中華民國第 57 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

第三名

052504

球體手機鎖

學校名稱：臺北市立建國高級中學

作者：	指導老師：
高二 陳冠廷 高二 黃煒勛 高二 鄭仲語	許雅淳

關鍵詞：正十二面體、Android App、手機解鎖

摘要

本研究試著在 Android Studio 的開發環境下用 Java 語言寫出手機鎖，將現行之平面手機鎖擴張至三維空間，運用立體的圖形增加可行解組合數及複雜度，進而在不困擾用戶的情況下提升螢幕解鎖的安全性。

壹、研究動機

在現今的社會中，資訊安全是一個被各界高度關注的議題，個人的行動裝置也不例外。在校園中也常常發生造成遺憾的事件，無心的玩笑有可能衍生出嚴重的誤解，而這些皆肇因於手機螢幕解鎖的安全機制不甚完善，讓我們意識到提升解鎖安全性的重要性。現行之螢幕解鎖圖形皆為平面，推測將其擴展為立體可增加其組合方法數及破解難易度，使圖形在可記憶的範疇內達到安全性的提升。

貳、研究目的

我們的目標是寫出適用於 Android 系統的手機鎖。前置作業中必須先對現行手機鎖進行組合數的分析，以確保我們程式的組合數有所提升，接著建立數學模型及立體圖像，待應用程式主體完成可實用後對其進行優化。

參、研究設備及器材

一、Mac BookPro (2015)

規格： CPU 2.7GHz Intel Core i5

記憶體 8 GB 1867 MHz DDR3

儲存空間 SSD 256GB

作業系統 OS X El Capitan 10.11.2

二、Zenfone 2 (ASUS_Z00AD)

規格： CPU Intel® Atom™ Quad Core Z3580 (2.3GHz), PowerVR G6430

記憶體 4GB LPDDR3

儲存空間 32GB

作業系統 Android™ 5.0

三、ASUS M700-PU451LD

規格： CPU i7-4510U

8GB DDR3L 1600

儲存空間 500GB 7200rpm

作業系統 Windows 7 Pro 64 bit

四、ZenFone 3 Max (ZC552KL)

