CS 4332 網路程式設計

期末專題報告

組員：9562119 藍家駿，9562171 雷禹恆，9562228 林哲男

題目：Pacman-Online

動機：
在要決定project題目時，適逢Pacman 30週年，Google在首頁上放了一個單機版的Pacman遊戲。因此我們有了以單機版的Pacman作為藍本，開發多人可玩而且是網路版的構想，而將題目命名為Pacman-Online。主要標榜的特色是：1. 玩家不僅可扮演Pacman，還可扮演Monster。2. 遊戲模式變成團體對戰，Pacman一隊，Monster一隊。

開發環境：

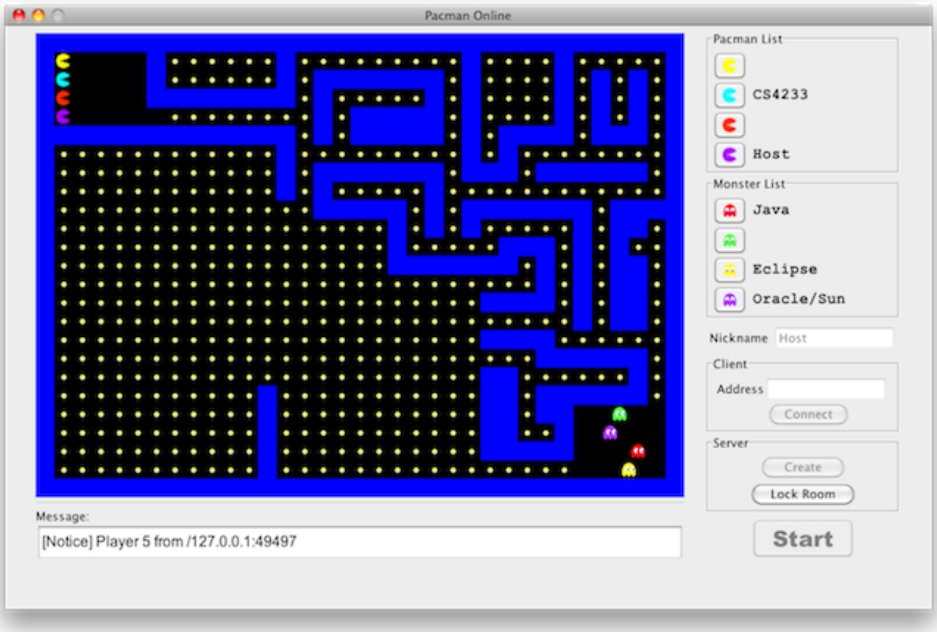
- 本專題使用Google Code專案託管 + Subversion版本控制：http://code.google.com/p/pacman-online
- 程式語言與IDE：Java + Eclipse。
- 測試作業系統：Windows 7、Windows XP、Mac OS X 10.6，理論上只要有Eclipse或JRE的系統都能執行。

