j]+0.1*j)*0.05,freeangle(j));}
119                     }
120                     else
121                     {
122                         if(ndc[j]==-1)
123                         {C[j].a+=1;}
124                         else
125                         {C[j].a+=min((ctem[ndc[j]]-ctem[j]+0.1*j)*0.05,freeangle(j));}
126                     }
127                      C[j].renewxyR();
128               }
129           }
130    }
```

↑ picture8 (in file　class_candy.cpp)

```
159   void candy::update()
160   {
161       int madc=-1;//cpoint with minimun angle and draw==true
162       for(int j=num;j>=0;j--)
163       {   if(C[j].draw==true)
164           {
165               madc=j;
166               break;
167           }
168       }
169       if(madc==-1)
170       {   C[num].newcpoint();
171           num++;                }
172       #if CHALLENGEMODE
173       #endif MODESWITCHINGDONE// CHALLENGEMODE
174       shoot=8;
175       #define MODESWITCHINGDONE
176           if(C[madc].a>shoot)
177           {   C[num].newcpoint();
178               num++;                }
179       plusa();
180       nocollision();
181       draw();
182       cleardevice();
183       for(int j=0; j<num; j++)
184       {   if(C[j].a>800&&C[j].draw==true)
185           {
186               gamecont=false;
187               break;
188           }
189       }
190   }
```

↑ picture9 (in file　class_candy.cpp)

4. update()

這個 function 的目的是一直更新小球的位置(呼叫 plusa())，讓小球有跑動的樣子，而且為了有撞擊的感覺，呼叫 plusa()之後要呼叫 nocollision()。還有判斷如果有小球到了軌道終點，遊戲必須結束。shoot 是用來控制發球速度(進而影響小球整體跑動的速度及遊戲難易度)，shoot=5 是用來控制當離發球點最近的球(C[madc])的角度大於 5 度時，會發射一顆新的球。

5. 軌道

軌道是阿基米德螺線，形如 $a - b \times \sin\theta$ 。

6. 小球軌道建構心得:

我們覺得軌道和小球的速度控制比我們原本想的還要難做，尤其是在控制小球的速度。

我們的 plusa()和 nocollision()經過許多次改版，不久之前 plusa()只有三四行，nocollision()的行數卻約是現今的 4 倍。

nocollision 演化史

最原始的 nocollision 是把小球的角度往後調(亦即相切時，是 index 大的減角度)，但是這會有一個非常直接的問題，就是發球太快的時候球會跑到發球口後面。

因此幾天之後改版成第二版的 nocollision(中間有許多次小更動，但是比較瑣碎，以大致的架構來說可以分類為三個版本)。這個版本最大的問題是，把小球的角度往前調的時候，是把所有角度小於某個 index 的小球都往前調，例如 C[j-1]和 C[j]相交，C[j-1]的角度只要加上 theta 就可以不和 C[j]相交，但是版 nocollision 調整的時候會把所有 index 小於 j-1 的小球的角度都加上 ndc，這樣不只沒必要，還會使整體小球行進得太快。