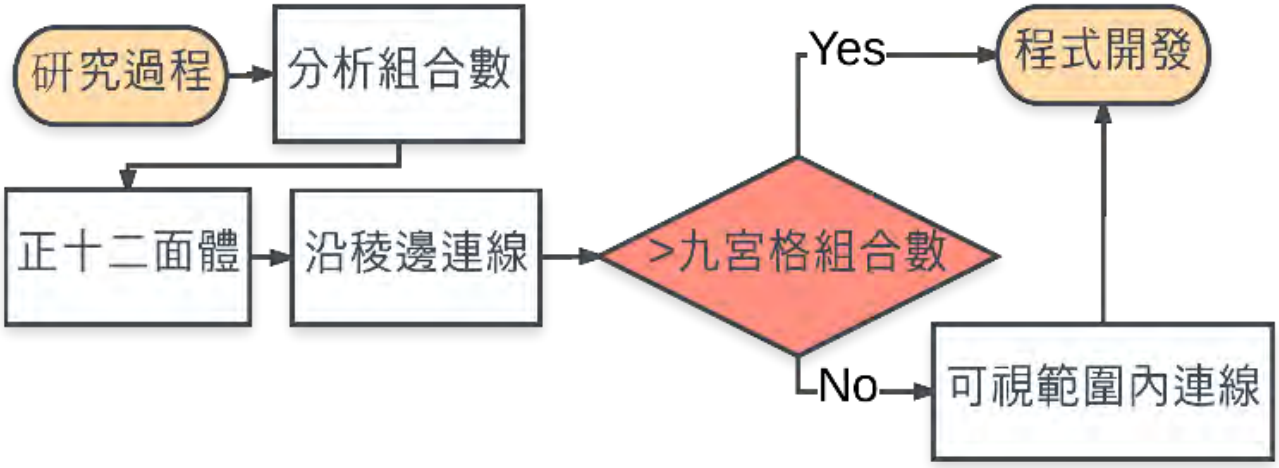
規格： CPU: 64 位元高通® S430 八核心處理器 @1.4GHz

記憶體 2GB LPDDR3

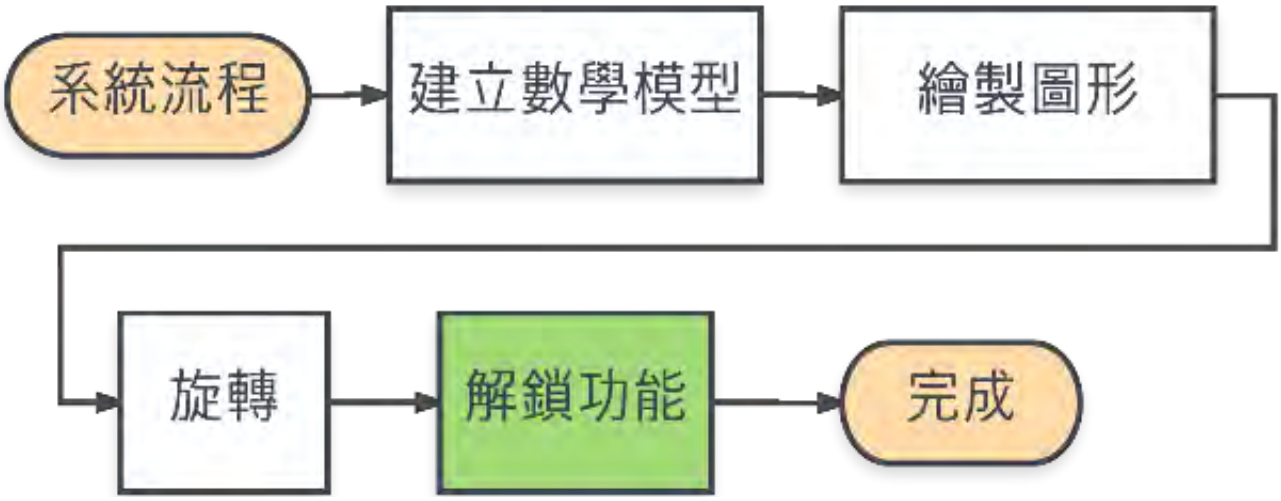
儲存空間 32GB

作業系統 Android™ 6.0

肆、研究過程或方法



圖一、研究過程的流程圖



圖二、程式執行的流程圖

一、分析比較組合數

(一) PIN 碼

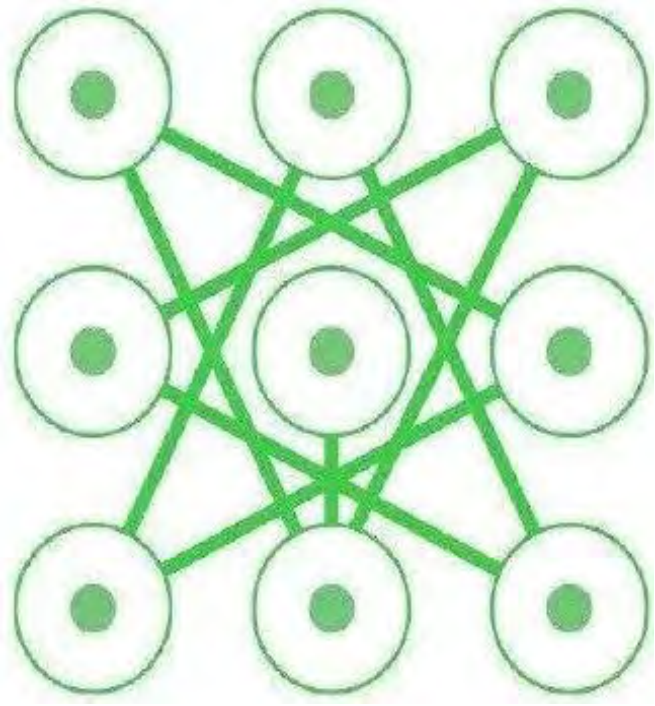
現行 PIN 碼所使用之字元為數字之 0~9，可輸入四碼，因此組合數有 10000 組， 惟設定 PIN 碼時經常使用生日、電話號碼，且數字便於記憶，容易在輸入時被旁觀者記住密碼。

(二) 文字密碼

相較於 PIN 碼，文字密碼則複雜許多，一般所使用的 ASCII 碼中共有 95 種可顯示的字元供使用，當輸入 n 個字元時即產生95^n組組合（在此不討論特殊符號、其他語言文字）。其缺點和 PIN 碼相似，但複雜度已有所提升。

(三) 九宮格圖形

現行智慧型裝置所安裝的圖形鎖為 3x3 之九宮格圖形，我們參考 54 屆中小學科展「按圖鎖驥」[1]所討論之組合數，以電腦運算可行解的數量（連結點數大於三個）得出結果為 389112 組，與「按圖鎖驥」中所求得之組合數吻合。



圖三、現行九宮格螢幕解鎖之示意圖

```
1: function COUNT(Point,Depth)
2:     for every other point B do
3:         if B is connected or B is not reachable then continue
4:         else
5:             COUNT(B,Depth + 1)
6:         end if
7:     end for
8:     if Depth ≥ 4 then
9:         Possibility + 1
10:    end if
11: end function
```

圖四、組合數求解方法

(四) 生物特徵

生物特徵近年來在市場上愈來愈普遍，提供用戶快速且精準的解鎖方案，但多數裝置在生物特徵解鎖外皆有圖形或密碼鎖，故在此不對生物特徵解鎖的安全性做討論。

二、建立數學模型

(一) 正十二面體基本性質

1. 各面皆為正五邊形

2. 共 12 個面、30 條邊、20 個頂點

3. 二面角：$\cos^{-1}-\frac{1}{\sqrt{5}}$，約等於 116.57 度

4. 各點可被描述為以下四組，其中 φ 是黃金分割數，$\varphi=\frac{1+\sqrt{5}}{2}$

$(0,\pm 1/\varphi,\pm\varphi)$、$(\pm 1/\varphi,\pm\varphi,0)$、$(\pm\varphi,0,\pm 1/\varphi)$、$(\pm 1,\pm 1,\pm 1)$

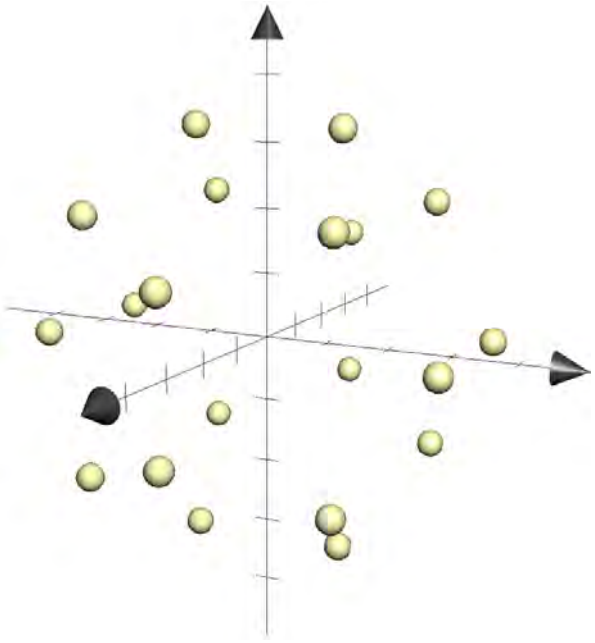
(二) grapher 繪製過程

1. 我們將欲繪製之正十二面體各點距原點距離定為$\sqrt{3}$，各點座標（笛卡兒）詳見表一：

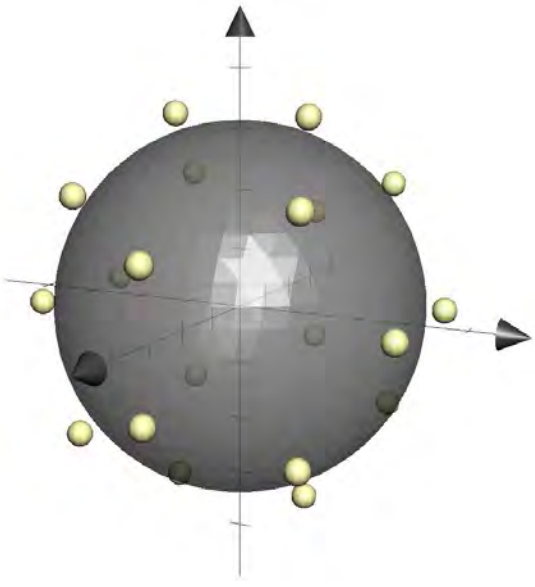
表一、正十二面體各點座標

x	y	z
0	0.618	1.618
0	-0.618	1.618
0	0.618	-1.618
0	-0.618	-1.618
0.618	1.618	0
0.618	-1.618	0
-0.618	1.618	0
-0.618	-1.618	0
1.618	0	0.618
1.618	0	-0.618
-1.618	0	0.618
-1.618	0	-0.618
1	1	1
-1	-1	-1
1	1	-1
-1	1	1
1	-1	1
-1	-1	1
1	-1	-1
-1	1	-1

