**俄羅方塊(魔術方塊，TETRIS)**

首先，讓我們來構思一下，俄羅斯方塊顯示的部分，該用什麼來實作呢？ 我採取了最簡單的Label(標籤)來完成，可是需要用總共200個Label，什麼？200個？你沒看錯，因為我們的俄羅斯方塊總共有20(列) X 10(行)個格子，你可以直接使用Visual Studio內的設計工具，拉出200個Label(那好累喔!)，也可以使用程式動態產生(嗯，這個比較好!)，然後再用程式把他們在視窗(Form)上面排好，OK! 這樣顯示的部分就完成了，就像這樣:

|  |  |
| --- | --- |
|  | Label[,] grids = new Label[20,10]; //main area, total 200 grids  for (int i = 0; i < 20; i++)  for (int j = 0; j < 10; j++)  {  grids[i, j] = new Label();  grids[i, j].Width = 30;  grids[i, j].Height = 30;  grids[i, j].BorderStyle = BorderStyle.FixedSingle;  grids[i, j].BackColor = Color.Black;  grids[i, j].Left = 150 + 30 \* j;  grids[i, j].Top = 600 - i \* 30;  grids[i, j].Visible = true;  this.Controls.Add(grids[i, j]);  } |

好啦，接下來當然就是"俄羅斯方塊"本尊了，俄羅斯方塊總共有七種，

|  |  |
| --- | --- |
| Type 1    Type 2    Type 3    Type 4 | Type 5    Type 6    Type 7 |

當方塊出現在最上方時，就只有這七種形式，但是在翻轉之後，會產生其他形式(黑點為參考點，紅點為旋轉支點，沒有紅點者即以參考點為支點)：

[https://1.bp.blogspot.com/-r0rJaqwFsnI/WgmaM2VqdxI/AAAAAAAAamc/TjXqdpzVKYg_M_pACjgMoYy457e7g7ICwCLcBGAs/s1600/type1d.jpg](https://1.bp.blogspot.com/-r0rJaqwFsnI/WgmaM2VqdxI/AAAAAAAAamc/TjXqdpzVKYg_M_pACjgMoYy457e7g7ICwCLcBGAs/s1600/type1d.jpg) [https://2.bp.blogspot.com/-EMY-BSqO2is/WgmaP9PS2NI/AAAAAAAAamg/SxAxDihqhX83D-Y3AwUr1ZICoklGz1SAQCLcBGAs/s1600/type11d.jpg](https://2.bp.blogspot.com/-EMY-BSqO2is/WgmaP9PS2NI/AAAAAAAAamg/SxAxDihqhX83D-Y3AwUr1ZICoklGz1SAQCLcBGAs/s1600/type11d.jpg)  ( 順時鐘轉)

    type 1         type 11

[https://3.bp.blogspot.com/-VZB0y14VWao/WgmcBuuci0I/AAAAAAAAams/YzRjDg5CbHsNp356j59sCWfGXvcrj7oAACLcBGAs/s1600/type2d.jpg](https://3.bp.blogspot.com/-VZB0y14VWao/WgmcBuuci0I/AAAAAAAAams/YzRjDg5CbHsNp356j59sCWfGXvcrj7oAACLcBGAs/s1600/type2d.jpg)  ( 不旋轉)

        type 2

[https://2.bp.blogspot.com/-B5RewNM7gbQ/WgmcQ-DiryI/AAAAAAAAam0/ZBPi79ADM8Yf3_6TuEvi1TNgzmZHCzwzwCLcBGAs/s1600/type3d.jpg](https://2.bp.blogspot.com/-B5RewNM7gbQ/WgmcQ-DiryI/AAAAAAAAam0/ZBPi79ADM8Yf3_6TuEvi1TNgzmZHCzwzwCLcBGAs/s1600/type3d.jpg) [https://2.bp.blogspot.com/-qJU0SQc43uU/WgmcQ2BRSOI/AAAAAAAAamw/wrxMaWr7iFsd-s2NMYFo0YquubxiWDnjQCLcBGAs/s1600/type13d.jpg](https://2.bp.blogspot.com/-qJU0SQc43uU/WgmcQ2BRSOI/AAAAAAAAamw/wrxMaWr7iFsd-s2NMYFo0YquubxiWDnjQCLcBGAs/s1600/type13d.jpg)  ( 逆時鐘轉)

