Final Project — Tetris

金家逸 B10502076

陳竣瑋 B10502056

林桓鈺 B10502013

遊戲玩法：

遊戲共有七種方塊，開始時會隨機落下其中一種;在落下期間，玩家可以左右平移、90度的旋轉方塊，當方塊落至最下方區域或者無法再向下移動時，方塊將會固定在該處，並且在面板上方會再隨機的產生另一塊方塊。當玩家將同一列的格子用方塊填滿時，即可將該列消除並且得到分數;同一次消除的列數越多，就可以得到越多的分數。當玩家分數越高，方塊下落的速度會越快。

實現操作:

1. 當方塊下落時，按壓左、右方向鍵可以使方塊平移。
2. 當方塊下落時，按壓空白鍵可以使方塊加速落至最下方。
3. 當方塊下落時，按壓上方向鍵可以使方塊旋轉。
4. 當方塊下落時，按壓 z 鍵可以使當前方塊與hold欄的方塊互換，被交換的方塊將會於下一回合出現;此功能於每回合只能使用一次（有一個方塊掉落至底部稱作為一回合）。
5. 在遊戲的右上方Next顯示下一回合的方塊。
6. 分數的計算。
    - 一回合內消除 1 行，得分數100。
    - 一回合內消除 2 行，得分數200。
    - 一回合內消除 3 行，得分數400。
    - 一回合內消除 4 行，得分數800。

7. 計分板。

當遊戲結束時，若玩家的分數為排行榜上的前五名，玩家可以在window上輸入玩家名稱。

遇到的問題：

在過程中最主要的問題是不太熟悉SDL這項新的技術，所以花了不少時間在網路上查資料，去學習SDL的功能以及語法，以下是參考的網站。

https://lazyfoo.net/tutorials/SDL/

如何執行：

直接在main.cpp裡執行即可。

外部使用的 library 有：

1. SDL2
2. SDL2_ttf
3. SDL2_mixer
4. SDL2_image

程式碼介紹：

```cpp
#include <iostream>
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include <SDL_mixer.h>
#include <string>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <fstream>
using namespace std;

const int ScreenW = 740;     //遊戲視窗的寬度
const int ScreenH = 780;     //遊戲視窗的長度
const int BlockW = 36;       //方塊的寬度
const int BlockH = 36;       //方塊的長度
const int Lines = 20;        //有20個橫列
const int Cols = 10;         //有10個直行
int gamevel = 400;           //遊戲的初始速度

//Globally used font
TTF_Font *gFont = NULL;

//The music that will be played
Mix_Music *gMusic = NULL;

SDL_Window* window = NULL;

SDL_Renderer* render = NULL;

SDL_Texture* gameover = NULL;
SDL_Texture* scoreboard = NULL;

string inputText = "";

string name[5];      //計分板上的名字
int scores[5];       //計分板上名字所對應到的分數

```

- 宣告關於遊戲的一些變數。

`class LTexture`

```
39 //Texture wrapper class
40 class LTexture
41 {
42 public:
43     //Initializes variables
44     LTexture();
45
46     //Deallocates memory
47     ~LTexture();
48
49 #if defined(SDL_TTF_MAJOR_VERSION)
50     //Creates image from font string
51     bool loadFromRenderedText( std::string textureText, SDL_Color textColor , SDL_Renderer* gR
52 #endif
53
54     //Deallocates texture
55     void free();
56
57     //Renders texture at given point
58     void render( int x, int y, SDL_Renderer* gRenderer, SDL_Rect* clip = NULL, double angle =
           SDL_FLIP_NONE );
59
60     //Gets image dimensions
61     int getWidth();
62     int getHeight();
63
64 private:
65     //The actual hardware texture
66     SDL_Texture* mTexture;
67
68     //Image dimensions
69     int mWidth;
70     int mHeight;
71 };
72
73 //Scene textures
74 LTexture gPromptTextTexture1;    //提示當玩家遊戲結束時的名次
75 LTexture gPromptTextTexture2;    //提示當玩家遊戲結束時的需要輸入名字 (Please Enter Your Name)
76 LTexture gInputTextTexture;      //玩家輸入名字的訊息
77 LTexture playername[5];          //儲存前五名玩家的名字
78 LTexture playerscore[5];         //儲存前五名玩家的分數
79
```