2. 以半徑 2.5 之圓球輔助觀察，在 Grapher 軟體內建立三維空間中立體、可旋轉之模型以觀察正十二面體旋轉狀況。

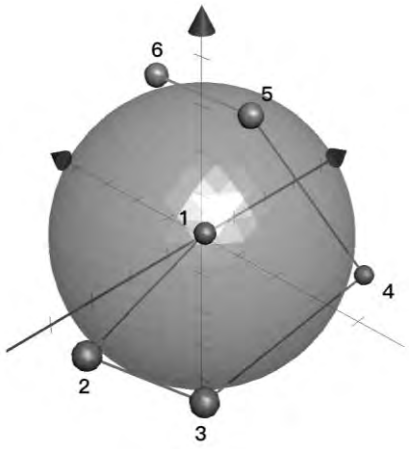


圖五、各點於三維空間之分佈



圖六、加入圓球方便觀察各點之前後關係

3. 建立點與點之間的連線以模擬連線順序，當點依照 1~6 之順序通過視野中央時，即達成解鎖的目標。



圖七、各點連線示意圖

三、畫出立體圖形

(一) 設定座標、繪製圖形

接著將一點固定在 (0,0,r) 的正十二面體的座標輸入，其中 r 為外接球的半徑，加以縮放後將形成邊的 30 對點座標，在執行階段保存在陣列內，導入 byte buffer 並彩現。

(二) 討論連線方式

1. 沿稜邊

判定條件：

Algorithm 1 Connect by Edge

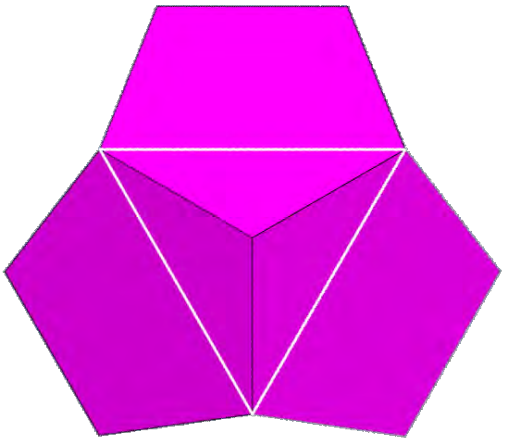
```
for all 0 ≤ i ≤ 19 do
    for all 0 ≤ j ≤ 19 do
        if i ≠ j then
            Matrix_ij = Distance between Vertix_i and Vertix_j ≤ 4/(√3(1+√5)) r
        end if
    end for
end for
```

圖八、連接稜邊頂點的組合數

組合數共 12538 組，較九宮格少，無法達到增加組合數之目的，故需將稜邊連線以外的頂點納入討論。

2. 可視半球內

圖九為正十二面體頂點位於視野中央時，可見的面。



圖九

判定條件：

Algorithm 2 Connect in Vision

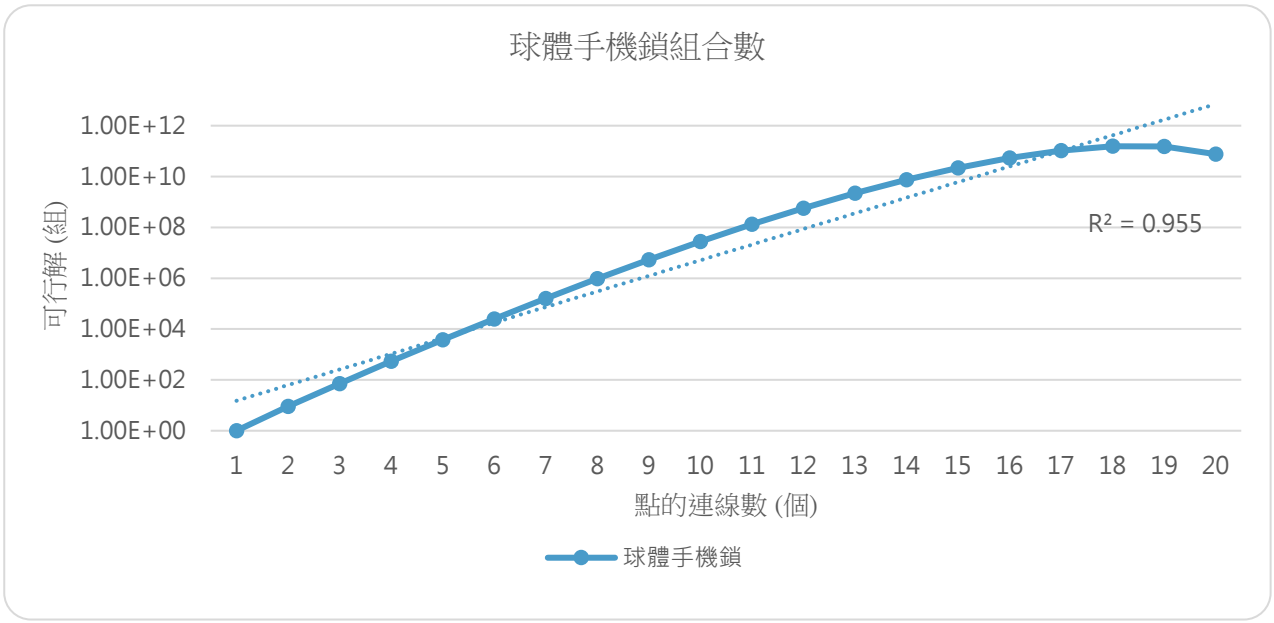
```
for all 0 ≤ i ≤ 19 do
    for all 0 ≤ j ≤ 19 do
        if i ≠ j then
            Matrix_ij = Distance between Vertix_i and Vertix_j ≤ √2r
        end if
    end for
end for
```