         type 3                 type 13

[](https://4.bp.blogspot.com/-Ew0kWBnW6Vk/WgmcpfXSCvI/AAAAAAAAanE/S7CINWBI8OYH9FkRZT4T2ag55hQ8o3czQCLcBGAs/s1600/type4d.jpg) [](https://2.bp.blogspot.com/-ZKpmFgKNu8g/Wgmch_YV1WI/AAAAAAAAam4/4an9DqXfTsADJWamQyftQ9hKZkFXcspIQCLcBGAs/s1600/type14d.jpg) ( 逆時鐘轉)

        type 4                    type 14

[https://3.bp.blogspot.com/-aae4NxH-paA/Wgmc9Rzii1I/AAAAAAAAanU/vIhIUz37bn8RDzxTVvKErs9GLfwFfo5-ACLcBGAs/s1600/type5d.jpg](https://3.bp.blogspot.com/-aae4NxH-paA/Wgmc9Rzii1I/AAAAAAAAanU/vIhIUz37bn8RDzxTVvKErs9GLfwFfo5-ACLcBGAs/s1600/type5d.jpg) [https://2.bp.blogspot.com/-DqYEI1vzJzA/Wgmc8gd7F4I/AAAAAAAAanI/yZEtBL9K46cPukpQNU5Fl3uHwslvaM2vwCLcBGAs/s1600/type15d.jpg](https://2.bp.blogspot.com/-DqYEI1vzJzA/Wgmc8gd7F4I/AAAAAAAAanI/yZEtBL9K46cPukpQNU5Fl3uHwslvaM2vwCLcBGAs/s1600/type15d.jpg) [https://4.bp.blogspot.com/-JBHO3B1GHdM/Wgmc8_rn3KI/AAAAAAAAanQ/n61CHqiWoDcB_l4WS4P9-BeSKSlOL0cgwCLcBGAs/s1600/type25d.jpg](https://4.bp.blogspot.com/-JBHO3B1GHdM/Wgmc8_rn3KI/AAAAAAAAanQ/n61CHqiWoDcB_l4WS4P9-BeSKSlOL0cgwCLcBGAs/s1600/type25d.jpg) [https://1.bp.blogspot.com/-wU0eVH7wurI/Wgmc8rRUBOI/AAAAAAAAanM/l6ubDHrOkPUEvTjL_FFFDb-x_vZwCrCAQCLcBGAs/s1600/type35d.jpg](https://1.bp.blogspot.com/-wU0eVH7wurI/Wgmc8rRUBOI/AAAAAAAAanM/l6ubDHrOkPUEvTjL_FFFDb-x_vZwCrCAQCLcBGAs/s1600/type35d.jpg) ( 逆時鐘轉)

       type 5                    type 15                type 25               type 35

[](https://4.bp.blogspot.com/-oWEJtSki8Pc/WgmdWBk5VwI/AAAAAAAAano/BPe-42ecY_0-5eEAXP2Y3tIdPMC9hEoywCLcBGAs/s1600/type6d.jpg) [](https://2.bp.blogspot.com/-rxt_T0Mnke8/WgmdVcCogwI/AAAAAAAAanc/ME_lMOnA2cAVdVjUR0WKbWbM75S6qKovQCLcBGAs/s1600/type16d.jpg) [](https://3.bp.blogspot.com/-LhdY-5DP1Ss/WgmdVfJNZXI/AAAAAAAAang/_Csc81NDyCUKJDgtkSydG9iNPgKP_TfkgCLcBGAs/s1600/type26d.jpg) [](https://4.bp.blogspot.com/-MVpHBfhjMnQ/WgmdVf7Q-4I/AAAAAAAAank/k_C-531q2WwPjKEscojoRBoGpJsCSUA1QCLcBGAs/s1600/type36d.jpg)  ( 順時鐘轉)

