國立中興大學資訊工程學系

資訊專題報告

題目:在區網內實現遊戲串流  
 (implementation of game streaming in local network)

專題題目說明，價值與貢獻自評:

本專題研究透過互相傳送螢幕畫面和鍵盤滑鼠訊號，藉此連線至另一台電腦遠端操作需要的應用程式，並且針對遊戲延遲及品質作特別優化

專題隊員

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | E-Mail | 負責項目說明 | 專題內貢獻度(%) |
| 游庭瑋 | edward.yu.0830@gmail.com | 組長 | 50 |
| 林子安 | hua10155174@gmail.com | 組員 | 50 |

指導教授簡述及簡評:

中 華 民 國 年 月 日

國立中興大學資訊工程學系

資訊專題報告

題目: 在區網內實現遊戲串流

(implementation of game streaming in local network)

指導教授: 王丕中 教授

學生: 資工三 游庭瑋

學生: 資工三 林子安

目錄

1. 摘要
2. 研究動機和目的
3. 重要貢獻
4. 團隊合作方式
5. 設計原理，研究方法和步驟
6. 系統實現和實驗
7. 效能評估和成果
8. 結論
9. 參考文獻

**一、摘要：**

現今不同的軟體相繼蜂湧而出，對於電腦的需求也逐漸提高，然而電腦配備價格並不是一般人隨意更換的，尤其在最近加密貨幣價格上漲，在大量挖礦的需求下，顯卡的價格更加高昂，下面我們蒐集了目前中高階顯卡的型號與價格做比較。

|  |  |  |
| --- | --- | --- |
| 顯卡型號 | 效能排行 | 最低價格（單位：新台幣） |
| NVIDIA RTX 3060 | 1 | 10900 |
| NVIDIA RTX 2060 | 2 | 12690 |
| NVIDIA GTX 1660 | 3 | 8940 |

（以上顯卡皆為桌面板）

（效能參考於<https://www.mydrivers.com/zhuanti/tianti/gpu/>，價格參考於<https://www.coolpc.com.tw/evaluate.php>）

一般人看到如此高昂的價格想必會精打細算，久久才會更換一次電腦設備，然而時間越久，這些設備會慢慢跟不上時代而又要讓荷包大失血．

此外不同的設備也許能運作相同的應用程式，但運作時產生的效果卻會有很大的差別，以下拿［古墓奇兵．起源］的高畫質和低畫質為例，其中最大差別大概就是光影的表現度了

高品質版本



低品質版本



雖然通常遊戲中可以自由地調整螢幕畫質，但是高畫質卻會對顯卡造成大量的負荷．

本專題以一個硬體提供商(server)的角度，運用遠端遙控讓客戶端(client)連線至設備較好的主機，值得一提的是只要網路環境允許，

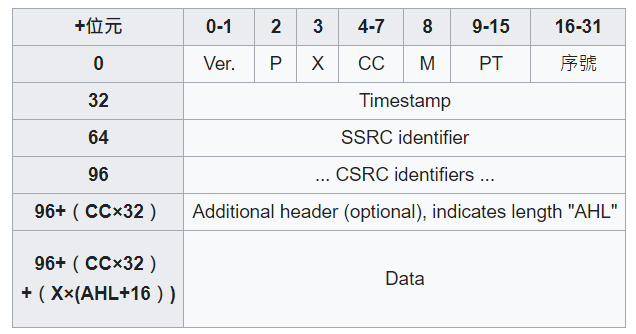
我們的程式希望使延遲盡可能降到最低；若是在不能降低延遲的情況下，則盡量使延遲穩定，才不會影響到使用者的操作體驗。

**二、研究動機與目的：**

現今不同的軟體相繼蜂湧而出，對於電腦的需求也逐漸提高，無論是ＡＩ訓練還是玩遊戲等等，效能較低的配備是很難跑得動的，因此在經費不足以購買更好的設備情況下，我們希望能連線至遠端較好的電腦來執行所需要的程式，另外有些人家裡會有桌上型電腦，而出門在外時只能使用筆電，此時若有資料是儲存於桌電的，也可以連線回去，並在自己的筆電上顯示出來