圖十、連接可視球面頂點的組合數

共 574812244887 組，組合數大幅提升，可滿足增加組合數之目的，本研究採用此方式進行討論。

(三) 組合數討論

為使使用者操作便利，本研究將起始點固定，討論各點數連接之組合數，並以 Excel 繪出。



圖十一、點的連線數與組合數之關係折線圖

橫軸為點的連線數，縱軸為經對數處理後的組合數（可行解），以折線圖繪出，我們發現組合數在連接 1 個點到 18 個之間，與組合數呈正相關，而在連接 19 個點、20 個點時，組合數下降，我們推測在需要連接第 19 個點時，不符合連線的點比率上升，超越可連接點的比率，許多原 18 個的連線無法往第 19 點連接，而導致組合數下降，這情況在連接 20 點時更為明顯，故組合數下降更多。

(四) 旋轉

利用可取得的 dx,dy(手滑軌跡的位移差)，算出手機鎖該旋轉的角度。根據使用者的操作，dx 將使手機鎖繞 y 軸旋轉，dy 則繞 x 軸旋轉依據矩陣運算。

繞 x 軸的旋轉矩陣：

$$Rx = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos dy & -\sin dy \\ 0 & \sin dy & \cos dy \end{bmatrix} \qquad (4.1)$$

及繞 y 軸的旋轉矩陣：

$$Ry = \begin{bmatrix} \cos dx & 0 & \sin dx \\ 0 & 1 & 0 \\ -\sin dx & 0 & \cos dx \end{bmatrix} \qquad (4.2)$$

將4.1及4.2式中之Rx與Ry矩陣相乘，獲得旋轉矩陣：

$$R = Rx * Ry$$

$$= \begin{bmatrix} \cos dx & 0 & \sin dx \\ \sin dy \sin dx & \cos dy & -\sin dy \cos dx \\ -\cos dy \sin dx & \sin dy & \cos dy \cos dx \end{bmatrix} \quad (4.3)$$

實作方法：

```
1: function ROTATE(rotateX, rotateY)
2:     for next three float x, y, z in array do
3:         newX = x cos(rotateX) + y sin(rotateY) sin(rotateX) − z sin(rotateX)
4:         newY = y cos(rotateY) + z sin(rotateY)
5:         newZ = x sin(rotateX) − y sin(rotateY) cos(rotateX) + z cos(rotateX) cos(rotateY)
6:         (x, y, z) = (newX, newY, newZ)
7:     end for
8: end function
```

圖十二、計算旋轉後座標的虛擬碼

(五) 追蹤點

實作方法：

```
for all Point P do
    if Distance between P and (0, 0, r) ≤ DetectingZoneRadius and Matrix_PQ is True and
Q ≠ P then
        connect to P
        Previous Point P = Q
    end if
end for
```

圖十三、連接各點

(六) 資料儲存

將程式執行過程時的數值，雜湊後儲存至Android 內部空間

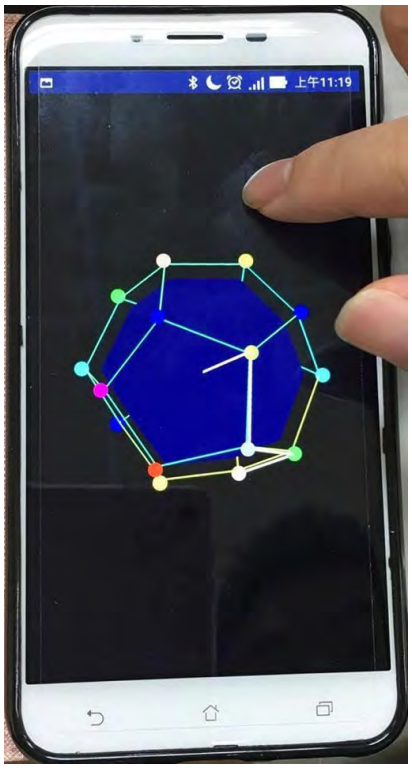
```
1: function DOUBLEHASH(ArrayHashCode)
2:     ArrayHashCode = ArrayHashCode xor RandomSeed
3:     return md5(salt(ArrayHashCode))
4: end function
```

圖十四、雜湊的虛擬碼

伍、研究結果

一、圖形及旋轉

已成功編寫出可在 Android 裝置上運行的應用程式，置於螢幕中央的正十二面體將沿手指滑動方向持續轉動，並在中央有一較小深色正十二面體方便分辨前後。各點以彩色表示方便辨識，以紀錄的點以白色連線標示，並設有旋轉速度上限。



圖十五、應用程式於 ZenFone 3 Max 上之運行狀況（Android 6.0 API 23）

圖十六、應用程式於模擬器（Nexus 5 API 22）之運行狀況

二、點的追蹤

可顯示進入視野中央特定範圍之頂點，並紀錄其與下一頂點連線順序。

表二、隨機滑動時於 Android Studio 的 event log 中所記錄到的點

03-10 11:06:22.418 3264-3264/com.example.a3d W/Rotate:Point detect: rotate lock: point 1 detected, total point 0
03-10 11:06:22.431 3264-3264/com.example.a3d W/Rotate:Point detect: rotate lock: point 1 detected, total point 1
03-10 11:06:32.530 3264-3264/com.example.a3d W/Rotate:Point detect: rotate lock: point 3 detected, total point 2
03-10 11:06:40.279 3264-3264/com.example.a3d W/Rotate:Point detect: rotate lock: point 15 detected, total point 3

三、導入解鎖功能

目前開發當中，須在應用程式運作時使實體按鍵無法操作，惟 Android 4.0 以上並無直接取消 home 鍵之系統功能，計畫將 home 鍵設為 custom launcher 實作