使用說明：

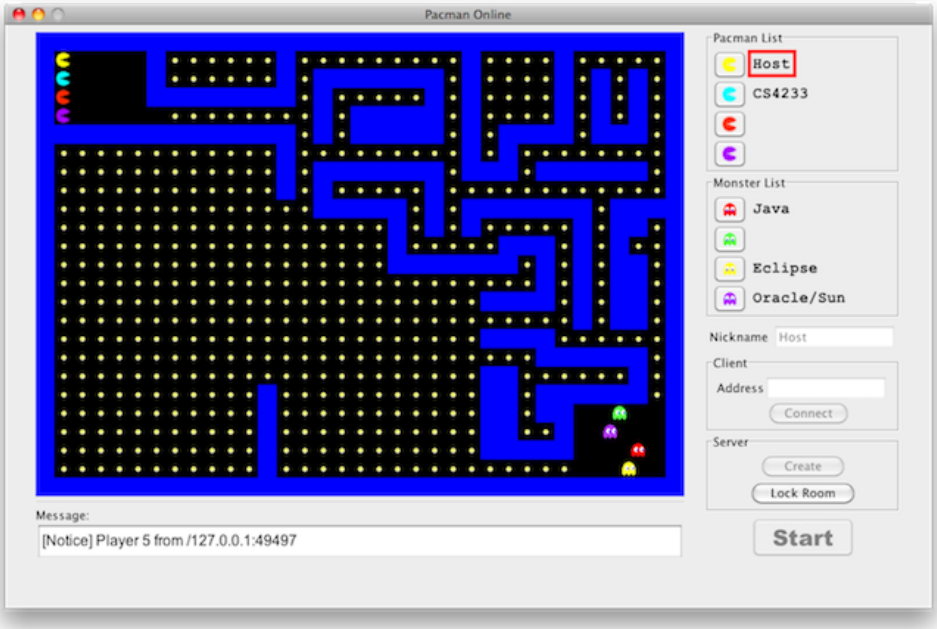
- Unix-like系統要用sudo啟動Eclipse，否則可能權限不足。
- 先打好自己的暱稱，若是server就按下*create*建立遊戲；若是client就打好server的位址，再按下*connect*連接遊戲。
- 可以選擇扮演pacman或是monster，並選擇沒被選過的角色圖示（未完成，只有server能選）。
- Server按下*Lock Room*以後，會鎖住房間不讓人進入，按下*Unlock Room*可復原。
- Clients已準備好進入遊戲，可按下*Ready*表示自己已準備好，若要反悔就按*Regret*。
- Server必須等到所有clients都Ready，*Start*鍵才會被enable，按下*Start*即可開始遊戲。
- 遊戲操作方式即規則如同一般熟知的Pacman遊戲，只是我們想要加入操縱Monster的玩法。