**三、專題重要貢獻：**

我們原本打算使用RTP作為傳輸的架構，但RTP多了很多我們不需要的參數，又或者是缺少的參數。因此在UDP上，利用自定義的封包頭結構，來達成無論在訊號或是畫面傳輸，甚至是品質控管，皆只須8 bytes即可處理。

RTP封包結構  


我們自定義的封包結構：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| +位元 | 0-7 | 7-16 | 17-23 | 24-31 |
| 0 | SEQ | | | |
| 32 | FRM | LAST | TYPE | FN |
| 64 | Timestamp | | | |

SEQ為封包的序號

FRM是當前傳遞的畫面序號，同一畫面為多個封包組成，因此也需記錄此數值

LAST為指定該封包是否為該畫面的最後一個封包

TYPE為傳遞封包的種類，我們定義了0為控制用，1為一般數據用

FN是針對TYPE=0所做的功能碼，其定義如下：

0-2為三向交握用：

0為三向交握的初始代碼

1為接收三向交握到連線

2為確認開始連線

3為控制停止訊號

4-7為品質控制：

4為要求降低畫質

5為要求提高畫質

6為對FN=4的ACK

7為對FN=5的ACK

8-9為螢幕解析度處理：

8指示payload為螢幕解析度參數，為一tuple=(width, height)

9為對FN=8的ACK

而針對滑鼠控制訊號，我們也定義了一個封包結構作為傳遞的標準：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| +位元 | 0-7 | 7-15 | 15-23 | 24-31 |
| 0 | TYPE | ACTION | X | |
| 32 | Y | | BTN | SPECIAL |

**四、團隊合作方式**

|  |  |
| --- | --- |
| **前期(9月~1月)** | 1. 規劃專題方向 2. 提出針對專題多種想法 3. 研究類似程式的設計方法 4. 設計GUI介面提供server和client使用 |
| **中期(2月~4月中)** | 1. 建立client和server連線 2. 先以json為基礎作為傳送格式 3. 分別傳送畫面和鍵盤滑鼠訊號 |
| **後期(4月~5月)** | 1. 製作專用封包 2. 將雙方程式碼合成至專用GUI中 3. 降低傳送畫面的延遲 4. 傳送聲音 5. 優化傳送滑鼠及鍵盤指令 6. 錄製結果影片 7. 撰寫專題報告 |

實驗室meeting時間:星期三下午4:00~5:00

個別工作表:

|  |  |  |
| --- | --- | --- |
|  | 林子安 | 游庭瑋 |
| **前期** | 1. 討論專題想法 2. 分配工作 | |
| **中期** | 1.鍵盤滑鼠訊號研究  2.嘗試在不同電腦間傳送訊號並操控 | 1. 畫面截圖數據研究  2. 畫面傳輸  3. 影像資料解碼 |
| **後期** | 1. 優化滑鼠、鍵盤指令 2. 程式合成 3. 完成畫面傳送 4. 動態優化傳輸品質 5. 錄製結果影片 6. 撰寫專題報告 | |