陸、討論

一、為何選用正十二面體？

起初，為了便於程式編寫，我們選擇正多面體作為我們立體手機鎖的解鎖工具，而在所有的正多面體，正十二面體是 5 種中（正四面體、正六面體、正八面體、正十二面體和正二十面體）中頂點數最多者。當我們使用正十二面體時，可視的半球面僅有 10 點，我們認為其與平面手機鎖的九點差異不多，將利於使用者的操作，且經過計算後，發現其擁有更多的組合數，因此最後我們選用正十二面體來做研究。

二、選擇沿著手指軌跡旋轉或沿滑動方向持續轉動的差別？

若手機鎖沿著手指軌跡旋轉，在使用上會有邊界的限制，意即無法在單一方向做無止盡地轉動，這將損失不少組合數；而若沿滑動方向持續轉動，就可往單一方向持續滾動，對於使用者將有更多連線方式可供選擇。

三、是否可以有效阻止不肖人士？

本手機鎖擁有比平面還要更多的組合數，更能防範不肖人士的暴力破解，且平面在解鎖的過程或完成，一旦被記錄下，就大幅增加被破解的機率，而製作成立體的好處，在於外人即使看到片段的過程，或最終結束畫面，皆仍無法輕易回推前面的連線過程，甚至看完使用者完整的連線過程，在立體的連線上亦會使外人更難以記憶。

四、在不同廠牌裝置上是否可以順利運行？有何差異？原因為何？

目前應用程式所使用的 SDK 版本為 23，對應的 Android 版本為 6.0（Marshmallow），此版本於 2015 年 10 月發行，截至 2016 年 5 月市佔率僅 7.5%，遠不及前代發布的版本，因此在低於 6.0 環境下運行之裝置可能出現圖形縮放、顏色上的異常。此外非原生 Android 系統亦會造成影響，我們在實測時同版本、不同廠牌所呈現之圖形有些許差異。

五、為何不使用整個球體，而僅使用可視範圍作為解鎖的條件?

由程式得到之結果已知可視範圍內頂點連線組合數遠超九宮格，可以達成研究目標，若使用整個球體會使連線方法過於複雜，令使用者難以記憶及操作。

六、擴張立體圖形討論範圍（八面、二十面等）

若把頂點數向下減少，像是正八面體和正二十面體，雖立體卻無法有效阻止不肖人士，運作上比起原平面九點手機鎖更為複雜，卻沒有更好的效果在組合數上，我們認為此舉將違背最初提升手機鎖“安全”的目的。

若把頂點數向上增加，像是截角二十面體（巴克球），因正十二面體已是所有正多面體中頂點數最多者，故我們只能尋求半正多面體或其他更為複雜的立體圖形，這在撰寫過程、連線方式與設計及美觀上，會違背最初手機鎖”實用“的目的。

柒、結論

一、將點與點連線組合數擴張至三維、更多頂點之圖形討論可得到較平面九宮格多的可行解數。

二、使用頂點數多，而圖形不致過於複雜者將可在使用者容易記憶的前提下，防止有心人士以窺視手指動作、軌跡的方法取得行動裝置內部資訊。

三、在起始點固定的情況下，再連接 6 點，則組合數與平面 9 點手機鎖相去不遠，而再連接 7 點以上時，則大幅超越在平面 9 點手機鎖同連接點數之組合數。

四、未來在圖形上做些改變，與半正多面體的手機鎖進行比較，對組合數及其相關數據分析，以探討手機鎖的安全性，並加以改進。

捌、參考資料及其他

[1]第 54 屆中小學科展 生活與應用學科 按圖鎖驥 黃昱睿;曾子洋;謝昊芫 2014 年

[2] OpenGL (https://www.opengl.org/)

[3]Java™ PlatformStandard Ed. 8(http://docs.oracle.com/javase/8/docs/api/)

[4]OpenGL ES | Android Developers (https://developer.android.com/guide/topics/graphics/opengl.html)

【評語】052504

本作品運用立體的圖形增加可行解組合數以及複雜度，來實現 Android 手機的螢幕鎖。

主題清楚，研究成果完整，且有資料的分析， 研究成果可直接運用在 Android 手機上。

或許可針對提供使用者友善使用環境考量得更仔細，讓使用者更容易上手。

作品海報

Introduction

本研究試著在 Android Studio 的開發環境下用 Java 語言寫出手機鎖，將現行之平面手機鎖擴張至三維空間，以正十二面體做為模型，運用其立體的特點，增加手機鎖的安全性及實用性。我們比較各類常見手機鎖，並針對正十二面體進行分析，探討其組合數在不同連線條件時的變化。發現不僅在組合上超越市面上常用的平面手機鎖，其立體的特性亦使外人無法輕易破解，且在個人化及美觀方面，因程式碼的完整建立，可以改變至其他立體圖形，進而增加專屬使用者的手機鎖選擇。此創意以 App 作為呈現，不像指紋解鎖需硬體支援，目前開發從 Android 5.0 以上皆能支援，給予廣大的手機用者另一個解決資安問題的選擇。

壹、研究動機

在現今的社會中，資訊安全是一個被各界高度關注的議題，個人的行動裝置也不例外。在校園中也常發生造成遺憾的事件，無心的玩笑有可能衍生出嚴重的誤解，而這些皆肇因於手機螢幕解鎖的安全機制不甚完善，讓我們意識到提升解鎖安全性的重要性。現行之螢幕解鎖圖形皆為平面，推測將其擴展為立體可增加其組合方法數及破解難易度，使圖形在可記憶的範疇內達到安全性的提升。