到了非常晚期，為了在 plusa 裡面控制速度，做了 ndc[] 和 ndc2[] 這兩個 array，這兩個 array 非常重要，因為我們主要是在控制螢幕上會出現的物件，相撞也是螢幕上出現的球在相撞，因此 ndc 和 ndc2 的概念非常重要。我覺得我真是太蠢了，怎麼沒有早點發現這點呢。ndc 和 ndc2 的概念在 plusa 裡面出現了一陣子之後，再度被反應球速太快，才想到應該要使用 ndc 和 needspace 修正我的 nocollision。

終於全部改版完之後，我們覺得當初為什麼要叫這個 funtion 為 nocollision 呢，應該叫 collision 或是 nointersection 才對啊，真怪。

needspace

其實很早就有這個概念，如果要解出真正 need 的 space，要解一個以θ為未知數的方程式，裡面有θ的平方項和cosθ ，怒猜他沒有基本函數解(至少我不想找)，所以果斷使用估計。原本使用的是另一種估計方法(picture10)(我們的軌道剛好適用這種估計方法)，算出來的值也會大於真正need的space，並且那個方法保證的最大誤差比現在的needspace可以保證的最大誤差的誤差更小。不過現在算 needspace 的 code 更為簡潔，所以就使用現在的估計方法。

plusa()

原始的 plusa()長得非常簡單，不過後來發現我們想要改成前面的球就要停下來等後面的球，直到後面的球撞上來才會推動前面的球前進。所以原本的 plusa 不敷使用，才趕緊加了 ctem,ndc,ndc2,freeangle 等，並且分好多 case 控制各種處境的小球的速度。plusa 後來意外地成為一個有點多東西的 fuction。

控制球速

控制球速是我寫的東西之中最難的，因為要兼顧非常多東西，例如畫面和諧(不能一邊太擠，一邊太空)、不能太快或太慢、後面的球要追上前面的球並且球速不能到後面變很快(這兩者一度互相牴觸，難以解決) 、RELAXMODE 和 CHALLENGEMODE 的難度分別(在 plusa 設計完善之前，怎麼改都改不出差別) 、以及最後要求的前面的球就要停下來等後面的球，直到後面的球撞上來推動前面的球前進。

最後的 plusa 是分成許多 case 處理各種處境的小球的速度。

```
double angle2=acos(((C[j].R)*(C[j].R)+(C[ndc[j]].R)*(C[ndc[j]].R)-400)/(2*(C[j].R)*(C[ndc[j]].R)));
```

↓ picture11 : second edition of nocollision()　　↑ picture10

```
85  void candy::nocollision()
86  {
87      for(int j=0; j<num; j++) //first adjestment 調整使順序正確
88      {
89          if(C[j].draw==true)
90          {
91              for(int t=j+1; t<=num; t++)
92              {
93                  if((C[t].draw==true)&&(C[t].a>=C[j].a))
94                  {
95                      {
96                          C[t].a=C[j].a-t/100;
97                      }
98                  }
99              }
100         }
101         for(int j=0; j<=num; j++) //third adjustment  : renewxyRn();
102         {
103             C[j].renewxyRn();
104         }
105     }
106     for(int j=num; j>0; j--) //second adjustment 調整使小球無交集
107     {
108         if(C[j].draw==true)
109         {
110             ndc[j]=-1;//ndc:nearest candy with draw==true
111             for(int t=j-1; t>=0; t--) //find the first candy (index>j) with draw==true
112             {
113                 if(C[t].draw==true)
114                 {
115                     ndc[j]=t;
116                     break;
117                 }
```

```
118             }
119             if(ndc[j]!=-1)
120             {
121                 double distancesquare=(C[j].x-C[ndc[j]].x)*(C[j].x-C[ndc[j]].x)+(C[j].y-C[ndc[j]].y)*(C[j].y-C[ndc[j]].y);