遊戲畫面：

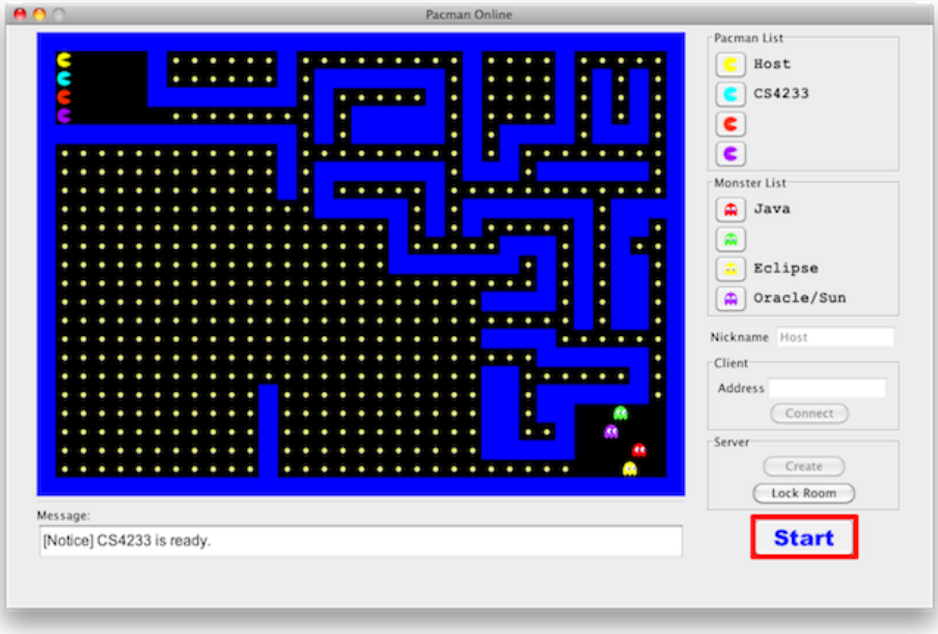
1. Server建立遊戲以後，clients可陸續加入，遊戲最多允許8人。



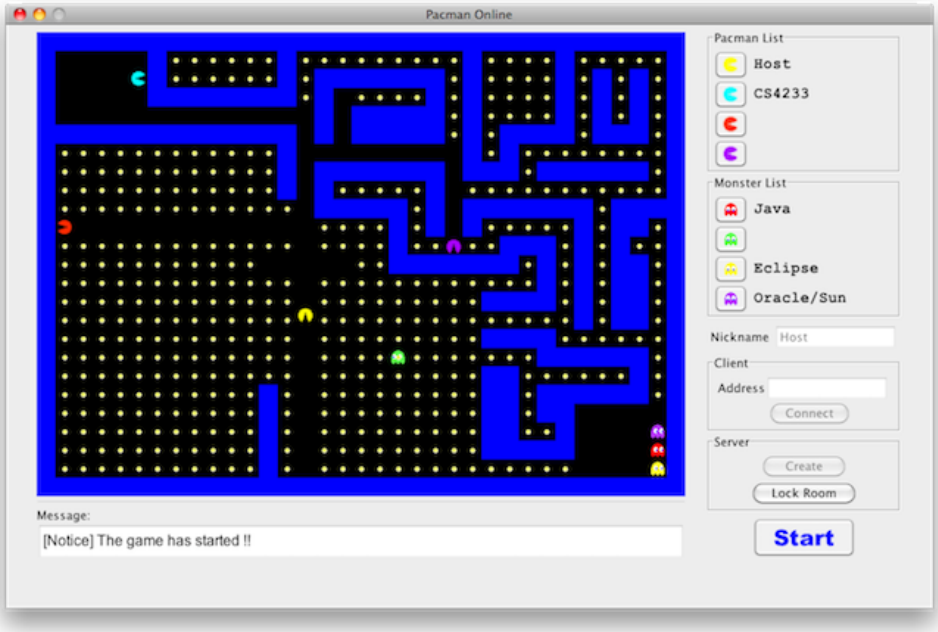
2. 選擇角色（目前只有server可選擇角色，並只能顯示在server的畫面上）。



2.
3. 當所有clients都按下Ready後，server的Start按鈕會開啟，按下後即開始遊戲。



4. Pacman-Online遊戲中的畫面。



程式設計與問題解決：

* Network

1. 使用TCP網路架構配上getInputStream()、getOutputStream()來重導I/O，然後自行定義char型態的client-server訊息，寫在Messages.java這個public interface。
2. accept導致 I/O blocking => Thread, start(), not run()
3. client/server都要有個Thread去處理收送訊息(避免I/O Blocking)
4. client自我感覺良好(isConnected只檢查server是否存在及port是否開啟)
    1. 一定要送個訊息試試才能確定有沒有真的accept
    2. setSoTimeout()可以設定等幾毫秒就timeout(一開始對著socket設定timeout時間)
        1. 只有對著read()或accept()才可以catch SocketTimeoutException。
        2. 其他java.net的method沒辦法。
5. 使用TCP connection, 強制送出client direction需要靠PrintStream.flush()。
6. Server可選擇"Lock Room / Unlock Room"，但為了防止同時有client要加入，兩邊必須用Java的synchronized來同步。
7. server thread 不能停掉(loop forever)，client thread 則可以重複 new
8. 訊息傳遞有自訂編碼方式：一般訊息是一個START_COMMAND以後接著一個char的字元；字串訊息式一個START_MESSAGE以後接著一個String的字串。所有的client-server訊息傳遞都是透過這種方式編碼、解碼。
9. 因為ServerThread裡的accept是包在while(true)裡，為了處理訊息傳遞，每有一個client連入，都會再開一個MessageThread來當作專線。