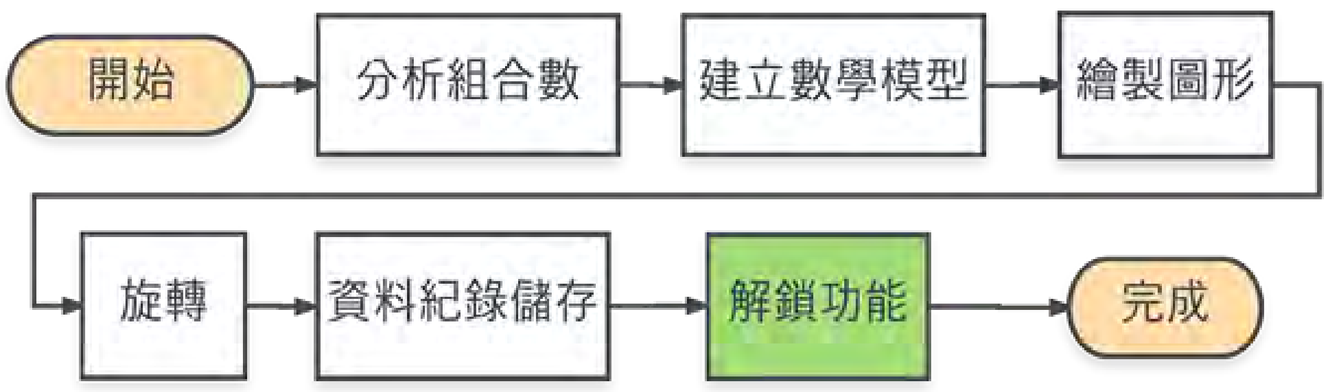
貳、研究目的

一、分析常見手機鎖之組合數

二、建立數學模型(正十二面體)

三、分析立體手機鎖之組合數

四、Android立體手機鎖的開發



圖一、研究進行之流程圖

參、研究設備及器材

- MacBook Pro (2015)
- ASUS M700-PU451LD
- Zenfone 2 (ASUS_Z00AD)
- ZenFone 3 Max (ZC552KL)

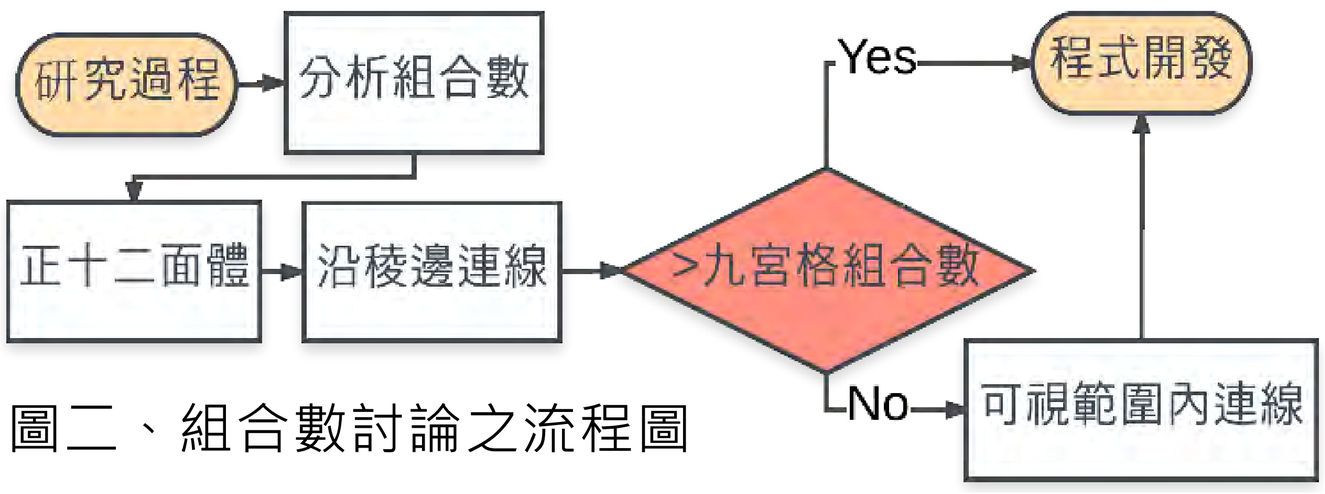
肆、研究過程或方法

一、分析比較組合數

(一) PIN 碼(4)：組合數有 **10000** 組，容易在輸入時被旁觀者記住密碼。

(二) 文字密碼：輸入 n 個字元時產生 95^n 組組合，其缺點和 PIN 碼相似，複雜度有所提升。

(三) 九宮格圖形：現行智慧型裝置所安裝的圖形鎖為 3x3 之九宮格圖形，我們參考 54 屆中小學科展「按圖鎖驥」[1]所討論之組合數，以電腦運算可行解的數量（連結點數大於三個）得出結果為 **389112** 組，與「按圖鎖驥」吻合。



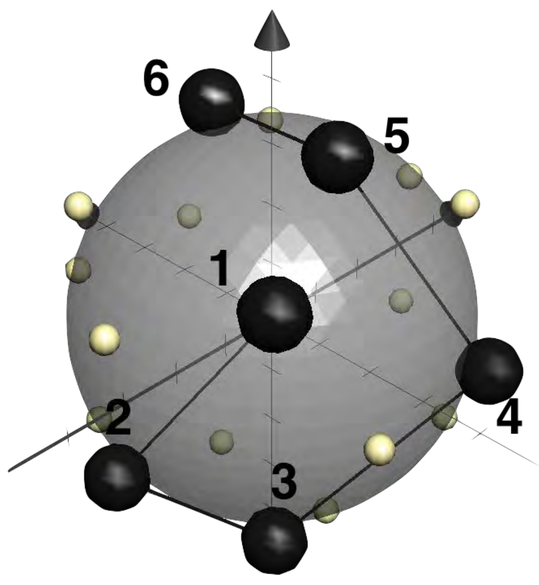
圖二、組合數討論之流程圖

二、建立數學模型--Grapher 繪製

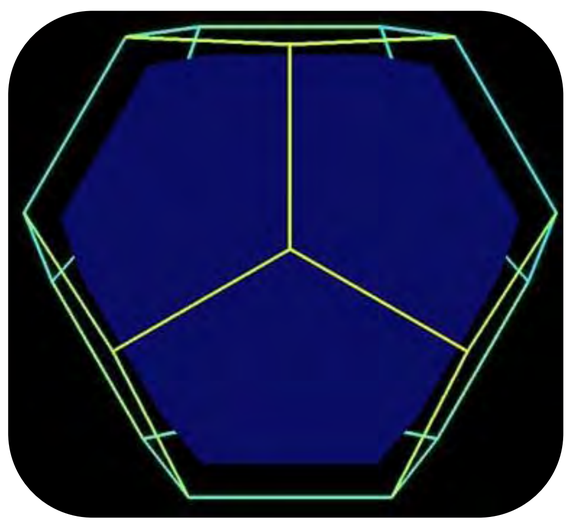
(一)我們將欲繪製之正十二面體各點距原點距離定為√3。

(二)以圓球輔助觀察，建立三維立體模型以觀察旋轉狀況。

(三)建立點與點之間的連線以模擬連線順序。



圖三、正十二面體的頂點在球上分布的情形



圖四、立體手機鎖在程式中繪製的情形

三、畫出立體圖形

(一) 設定座標、繪製圖形

以 r 表示其半徑，將一點固定在 (0,0,r) 的正十二面體的座標輸入，加以縮放後將形成邊的 30 對點座標，在執行階段保存在陣列內，導入 byte buffer 並彩現。