122                 if(distancesquare<400)//if the intersection of ndc and j is not empty
123                 {
124                     double angle1=acos(((C[j].R)*(C[j].R)+(C[ndc[j]].R)*(C[ndc[j]].R)-distancesquare)/(2*(C[j].R)*(C[ndc[j]].R)));
125                     double angle2=acos(((C[j].R)*(C[j].R)+(C[ndc[j]].R)*(C[ndc[j]].R)-400)/(2*(C[j].R)*(C[ndc[j]].R)));
126
127
128                     for(int u=ndc[j]; u>=0; u--)
129                     {
130
131                         C[u].a+=(angle2-angle1)*180/PI;//not the precise solution but with acceptable error
132                     }
133                 }
134             }
```

六、　程式說明—小球的發球設計：

1. Class Frogcandy & Strucr fpoint

此部分為控制中間青蛙發球的程式碼。首先須先建立一個叫做 FrogCandy 的 class，此 class 裡面有一個 member 叫做 fpoint 的 structure，此 structure 裡面會以矩陣的方式，記錄每顆欲發出的球的各種資訊，包含其發射角度、運動位置、顏色及是否需畫出來……等等。

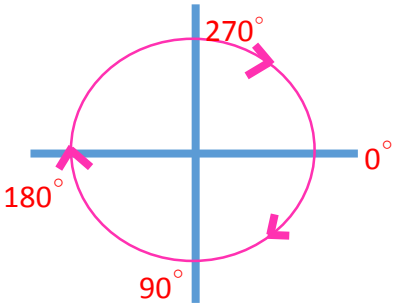
當玩家按下滑鼠後，程式會呼叫其 member function :mouse()，此函數會先取得滑鼠座標位置，再透過反函數和象限換算，得到和發射原點間的角度(如圖一)，需注意的是，為了配合 class candy 中小球在軌道上順時針的運動方式，此程式的角度為從起點開始順時鐘針遞增，而非一般習慣的逆時針。取得此角度後，再透過極座標的方式，讓每次發出去的球位置變為

FC[k].x += cos (angle) * 10;

FC[k].y += sin (angle) * 10;

即可控制發球方向。

為了讓發出球後，可以先顯示下一顆球的顏色，所以我們寫了一個 drawnext 的 funtion，讓玩家可以有心理準備下一顆球為何。




```
29  struct fpoint
30  {
31      double x, y;
32      double ang;
33      PIMAGE img2;
34      color_t color;
35      fpoint();
36      bool draw2=true;
37
38  };
39  class FrogCandy
40  {
41
42      int  num;
43  public:
44      fpoint FC[MaxNum];
45      FrogCandy() ;
46      ~FrogCandy ()
47      {
48          closegraph ();
49      }
50      void initial (int x0, int y0);
51      void draw (int);
52      void drawnext(int);
53      void update (int,double);
54      double mouse ();
55
56  };
```

七、　程式說明—判斷將球塞入軌道與連續三個消除

1. Func_PutFConthetrack

此部分為控制發出去的球如果沒有遇到三個連續相同顏色的情形的話，就要塞入軌道內。關於這部分，其實我們思考了很久，因為發出去的球跟軌道上的球是屬於不同 class，一個是 class frogcandy 另一個是 class candy，兩者屬於不同編號系統，如果只是單純讓發出去的球停留在軌道上，它將不會照著軌道前進，會打亂整個程式的結構，產生很多問題。所以後來我們想到一個方法，就是先判斷發出去的球是否抵達了軌道，如果抵達軌道，就讓發出去的球消失，並在軌道上產生一個和發出去的球一樣位置，一樣顏色，但是是屬於 class candy 的小球，如

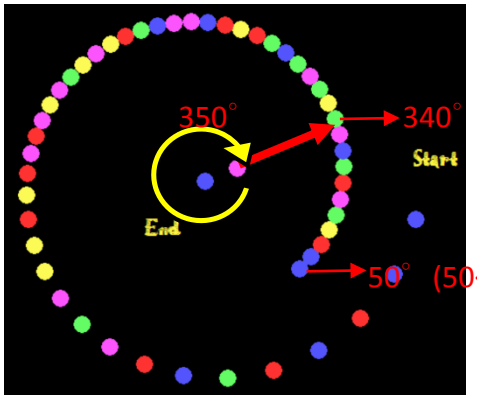
此一來，雖然程式寫起來兩顆是不同的球，可是玩家會誤以為是發出去的小球被塞進去了。

為了維持程式的結構，產生出來小球的編號勢必要介於兩旁的小球編號，否則之後小球運行的判斷會有誤，所以我們必須先判斷究竟產生的小球的編號是什麼？