          type 6                   type 16             type 26                 type 36

[https://1.bp.blogspot.com/-jrNq2oj02j0/WgmfDDU3yjI/AAAAAAAAaoA/NjyrKuDlLXY0lDoahoG79E7BzHr9Ac5_gCLcBGAs/s1600/type7d.jpg](https://1.bp.blogspot.com/-jrNq2oj02j0/WgmfDDU3yjI/AAAAAAAAaoA/NjyrKuDlLXY0lDoahoG79E7BzHr9Ac5_gCLcBGAs/s1600/type7d.jpg) [https://3.bp.blogspot.com/-XZJibvSkdZw/WgmfDsZ24YI/AAAAAAAAan8/Fh8r0GGkq78vul9FYRQyCDLyB5QLPVfigCLcBGAs/s1600/type17d.jpg](https://3.bp.blogspot.com/-XZJibvSkdZw/WgmfDsZ24YI/AAAAAAAAan8/Fh8r0GGkq78vul9FYRQyCDLyB5QLPVfigCLcBGAs/s1600/type17d.jpg) [https://1.bp.blogspot.com/-vq24ifTH6cA/WgmfCamfaJI/AAAAAAAAan4/Y1VP5u9WV2Mpo_mhbgrAddgKV1Wyp6eNQCLcBGAs/s1600/type27d.jpg](https://1.bp.blogspot.com/-vq24ifTH6cA/WgmfCamfaJI/AAAAAAAAan4/Y1VP5u9WV2Mpo_mhbgrAddgKV1Wyp6eNQCLcBGAs/s1600/type27d.jpg) [https://4.bp.blogspot.com/-hEufjrnsuhg/WgmfCdvyxNI/AAAAAAAAan0/UVeGYH028k0saQSY52PyqCb2XC4Z4IotgCLcBGAs/s1600/type37d.jpg](https://4.bp.blogspot.com/-hEufjrnsuhg/WgmfCdvyxNI/AAAAAAAAan0/UVeGYH028k0saQSY52PyqCb2XC4Z4IotgCLcBGAs/s1600/type37d.jpg)  ( 順時鐘轉)

         type 7                 type 17                 type 27               type 37

以上是我的旋轉規則和分類方式，你也可以改用你喜歡的方式，

分類完之後，就要開始寫程式了：我用了幾個全域變數紀錄遊戲進行的狀況，包括一個24x10的二維陣列(bool Array)，紀錄那些格點被佔用了，還有現在正掉落的的方塊的型態和參考點位置，當然你還得啟動一個計時器，由計時器的tick來觸發方塊的掉落，當方塊觸底或被之前的方塊擋住時，就要馬上停止，並啟動下一個方塊。

這個函數根據方塊的型態，和位置，將它紀錄在格點中

void update\_block(uint i, uint j, uint type)

{

switch (type)

{

case 1:

signs[i, j] = signs[i+1, j] = signs[i+2, j] = signs[i+3, j] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i + 2, j] = grids\_color[i + 3, j] = Color.Blue;

break;

case 11:

signs[i, j] = signs[i, j+1] = signs[i, j+2] = signs[i, j+3] = true;

grids\_color[i, j] = grids\_color[i, j + 1] = grids\_color[i, j + 2] = grids\_color[i, j + 3] = Color.Blue;

break;

case 2:

signs[i, j] = signs[i + 1, j] = signs[i , j+1] = signs[i + 1, j+1] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i, j + 1] = grids\_color[i + 1, j + 1] = Color.Yellow;

break;

case 3:

signs[i, j] = signs[i + 1, j] = signs[i+1, j-1] = signs[i, j + 1] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i + 1, j - 1] = grids\_color[i, j + 1] = Color.Red;

break;

case 13:

signs[i, j] = signs[i - 1, j] = signs[i , j + 1] = signs[i+1, j + 1] = true;

grids\_color[i, j] = grids\_color[i - 1, j] = grids\_color[i, j + 1] = grids\_color[i + 1, j + 1] = Color.Red;

break;

case 4:

signs[i, j] = signs[i , j-1] = signs[i + 1, j] = signs[i+1, j + 1] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i + 1, j] = grids\_color[i + 1, j + 1] = Color.Green;

break;

case 14:

signs[i, j] = signs[i+1, j] = signs[i, j+1] = signs[i-1, j + 1] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i, j + 1] = grids\_color[i - 1, j + 1] = Color.Green;

break;

case 5:

signs[i, j] = signs[i+1, j] = signs[i + 1, j+1] = signs[i + 1, j + 2] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i + 1, j + 1] = grids\_color[i + 1, j + 2] = Color.Orange;

break;

case 15:

signs[i, j] = signs[i, j-1] = signs[i + 1, j - 1] = signs[i + 2, j -1] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i + 1, j - 1] = grids\_color[i + 2, j - 1] = Color.Orange;

break;

case 25:

signs[i, j] = signs[i-1, j] = signs[i - 1, j - 1] = signs[i -1, j - 2] = true;

grids\_color[i, j] = grids\_color[i - 1, j] = grids\_color[i - 1, j - 1] = grids\_color[i - 1, j - 2] = Color.Orange;

break;

case 35:

signs[i, j] = signs[i, j+1] = signs[i - 1, j + 1] = signs[i - 2, j +1] = true;

grids\_color[i, j] = grids\_color[i, j + 1] = grids\_color[i - 1, j + 1] = grids\_color[i - 2, j + 1] = Color.Orange;

break;

case 6:

signs[i, j] = signs[i + 1, j] = signs[i + 1, j - 1] = signs[i + 1, j - 2] = true;

grids\_color[i, j] = grids\_color[i + 1, j] = grids\_color[i + 1, j - 1] = grids\_color[i + 1, j - 2] = Color.LightBlue;

break;

case 16:

signs[i, j] = signs[i, j+1] = signs[i + 1, j + 1] = signs[i + 2, j + 1] = true;

grids\_color[i, j] = grids\_color[i, j + 1] = grids\_color[i + 1, j + 1] = grids\_color[i + 2, j + 1] = Color.LightBlue;

break;

case 26:

signs[i, j] = signs[i-1, j] = signs[i-1, j + 1] = signs[i -1, j + 2] = true;

grids\_color[i, j] = grids\_color[i - 1, j] = grids\_color[i - 1, j + 1] = grids\_color[i - 1, j + 2] = Color.LightBlue;

break;

case 36:

signs[i, j] = signs[i, j-1] = signs[i - 1, j - 1] = signs[i - 2, j -1] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i - 1, j - 1] = grids\_color[i - 2, j - 1] = Color.LightBlue;

break;

case 7:

signs[i, j] = signs[i, j-1] = signs[i, j+1] = signs[i+1, j] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i, j + 1] = grids\_color[i + 1, j] = Color.Purple;

break;

case 17:

signs[i, j] = signs[i, j + 1] = signs[i-1, j] = signs[i + 1, j] = true;

grids\_color[i, j] = grids\_color[i, j + 1] = grids\_color[i - 1, j] = grids\_color[i + 1, j] = Color.Purple;

break;

case 27:

signs[i, j] = signs[i, j - 1] = signs[i, j+1] = signs[i - 1, j] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i, j + 1] = grids\_color[i - 1, j] = Color.Purple;

break;

case 37:

signs[i, j] = signs[i, j - 1] = signs[i+1, j] = signs[i - 1, j] = true;

grids\_color[i, j] = grids\_color[i, j - 1] = grids\_color[i + 1, j] = grids\_color[i - 1, j] = Color.Purple;

break;

}

}

這個函數偵測垂直方向是否已經落底或有障礙物:

bool y\_direction(uint type, uint i, uint j)

{

switch (type)

{

case 1:

if (i != 0 && !signs[i-1, j]) return true;

else return false;

case 11:

if (i != 0 && !signs[i - 1, j] && !signs[i - 1, j + 1] && !signs[i - 1, j + 2] && !signs[i - 1, j + 3]) return true;

else return false;

case 2:

if (i != 0 && !signs[i-1, j] && !signs[i-1, j+1]) return true;

else return false;

case 3:

if (i != 0 && !signs[i, j-1] && !signs[i-1, j] && !signs[i-1, j+1]) return true;

else return false;

case 13:

if (i != 1 && !signs[i-2, j] && !signs[i-1, j+1]) return true;

else return false;

case 4:

if (i != 0 && !signs[i, j+1] && !signs[i-1, j] && !signs[i-1, j-1]) return true;

else return false;

case 14:

if (i != 1 && !signs[i-1, j] && !signs[i-2, j+1]) return true;

else return false;

case 5:

if (i != 0 && !signs[i-1, j] && !signs[i, j+1] && !signs[i, j+2]) return true;

else return false;

case 15:

if (i != 0 && !signs[i - 1, j] && !signs[i-1, j-1]) return true;

else return false;

case 25:

if (i != 1 && !signs[i - 2, j] && !signs[i - 2, j - 1] && !signs[i - 2, j - 2]) return true;

else return false;

case 35:

if (i != 2 && !signs[i - 1, j] && !signs[i - 3, j + 1]) return true;

else return false;

case 6:

if (i != 0 && !signs[i, j-1] && !signs[i, j-2] && !signs[i-1, j]) return true;

else return false;

case 16:

if (i != 0 && !signs[i-1, j] && !signs[i-1, j+1]) return true;

else return false;

case 26:

if (i != 1 && !signs[i-2, j] && !signs[i-2, j + 1] && !signs[i-2, j+2]) return true;

else return false;

case 36:

if (i != 2 && !signs[i-1, j] && !signs[i-3, j-1]) return true;

else return false;

case 7:

if (i != 0 && !signs[i-1, j-1] && !signs[i-1, j] && !signs[i-1, j+1]) return true;

else return false;

case 17:

if (i != 1 && !signs[i-2, j] && !signs[i - 1, j+1]) return true;

else return false;

case 27:

if (i != 1 && !signs[i - 1, j - 1] && !signs[i - 1, j + 1] && !signs[i - 2, j]) return true;

else return false;

case 37:

if (i != 1 && !signs[i-2, j] && !signs[i-1, j-1]) return true;

else return false;

default:

return false;

}

}

**這個函數偵測水平方向是否有障礙物:**

bool x\_direction(uint type, uint i, uint j, int d)

{

switch(type)

{

case 1:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j - 1] && !signs[i + 2, j - 1] && !signs[i + 3, j - 1]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i + 1, j + 1] && !signs[i + 2, j + 1] && !signs[i + 3, j + 1]) return true;

else return false;

}

case 11:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1]) return true;

else return false;

}

else

{

if (j != 6 && !signs[i, j + 4]) return true;

else return false;

}

case 2:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j - 1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 2]) return true;

else return false;

}

case 3:

if (d == -1)

{

if (j != 1 && !signs[i, j - 1] && !signs[i + 1, j - 2]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 1]) return true;

else return false;

}

case 13:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j] && !signs[i + 1, j-1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 2] && !signs[i - 1, j + 1]) return true;

else return false;

}

case 4:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i + 1, j - 1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 1] && !signs[i + 1, j + 2]) return true;

else return false;

}

case 14:

if (d == -1)

{

if (j != 0 && !signs[i, j-1] && !signs[i+1, j-1] && !signs[i-1, j]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 1] && !signs[i-1, j + 2]) return true;

else return false;

}

case 5:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j - 1]) return true;

else return false;

}

else

{

if (j != 7 && !signs[i, j + 1] && !signs[i + 1, j + 3]) return true;

else return false;

}

case 15:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i + 1, j - 2] && !signs[i + 2, j - 2]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i + 1, j] && !signs[i + 2, j]) return true;

else return false;

}

case 25:

if (d == -1)

{

if (j != 2 && !signs[i, j-1] && !signs[i-1, j-3]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i-1, j+1]) return true;

else return false;

}

case 35:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i-1, j] && !signs[i-2, j]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i-1, j+2] && !signs[i-2, j+2]) return true;

else return false;

}

case 6:

if (d == -1)

{

if (j != 2 && !signs[i, j - 1] && !signs[i + 1, j - 3]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i + 1, j + 1]) return true;

else return false;

}

case 16:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j] && !signs[i + 2, j]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 2] && !signs[i + 2, j + 2]) return true;

else return false;

}

case 26:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i - 1, j-1]) return true;

else return false;

}

else

{

if (j != 7 && !signs[i, j + 1] && !signs[i - 1, j + 3]) return true;

else return false;

}

case 36:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i - 1, j - 2] && !signs[i - 2, j - 2]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i - 1, j] && !signs[i - 2, j]) return true;

else return false;

}

case 7:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i + 1, j - 1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 1]) return true;

else return false;

}

case 17:

if (d == -1)