(二) 討論連線方式

1.沿稜邊：組合數共 **12538** 組，較九宮格少，無法增加組合數，需將稜邊連線以外的頂點納入討論。

2.**可視半球內**判定條件：

Algorithm 2 Connect in Vision

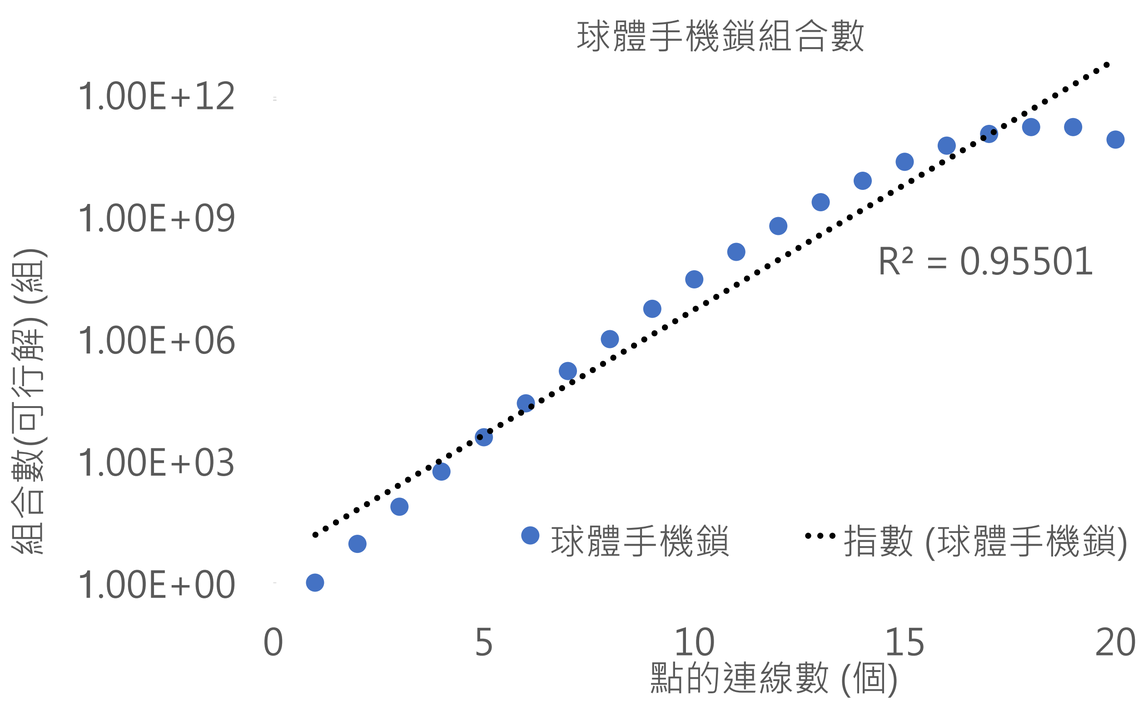
```
for all 0 ≤ i ≤ 19 do
    for all 0 ≤ j ≤ 19 do
        if i ≠ j then
            Matrix_ij = Distance between Vertix_i and Vertix_j ≤ √2r
        end if
    end for
end for
```

圖五、判定連線條件之pseudo code

在連接**6**點時組合數有 **24930** 組，與同點數連接時的平面九點手機鎖**26016**組相差不大，而**7**點以上則大幅超越。本立體手機鎖的組合數共 **574812244887** 組，研究採用此連線方式進行討論。

(三) 組合數討論

橫軸為起始點固定條件下的點連接數，縱軸為組合數（可行解）的對數值。當連接點數≦18時，組合數成指數增長，但在>18時下降，推測在連接第19個點時，**可視半球內**不符合連線的點比率上升，超越可連接之比率，許多原18個的連線無法往第19點連接，導致組合數下降，在連接20點時組合數下降更多。



圖六、點的連線數與組合數之關係散佈圖

(四) 旋轉

利用可取得的 Δx, Δy (手滑軌跡的位移差)，算出手機鎖該旋轉的角度。根據使用者的操作， Δx 將使手機鎖繞 y 軸旋轉， Δy 則繞 x 軸旋轉依據矩陣運算。

$$\begin{bmatrix} \cos dx & 0 & \sin dx \\ \sin dy \sin dx & \cos dy & -\sin dy \cos dx \\ -\cos dy \sin dx & \sin dy & \cos dy \cos dx \end{bmatrix}$$

圖七、程式所使用的旋轉矩陣

(五) 追蹤點

實作方法：

```
for all Point P do
    if Distance between P and (0,0,r) ≤ DetectingZoneRadius and Matrix_PQ is True and Q ≠ P then
        connect to P
        Previous Point P = Q
    end if
end for
```