我們首先以角度來判斷小球應該要塞入哪兩顆小球之間，for 迴圈從第 0 顆小球開始判斷起，當發出去的球的角度大於小球軌道上第 i 顆球時，即可知產生出來的小球介於第 i-1 顆和第 i 顆間。於是我們將第 i 顆以後的球的編碼都加一，而產生生來的球的編碼為 i。

當我們寫到這裡的時候，又碰到問題了，因為小球軌道上的球的角度為 0~800 度，而發出去的球的角度為 0~360 度，發出去的球的角度不會因為轉的圈數而有所變化，所以會有判斷上的問題，當轉到第二圈後，球不管往哪裡射，FCangle 都不會大於 c.C[i].a。為了解決此問題，我們在 cpoint 裡面多紀錄了一個 round 也就是每個軌道上小球轉的圈數的資料，當軌道上的小球轉超過一圈時，round=1，兩圈 round=2…。所以角度判斷是以 c.C[i].a-round*360 和 FCangle(發出去的球的角度)來判斷。

但是，我們發現，這樣雖然在只轉了一圈的情況下都是對的，小球都可以正常塞入軌道，但是在小球轉了超過一圈後，就會出現問題，以下圖為例，當 FCangle 為 350 度，其應該塞入 340 度小球的旁邊，但是因為前面的球已經進入第二圈，角度從 0 開始，所以一定小於 350 度，造成此顆球會塞進最前端，產生錯誤。



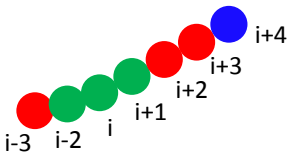
為了解決這個有關角度與象限造成的問題，我們想到應該先判斷發出去的球和軌道上的球是否在同一象限，如果不在同一個象限，那就跳過，直到遇到同一象限的小球後，才開始進入 for 迴圈，判斷角度。如此一來，我們成功地把球塞進軌道內了。

2. Func_ThreeOrMoreSame:

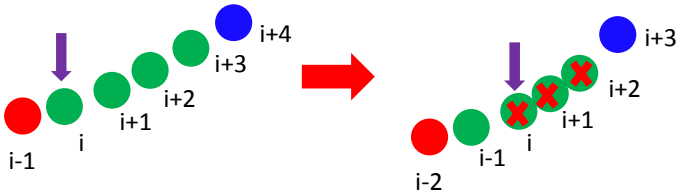
此部分為判斷是否有三個或以上相同顏色的球相鄰，如果相鄰且個數大於等於三個，則讓它們全部消失。首先，以下圖為例，此程式會以第 i 顆球為基準，判斷其右邊下一顆 draw==true〔也就是尚未被消除的球〕是什麼顏色，此例子中第 i+1 顆 draw==true&&顏色相同，則可知有連續兩顆一樣顏色，並且繼續判斷下一顆球，如果不相同，則開始判斷其左邊有無連續顏色的球，方法同右邊。

在判斷的過程中，因為不確定是否連續的球數會超過三顆，所以必須先將有可能須被消除的球的資訊存入矩陣 samecolorballright [ ] & samecolorballleft [ ]中，此例子中存入 i-2,i,i+1。等左右兩邊都判斷完後，如果球數大於三顆，則讓此矩陣內的球都 draw=false，如果小於三顆，就維持 draw==true。

不過，在寫此段程式的時候，碰到了一個困難就是，如果只寫到這樣，當我們消掉圖中第 i 顆球時，其實它旁邊 i-3,i+2,i+3 的球也會同時判斷是否有連續三顆，所以變成一連串 i-3~i+3 的球一起消失，在畫面上看起來很怪，玩家應該會期待先消失綠色的三顆球，過一小段時間後，再消失紅色的三顆球，所以我們在判斷小球是否消失的判斷式中，加一個條件，就是相鄰連續顏色的小球必須在一定的距離內才會被消除，否則需等到它們跑到互相很靠近的時候才會消除，這樣問題就解決了。



此段程式看似很長，但是其實是因為程式需分別判斷左右兩邊是否有連續顏色的球造成的，左右邊其實是相似的程式碼。我們也有想過只從每組球的第一顆開始判斷，如下圖，從 i 開始，有連續四個綠球，這樣就可以只判斷一邊，程式碼就不用寫的那麼複雜，但是這樣當有連續四顆球時，程式有時就會出錯，因為在判斷每組第一顆時，小球間間隔太大了，尚無法消除，但是當判斷到該組第二顆球時，小球間隔瞬間滿足條件，所以就只有第 2,3,4 顆球被消除，而第一顆就被忽略，所以才有左右兩邊的設計。



八、 程式說明—其他

1. main function

Main 裡面放的主要是各種 initialization 和小球運動 function。在改 main 的最難的部分在於排好每一個 function 的位置。以最簡單的 print 背景圖為例，剛開始將 putimage 放在所有 function 的最前面，但是一閃就被 cleardevice，然後放在 for 迴圈裡面的 cleardevice 前面，run 的時候剛 print 出來就被 clean，最後加在了 cleardevice 後面，並且因為 main 裡面出現了很多次清屏，因此需要在每一次 clear 之後都加上 putimage。除此之外的很多與小球運動和 print score 有關的重要的 function 都做了很多次的順序調整才確定先後順序，有些 function 甚至需要在 main 的不同位置重複使用，例如 mousemsg 部分。

九、總而言之 main 因為關聯了所有 function 的進行所以最為複雜(例如有 6 層大括弧)，每一次 debug 都需要很大的耐心和很強的邏輯。

2. color function:

對於小球的顏色，主要用到的 EGE 函式是 RGB，setcolor 和 setfillcolor。函式 RGB 可以通過紅綠藍三個十六進制值調和顏色並回傳。因此在這三個參數的位置上加入 random 函式就可以使每一個小球的顏色有多種可能性。