{

if (j != 0 && !signs[i, j - 1] && !signs[i + 1, j - 1] && !signs[i - 1, j - 1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i + 1, j + 1] && !signs[i - 1, j + 1]) return true;

else return false;

}

case 27:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i - 1, j - 1]) return true;

else return false;

}

else

{

if (j != 8 && !signs[i, j + 2] && !signs[i - 1, j + 1]) return true;

else return false;

}

case 37:

if (d == -1)

{

if (j != 1 && !signs[i, j - 2] && !signs[i + 1, j - 1] && !signs[i - 1, j - 1]) return true;

else return false;

}

else

{

if (j != 9 && !signs[i, j + 1] && !signs[i + 1, j + 1] && !signs[i - 1, j + 1]) return true;

else return false;

}

default:

return false;

}

}

**這個函數檢查是否有填滿的"列"，如果有將其刪除，並將剩餘的格點往下向一列，直到沒有可刪除的列:**

void full\_line\_check()

{

uint row\_sum;

uint i, j;

i = 0;

while(i < 20)

{

row\_sum = 0;

for (j = 0; j < 10; j++)

if (signs[i, j]) row\_sum++;

if (row\_sum == 10)

{

score += 20;

label\_score.Text = "Score:" + score.ToString();

for (j = 0; j < 10; j++)

signs[i, j] = false;

show\_grids(); // show a black line

for (uint y = i; y < 21; y++)

for (j = 0; j < 10; j++)

{

signs[y, j] = signs[y + 1, j];

grids\_color[y, j] = grids\_color[y + 1, j];

}

show\_grids();

}

else i++;

}

}

**每次timer tick要做的事: 基本上就是檢查方塊是否遇到障礙物，如果沒有就往下降一格；如果方塊遇到障礙物就停止並檢查是否有可刪除的列，之後再降下新的方塊，最後就是顯示格點內的最新狀況。**

private void timer1\_Tick(object sender, EventArgs e)

{

if (y\_direction(block\_type, block\_row, block\_col))

{

block\_row\_pre = block\_row; block\_row\_pre = block\_row; block\_type\_pre = block\_type;

block\_row--;

if (block\_row == 19)

{

block\_type\_next = (uint)rander.Next(0, 7) + 1;

display\_next\_block(block\_type\_next);

block\_count++;

score += 5;

label\_block\_count.Text = "Blocks:" + block\_count.ToString();

label\_score.Text = "Score:" + score.ToString();

if (game\_mode == 1)

{

timer\_interval = 1010 - (int)(score / 150) \* 50;

if (timer\_interval <= 0)

timer\_interval = 10;

timer1.Interval = timer\_interval;

label\_level.Text = "Level:" + (1010 - timer\_interval) / 50;

}

}

erase\_block(block\_row\_pre, block\_col\_pre, block\_type\_pre);

update\_block(block\_row, block\_col, block\_type);

show\_grids();

block\_row\_pre = block\_row;

block\_changed = false;

}

else

{

show\_grids();

full\_line\_check();

if (block\_row == 20)

{

label\_info.Text = "Game Over!";

button1.Visible = true;

button1.Enabled = true;

timer1.Enabled = false;

return;

};

block\_type = block\_type\_next;

block\_row = 20;

block\_col = 4;

block\_row\_pre = 20;

block\_col\_pre = 4;

block\_type\_pre = block\_type;

block\_changed = false;

}

}

另外，偵測鍵盤:

包括左右方向鍵(左右移動):

當方塊觸及左右邊界時，須停止移動。

下方向鍵(將方塊快速拉下): 當方塊遇到障礙物，須停止下降。

空白建(翻轉方塊):

根據方塊目前的型態並偵測是否可進行翻轉(空間是否足夠?)，條件符合才進行翻轉，否則型態不變。

偵測按鍵，並做對應的動作:

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.P)

{

if (game\_mode == 0) { game\_mode = 1; timer1.Enabled = true; }

else { game\_mode = 0; timer1.Enabled = false; }

}

if (e.KeyCode == Keys.Left)

{

if (x\_direction(block\_type, block\_row, block\_col, -1))

{

block\_col\_pre = block\_col; block\_col--;

block\_changed = true;

}

}

if (e.KeyCode == Keys.Right)