圖八、連接判定範圍的點之pseudo code

若有個點在可視半球面內，且未被連接過，一旦進入中央的判定範圍，程式會把點連接上。

(六) 資料儲存

將程式執行過程中的數值雜湊後儲存至內部空間

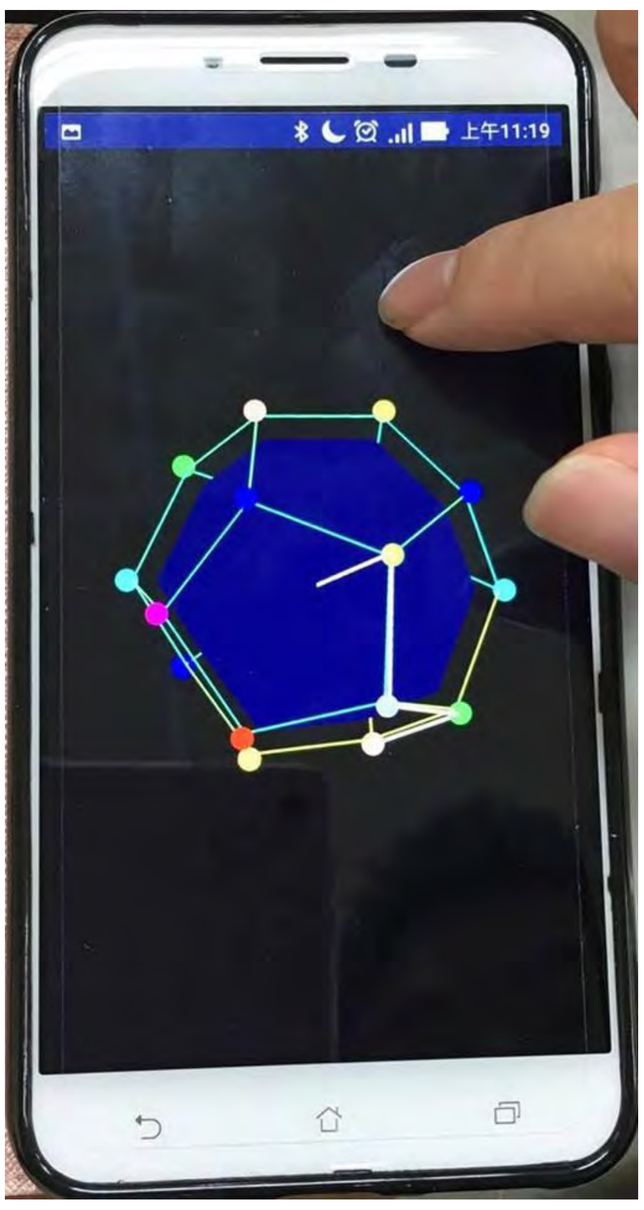
```
function DoubleHash(ArrayHashCode)
    ArrayHashCode = ArrayHashCode xor RandomSeed
    return md5(salt(ArrayHashCode))
end function
```

圖九、將數值進行雜湊之pseudo code

伍、研究結果

一、圖形及旋轉

已成功編寫出可在 Android 裝置上運行的應用程式，置於螢幕中央的正十二面體將沿手指滑動方向持續轉動，並在中央有一較小深色正十二面體方便分辨前後。各點以彩色表示方便辨識，已紀錄的點以白色連線標示，並設有旋轉速度上限。

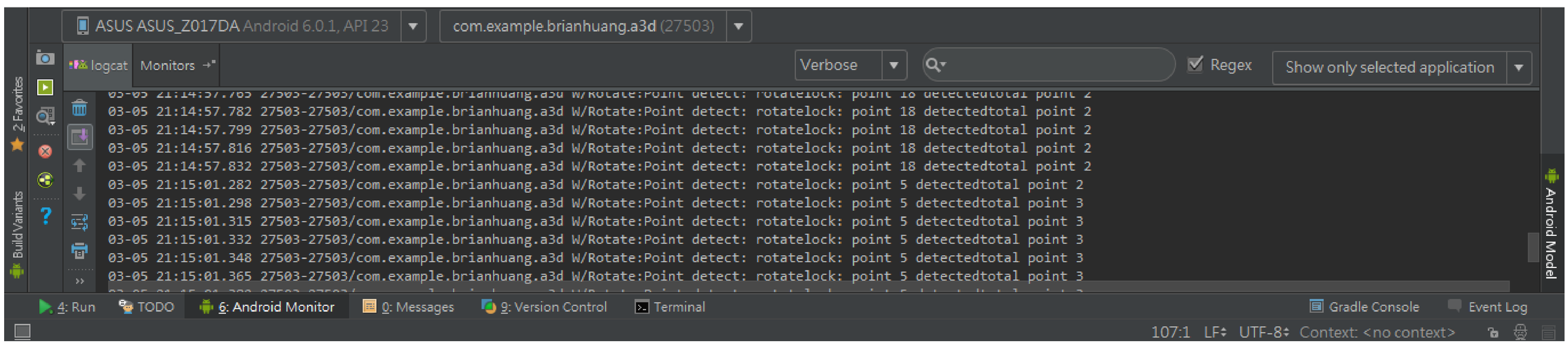


(左)圖十、應用程式於 ZenFone 3 Max 上之運行狀況（Android 6.0 API 23）

(右)圖十一、應用程式於模擬器（Nexus 5 API 23）之運行狀況

二、點的追蹤

可顯示進入視野中央特定範圍之頂點，並紀錄其與下一頂點連線順序。



圖十二、隨機滑動時於 Android Studio 的 event log 中所記錄到的點

三、導入解鎖功能

目前開發當中，須在應用程式運作時使實體按鍵無法操作，惟 Android 4.0 以上並無直接取消 home 鍵之系統功能，計畫將 home 鍵設為 custom launcher 實作

陸、結論

一、將點與點連線擴張至三維、更多頂點之圖形，可得到較原平面9點手機鎖更多的可行解數。

二、使用頂點數多，而圖形不致過於複雜者將可在使用者容易記憶的前提下，防止有心人士以窺視手指動作、軌跡的方法取得行動裝置內部資訊。

三、在起始點固定的情況下，若是連接6點，則立體手機鎖的組合數與平面9點手機鎖相去不遠，而在連接7點以上時，則大幅超越在平面9點手機鎖同點數連接時的組合數。

四、未來在圖形上做些改變，例如使用半正多面體，對組合數及其相關數據分析，以探討手機鎖的安全性，並加以改進。

七、參考資料與其他

[1]第54屆中小學科展 生活與應用學科 按圖鎖驥 黃昱睿;曾子洋;謝昊芫 2014年

[2] OpenGL (https://www.opengl.org/)

[3]Java™ PlatformStandard Ed. 8(http://docs.oracle.com/javase/8/docs/api/)

[4]OpenGL ES | Android Developers (https://developer.android.com/guide/topics/graphics/opengl.html)