- 將圖片、文字都包裝，並渲染到遊戲視窗的物件（為固定寫法）。

`void clean();`

```
922 void LTexture::free()
923 {
924     //Free texture if it exists
925     if( mTexture != NULL )
926     {
927         SDL_DestroyTexture( mTexture );
928         mTexture = NULL;
929         mWidth = 0;
930         mHeight = 0;
931     }
932 }
```

- 關閉SDL套件。

- 以下是方塊的物件

`class Tetris`

```
class Tetris
{
public:
    void setCurrentTime(Uint32 t)
    {
        currentTime = t;
    }
    bool isrunning()
    {
        return running;
    }
    bool isvalid();
    int loadLeaderboard();
    void save_leaderboard(int);
    bool init(const char* title);   //初始化SDL套件
    bool loadMedia();               //將圖片、文字、音檔渲染至視窗
    void nextTetrimino();
    void swapnextTetrimino();
    void handleEvents();
    void setRectPos(SDL_Rect& rect, int x = 0, int y = 0,int w = BlockW, int h = BlockH);
    void moveRectPos(SDL_Rect& rect, int x, int y);
    void gameplay();
    void updateRender();            //更新螢幕畫面
    void clean();

private:
    SDL_Texture* background = NULL, * blocks = NULL;
    SDL_Rect srcR = { 0, 0, BlockW, BlockH }, destR =
        { 0, 0, BlockW, BlockH }, holdR = { 0, 0, BlockW, BlockH }, tempR = { 0, 0, BlockW, BlockH };
    SDL_Rect nextR = { 0, 0, BlockW, BlockH };
    //Rendered texture
    LTexture leveltext;
    LTexture scoretext;
    LTexture nexttext;
    LTexture holdtext;

    int field[Lines][Cols] = {0};
    static const int figures[7][4];
    int temp_figure[4];
    int next_figure[4];             //下一個圖形
    struct Point                    //方塊的位置
    {
        int x, y;
    } items[4], backup[4];
    int color = 1 + rand() % 7;     //隨機產生一種顏色
    int kind = rand() % 7;          //隨機產生一個種類
    int next_kind;                  //下一個方塊的種類
    int next_color;                 //下一個方塊的顏色
    int dx = 0;                     //左右的一小段位移
    int score = -100;               //分數
    int level = 1;                  //遊戲難度
    int time = 1;                   //在一回合中消除的行數
    int temp_kind;                  //現在方塊的種類
    int temp_color;                 //現在方塊的顏色
    int delay = gamevel;            //遊戲速度
    bool have_rows = false;         //判斷是否有一橫列全部填滿
    bool rotate = false;            //是否執行旋轉
    bool swap = false;              //是否執行交換
    bool can_swap = true;           //是否可以交換
    bool running = false;           //遊戲是否結束
    bool first = true;
    Uint32 startTime = 0, currentTime = 0;
};
```

```
bool isvalid();
```

```
651 bool Tetris::isvalid()
652 {
653     for (int i = 0; i < 4; i++)
654         if (items[i].x < 0 || items[i].x >= Cols || items[i].y >= Lines)
655             return false;
656         else if (field[items[i].y][items[i].x])
657             return false;
658     return true;
659 }
```

- 因為每種方塊都由四顆小方形所組成，所以用for迴圈判斷四次；654 行的if是判斷各個小方塊是否有移動超出邊界；第656行中的是判斷各個小方塊是否有碰到其他方塊，若有碰到其他方塊則field [ ][ ]中的值不為零。

```
int loadLeaderboard();
```

```
163 int Tetris::loadLeaderboard() {
164     int n = -1;
165     fstream file;
166     file.open("rank.txt");
167     for (int i = 0; i < 5; i++) {
168         file >> name[i] >> scores[i];
169     }
170     for (int i=0 ; i<4 ; ++i){
171         for (int j=0 ; j<4-i ; ++j){
172             if (scores[j] < scores[j+1]){
173                 int temp = scores[j];
174                 scores[j] = scores[j+1];
175                 scores[j+1] = temp;
176                 string s = name[j];
177                 name[j] = name[j+1];
178                 name[j+1] = s;
179             }
180         }
181     }
182     for (int i = 0; i < 5; i++) {
183         if (score > scores[i]) {
184             n = i;
185             break;
186         }
187     }
188     file.close();
189     return n;
190 }
191
```