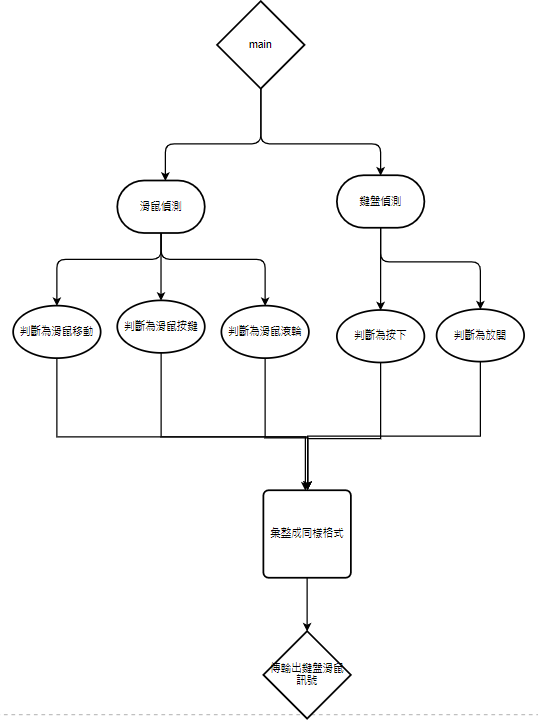
**五、設計原理、研究方法和步驟**

本專題主要分成３個部分，滑鼠鍵盤，畫面，以及封包傳輸

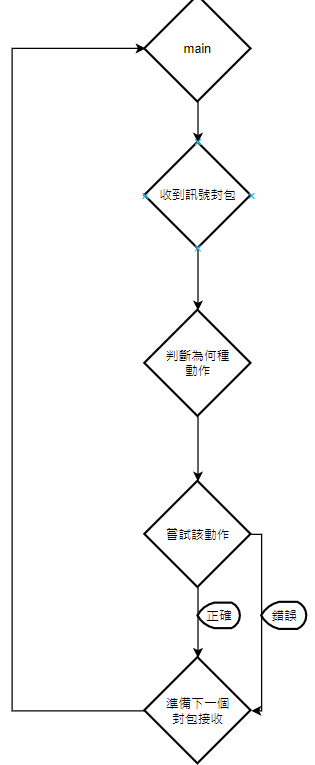
首先鍵盤滑鼠我們使用python 的pynput作為主要的開發套件，pynput最主要的功能有四種分別是

1. 滑鼠偵測
2. 滑鼠操控
3. 鍵盤偵測
4. 鍵盤控制

偵測的流程如下



控制的流程如下



以這四種功能作為主軸，我們的目的是想要達成在client端偵測滑鼠及鍵盤訊號並把所有不同作用的動作(滑鼠移動,滾輪,點擊，鍵盤按下和放開哪個按鍵)，統一做成同樣的封包格式一起傳送至server端，server端依照封包裡的訊號編碼進行client端要求的動作

接下來是畫面的部分，我們使用python的d3dshot + libjpeg turbo作為主要的開發套件，使用的功能有幾種分別是

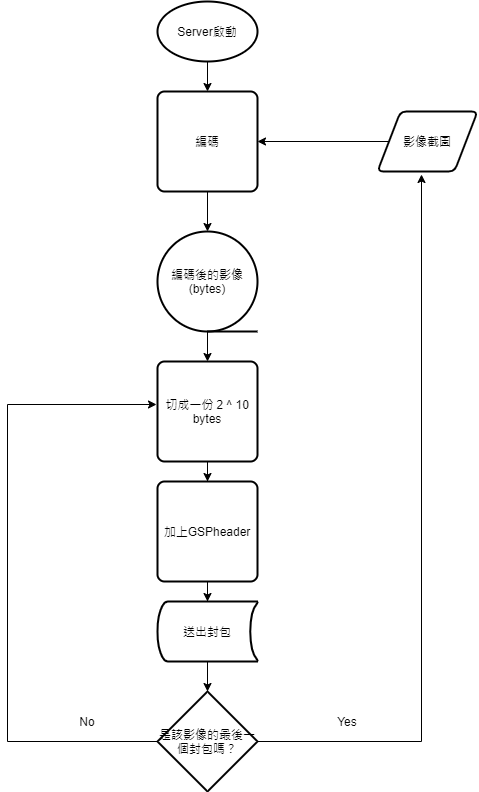
1. 螢幕截圖
2. 圖片編碼
3. 圖片切割
4. 圖片組合並呈現

**Server處理流程**

在截圖方面，我們使用目前速度最快的d3dshot套件，它是基於微軟的Windows Desktop Duplication API和DXGI、Direct3D等library所完成的，提供了非常高速的截圖速度，在內顯即可有每秒60影格的表現。

再來是圖片編解碼，我們使用libjpeg turbo來做編碼，並藉由PyTurboJPEG作為中介來省去直接操作API的麻煩，並且提供相較於opencv至少三倍的編解碼速度。