由於在程式 running 的過程中會不斷地 cleardevice，所以也要不斷地重新畫出小球。為了讓小球的顏色在每次被畫出時都能保持相同，我們將控制顏色的參數寫進了小球的 class 裡面。

zooma 遊戲中，只有在發射的小球顏色與軌道上的小球顏色相同時，軌道上的球才能消掉。所以如果在 random 小球顏色的時候不加以控制，會因為球的顏色種類實在太多而無法碰到相同顏色的軌道球。解決這個問題的時候我們把在 RGB 裡面直接 random 改為了 random 0 到 5 之間的一個數字，每個數字對應一個包含 6 個元素的 array 裡面的其中一個元素。再用 switch 將對應顏色值回傳。通過這個方法，我們將小球控制在 6 種顏色的範圍內。

但在多次測試遊戲程式之後發現每次玩的時候，小球顏色的排列都是一樣的。但是如果在 function 裡面加 srand(time(NULL))之後，所有小球又都變成了同一種顏色。這個問題始終未能得到解決，在將 srand 在 function 裡面刪掉又加上很多次之後我們才發現，將 srand 放在 main 裡面才能正常發揮作用。至此，小球 color 的問題解決完畢。

3. 起始結束頁面：

start 和 end page 主要起到的作用是開始之前選擇 mode，和結束之後顯示 total score，不會直接閃退。

寫這兩個部分最主要解決的幾個部分是對滑鼠所在位置和點擊位置判斷並作出回應，還有與遊戲主體的銜接。

關於滑鼠位置，我們在 project 當中都是用 msg = getmouse()來 get 滑鼠所在的 x，y 坐標，然後用 if 加判斷式來判斷字體顏色是否應該從紅色變成綠色，以及是否點擊進入遊戲主體部分。但剛寫出來的時候雖然把滑鼠移到文字上的時候顏色會變但是一直在閃爍而且 click 的時候很不靈敏，常常點不進去，但 debug 的時候 x，y 軸的判斷式沒有出問題。後來發現顏色一直閃爍不穩定是因為多加了一個 cleardevice，導致字體變綠之後會被清屏然後又畫成紅色。而 click 不靈敏的問題被發現是因為我們在每次 while (mousemsg()){msg = getmouse();}之前都加了一個初始化 mouse_msg msg={0};而 while (mousemsg())使得只有當滑鼠在移動時才會 get 新的 mouse 位置，如果滑鼠不動，xy 軸就保持當初被初始化的 0. 所以最後的結果是，只有一邊滑鼠移動一邊 click 才能進入到遊戲。所以將 while 之前的初始化 msg 刪去，將唯一的 initial 放在 for (; is_run(); delay_fps(60))的外面。

與遊戲主體的銜接主要與 endpage()在 main 裡面的擺放位置和 class_candy 最後判斷 gameover 的指令有關。因為在我們之前寫的版本中，當軌道球滾到中心發球處附近之後就會直接 gameover 並 closegragh。但是如果只是簡單地把 class_candy 最後的 closegragh 改成 break，遊戲結束之後程序不會跑到 main 最後的 endpage 的 function，而是停在 for 迴圈的中間繼續跑剩下的程式，然後出現很詭異的畫面。所以我們又在 class_candy 裡面引入一個 bool 參數 gamecont 配合 main 裡面的 if(gamecont)來判斷是否 continue 遊戲。如果遊戲結束，則 break for 迴圈進入 endpage。

除此之外，在寫 endpage 過程中曾把 totalscore 設成了 header 檔裡面的一個 global 的參數，本想藉此在 endpage 所在 file 裡面 access 到 printtotalscore file 裡面被修改到的總分數。但是發現不論 printtotalscore 的 file 裡面 totalscore 被改成了多少，endpage 能 access 到的 totalscore 都是 0. 後來才知道 printtotalscore 函數無法改動到 totalscore 的值，只能通過變數值傳入 function 的方法，在另外一個 file 裡面 show 出被最後修改過的 totalscore 值。

4. Func_printscore

此部分為印出 1.遊戲畫面上方的 totalscore　2.連續小球消除後產生的分數的程式碼。比較特別的是，我們希望消出小球時，分數出現後大概只要維持二秒就要消失，而不是一直存在，或是馬上就被 cleardevice 清除而不見，所以我們設計了一個 numberoftimes 的參數，此參數會記錄畫面總共更新了幾次，當更新了 20 次，也就是大概 2 秒後，或是玩家又消掉了一組球後，原先的分數就不再印出。並且為了讓分數出現時能在消掉的球的旁邊，和有點小小上升的動畫，我們寫了：

outtextxy(c.C[numball].x-20, c.C[numball].y-numberoftimes, printscore);

Y 座標會隨著時間而改變，就可以使分數往上跑。