* GUI

1. 我們限制nickname的長度小等於13字元，否則會讓整個版亂掉。在連線前就過濾**這個錯誤。**
2. 圖形介面的部分使用了一些java已經有的Layout Manager，大致上參照範例就可以使用。只是如果要對介面有比較好的掌控的話（如直接指定座標、大小），就不能使用Layout Manager，也就是所謂的*Absolute Layout*。但是有了好的掌控性，自然要犧牲方便性。下面就列出幾項和Absolute Layout有關的問題。
    1. **setPreferredSize的問題**：因為gamepanel沒有設定Layoutmanager，所以沒有預設大小。除非自行指定，否則看不見gamepanel。
    2. **setFocusable(false)的問題**：gamepanel也要手動設定focusable為true，不然foucs不在gamepanel上的話，KeyListener沒辦法接收到鍵盤指令。另外，在點選其他按鍵之後，也要request focus給gamepanel，否則還要用滑鼠點一下才能用鍵盤操控。
3. 複雜的GridBagLayout：右半邊的connectPanel使用gridbaglayout，但是由於結構複雜，弄了很久才搞定。不過girdbaglayout是個很強大的layoutmanager，使用得當可以搞定大部分的介面。

4. 用Java Sound API的Clip來播放音效，不過其支援的檔案格式有點差。同樣是wav檔，從網路上下載的，在getAudioInputStream()出現UnsupportedAudioFileException；後來用Audacity這套軟體匯入再匯出成wav檔後，才能直接播放，實際原因我們也不太清楚。

* Game

1. 地圖的座標和陣列引數的概念一開始搞不清楚卡了一陣子。
2. setVisible(false) when eating beans
    1. 發現一開始就new好全部物件比較不會有殘影的問題
    2. 被吃掉的豆子只要利用setVisible(false)讓它變成看不見就好
3. 依照方向改變玩家嘴巴方向：這個效果純粹是在實際上轉彎時把顯示的圖片換掉。
4. 吃掉別人的判定
    1. 在同一個Grid內的判定是根據在螢幕上的座標，如果已經深入下一行或下一列超過一半，就會被認定為是下一行或下一列。
    2. 被吃掉的玩家要回到重生點。
5. 行進時效果
    1. 行進基本上是靠一個timer控制，每隔一段時間就會往目前方向前進一點。
    2. 我們現在作到的效果是你按方向鍵不會馬上轉方向，直到真的可以轉彎的地方才會真的轉。這效果是先儲存按鍵所指的方向，然後不斷試著往這個方向移動，如果可行再把這個方向改為目前的方向。
    3. 之前做到一半時有發生一種很好玩的現象就是，Pacman嘴巴的方向在還沒真的轉彎一按就會變了，導致看起來很像在甩尾過彎。這是因為在按按鍵時就換圖了。

* 放棄或來不及做到的想法

1. 只有Server的KeyListener是對應到直接修改座標值
    1. Client的KeyListen經測試需要放在ClientThread裡面才能一偵測到按鍵就從Socket送出
        1. Client的收資料必須要開另外一個ClientMessageThread
            1. 因為目前的Thread forever被KeyListener佔用
        2. 因為Timer只有Server能有
    2. Server的InputStream必須polling全部玩家
        1. 取得全部玩家的這個clock的前進方向才能運算下一個clock全部人的座標變化
2. Server的OutputStream必須根據timer
    1. 每次clock到的時候就送出PropertyChanged to Everyone
    2. client收到值的改變時就更新自己的gameFrame

3. Server的訊息傳遞，以無窮盡polling的方式藉由read()去問clients要傳過來的訊息；同樣道理，clients也以無窮盡polling的方式藉由read()去問server要傳過來的訊息。
4. **嚴格要求一來一往**
    1. **每個onClock server都一定要主動送**
        1. 送出
            1. 全部人的座標
            2. server的畫面..
        2. 接收
            1. 全部人的direction
            2. 如果有人沒送 player_timeout_counter++
            3. player_timeout_counter>2 視為斷線: kill client
    2. **client每收到一次message**
        1. 立刻回傳目前direction(用一個變數buffer著最後一次按的方向)
        2. 更新自己畫面

*** 未來的想法:**

1. 也許server要改成傳送所有值的改變Loading會比較小
    1. 值的改變大概可以用onPropertyChange: Put Into OutputStream
    2. PropertyChange要用什麼編碼技巧比較省封包可能需要思考一下