整體流程圖如下：



**Client處理流程**

**六、系統實現和實驗**

|  |  |  |
| --- | --- | --- |
|  | Client端 | Server端 |
| 處理器 | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz | Intel I5-4460 3.2ghz |
| 記憶體 | 8GB | 8GB |
| 顯示卡 | Nvidia Geforce MX150 | Nvidia Geforce GTX 1060 3GB |

鍵盤訊號傳遞成功:（圖片）

畫面截圖傳遞成功:（圖片）

畫面延遲秒數:

(圖片)

測試遊戲:古墓奇兵起源

以下為古墓奇兵 起源的最低配備要求

|  |  |
| --- | --- |
| 處理器 | Intel i3-3220 |
| 記憶體 | 8GB |
| 顯示卡 | Geforce GTX 660/GTX 1050 AMD Radeon HD 7770 |

以下為古墓奇兵 起源的建議配備:

|  |  |
| --- | --- |
| 處理器 | Intel Core I7-4770K /AMD Ryzen 5 1600 |
| 記憶體 | 16GB |
| 顯示卡 | Geforce GTX 1060 6GB、AMD Radeon RX 480 8GB |

測試遊戲:皮卡丘打排球:

由拍照畫面雖然可看出球的畫面還是有一點點的差距，但其實這個差距秒數已經在可以遊玩的接受範圍了，另外也可遠端進行對戰

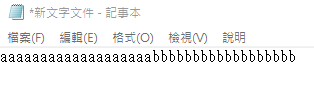
**七、效能評估與成果**

在製作中我們遇到了許多問題，也把一些重要的整理出來，在實驗的過程中我們更從這些問題中理解到一些關於開發網路和硬體訊號我們從未想過的事情．

在鍵盤滑鼠中，我們並不希望任何封包丟失，因此決定使用TCP來傳送封包，此時我們害怕當滑鼠快速移動或是按住一個按鍵不放開時，電腦會產生大量訊號而傳送時造成封包過多等等狀況，因此改良了一些演算法減少不必要的封包傳送數量，但有時封包延遲卻會造成server的動作不自然，因此在最後我們決定不使用json作為封包的包裝方法而是另外自己寫一個能盡可能讓封包大小減少的格式，此時就算大量傳送封包也不會造成server端接收及解碼方面的負擔．

另一個值得探討的是，我們在實驗中發現電腦在判斷鍵盤訊號時，如果還未放開前一個按鍵即按下下一個按鍵時，前一個按鍵訊號會被中斷改為下一個訊號，而前一個按鍵只有在放開時會輸出一個放開的訊號就結束工作了

圖為按下a後未放開即按下b的結果



當時我們研究了市面上不同鍵盤的說明，發現這是所有鍵盤共同的處理方式，在某些遊戲之所以可以同時判斷不同鍵盤(例如皮卡丘打排球的前進及跳躍)，應該是因為內部的演算法功能自行判斷的．

在畫面中我們一直在糾結使用UDP還是TCP方式傳送

表格為UDP及TCP在實作時的比較

|  |  |  |
| --- | --- | --- |
|  | UDP | TCP |
| 優點 | 1.畫面傳送速度快，延遲低  2.螢幕比例縮放處理較快 | 1.畫面穩定  2.首次連線時能保證有連線上 |
| 缺點 | 1.畫面不穩定，有時會有閃爍狀況  2.因畫面切片過，因此可能有”部分”畫面缺失 | 1.畫面延遲大 |

**八、結論**

單就畫面及訊號的傳遞方面我們的成果算是挺成功的，再延遲上甚至能跟市面上的遠端遙控軟體做比較，且我們的功能等等也不太多，因此網路的流量消耗也較低，不過當然這個程式也還有能成長的空間，未來如果還有在持續開發的話，我們會增加音訊的傳輸功能，配合每張畫面上的標籤，同步顯示在client的電腦上，鍵盤則是可以再

**九、參考文獻**

<https://pypi.org/project/d3dshot/>

<https://pypi.org/project/PyTurboJPEG/>