{

if (x\_direction(block\_type, block\_row, block\_col, 1))

{

block\_col\_pre = block\_col; block\_col++;

block\_changed = true;

}

}

if (e.KeyCode == Keys.Space)

{

block\_type\_pre = block\_type;

block\_col\_pre = block\_col; block\_row\_pre = block\_row;

block\_type = next\_block\_type(block\_type, block\_row, block\_col);

if(block\_type != block\_type\_pre)

block\_changed = true;

}

if (e.KeyCode == Keys.S)

{

game\_mode = 2;

timer\_interval -= 50;

if(timer\_interval <= 0)

timer\_interval = 1;

timer1.Interval = timer\_interval;

label\_level.Text = "Level:" + (1010 - timer\_interval) / 50;

}

if (e.KeyCode == Keys.A)

{

game\_mode = 2;

timer\_interval += 50;

if (timer\_interval >= 1010)

timer\_interval = 1010;

timer1.Interval = timer\_interval;

label\_level.Text = "Level:" + (1010 - timer\_interval) / 50;

}

if (e.KeyCode == Keys.Down)

timer1.Interval = 15;

if (block\_changed)

{

erase\_block(block\_row\_pre, block\_col\_pre, block\_type\_pre);

update\_block(block\_row, block\_col, block\_type);

show\_grids();

block\_row\_pre = block\_row; block\_col\_pre = block\_col; block\_type\_pre = block\_type;

block\_changed = false;

}

}

**檢查方塊是否可翻轉，並做對應的動作:**

uint next\_block\_type(uint type, uint i, uint j)

{

switch(type)

{

case 1:

if (j <= 7 && j>=1 && !signs[i+2, j-1] && !signs[i+2, j + 1] && !signs[i+2, j + 2])

{

block\_row = i + 2; block\_col = j - 1;

return 11;

}

else return 1;

case 11:

if (i >= 2 && !signs[i - 1, j+1] && !signs[i - 2, j+1] && !signs[i +1, j+1])

{

block\_row = i -2; block\_col = j + 1;

return 1;

}

else return 11;

case 2: return 2;

case 3:

if (i >= 1 && !signs[i + 1, j+1] && !signs[i-1, j])

return 13;

else return 3;

case 13:

if (j >= 1 && !signs[i + 1, j] && !signs[i+1, j-1])

return 3;

else return 13;

case 4:

if (i >= 1 && !signs[i, j+1] && !signs[i-1, j+1])

return 14;

else return 4;

case 14:

if (j >= 1 && !signs[i, j-1] && !signs[i + 1, j + 1])

return 4;

else return 14;

case 5:

if (!signs[i + 2, j] && !signs[i, j + 1])

{

block\_col = j + 1;

return 15;

}

else return 5;

case 15:

if (j >= 2 && !signs[i, j-2] && !signs[i + 1, j] )

{

block\_row = i + 1;

return 25;

}

else return 15;

case 25:

if (i >= 2 && !signs[i, j - 1] && !signs[i - 2, j])

{

block\_col = j - 1;

return 35;

}

else return 25;

case 35:

if (j <= 7 && !signs[i - 1, j] && !signs[i, j + 2])

{

block\_row = i - 1;

return 5;

}

else return 35;

case 6:

if (!signs[i, j - 1] && !signs[i + 2, j] )

{

block\_col = j - 1;

return 16;

}

else return 6;

case 16:

if (j <= 7 && !signs[i - 1, j] && !signs[i, j + 2])

{

block\_row = i + 1;

return 26;

}

else return 16;

case 26:

if (i >= 2 && !signs[i, j +1] && !signs[i - 2, j])

{

block\_col = j + 1;

return 36;

}

else return 26;

case 36:

if (j >= 2 && !signs[i, j-2] && !signs[i - 1, j])

{

block\_row = i - 1;

return 6;

}

else return 36;

case 7:

if (i>=1 && !signs[i-1, j])

return 17;

else return 7;

case 17:

if (j >= 1 && !signs[i, j-1])

return 27;

else return 17;

case 27:

if (!signs[i+1, j])

return 37;

else return 27;

case 37:

if (j<=8 && !signs[i, j+1])

return 7;

else return 37;

default: return 0;

}