- 第164至169行：將rank.txt打開，並將檔案裡面的資料讀入。第170至第181行：將rank.txt中的資料做排序（由大到小）。第182至188行：判斷此玩家的分數是否在記分板的前五名，若在前五名則回傳他的名次n，否則回穿-1。

```
void save_leaderboard(int);
```

```
192 void Tetris::save_leaderboard(int i /* i 為名次*/) {
193     fstream file;
194     file.open("rank.txt");
195     bool name_exist = false;
196     for (int i=0 ; i<5 ; ++i){
197         if (name[i] == inputText){
198             name_exist = true;
199             scores[i] = score;
200             break;
201         }
202     }
203     if (name_exist == true){
204         for (int i=0 ; i<4 ; ++i){
205             for (int j=0 ; j<4-i ; ++j){
206                 if (scores[j] < scores[j+1]){
207                     int temp = scores[j];
208                     scores[j] = scores[j+1];
209                     scores[j+1] = temp;
210                     string s = name[j];
211                     name[j] = name[j+1];
212                     name[j+1] = s;
213                 }
214             }
215         }
216     } else {
217         for (int j = 4; j > i; j--) {
218             scores[j] = scores[j-1];
219             name[j] = name[j - 1];
220         }
221         scores[i] = score;
222         name[i] = inputText;
223     }
224     for (int i = 0; i < 5; i++) {
225         file << name[i] << " " << scores[i] << "\n";
226     }
227     file.close();
228 }
229
```

- 第196至202行：判斷此玩家是否已經出現在記分板上，若已經出現過則將其分數更新。第203至223行：若名字已經出現在記分板上則將計分板重新排序；否則將玩家的名字及分數插入名次 i 並且將之後的玩家名次往下移動。

```
static const int figures[7][4];
```

```
422 const int Tetris::figures[7][4] =
423 {
424     0,1,2,3,    // I
425     0,4,5,6,    // J
426     2,6,5,4,    // L
427     1,2,5,6,    // O
428     2,1,5,4,    // S
429     1,4,5,6,    // T
430     0,1,5,6,    // Z
431 };
432
433 /*
434     0    1    2    3
435     4    5    6    7
436 */
```

- 表示七種方塊的形狀。

```
void setRectPos(SDL_Rect& rect, int x = 0, int y = 0, int w = BlockW, int h = BlockH);
```

```
640 void Tetris::setRectPos(SDL_Rect& rect, int x, int y, int w, int h)
641 {
642     rect = { x, y, w, h };
643 }
644
```

- 設定方塊的位置。

```
void moveRectPos(SDL_Rect& rect, int x, int y);
```

```
645 void Tetris::moveRectPos(SDL_Rect& rect, int x, int y)
646 {
647     rect.x += x;
648     rect.y += y;
649 }
650
```

- 將方塊的位置移動。

```
void swapnextTetrimino();
```

```
563 void Tetris::swapnextTetrimino()
564 {
565     for (int i = 0; i < 4; i++)
566     {
567         items[i].x = figures[temp_kind][i] % 4;
568         items[i].y = int(figures[temp_kind][i] / 4);
569     }
570 }
571
```

- 將當前方塊與hold的方塊做交換。

```
void nextTetrimino();
```

```
544 void Tetris::nextTetrimino()
545 {
546     can_swap = true;
547     if (first != true){
548         color = next_color;
549         kind = next_kind;
550     }
551     first = false;
552     for (int i = 0; i < 4; i++)
553     {
554         items[i].x = figures[kind][i] % 4;
555         items[i].y = int(figures[kind][i] / 4);
556     }
557     next_kind = rand() % 7;
558     next_color =  1 + rand() % 7;
559     for (int i=0 ; i<4 ; ++i){
560         next_figure[i] = figures[next_kind][i];
561     }
562 }
```

- 此函數為產生新的方塊（也就是新一回合），所以將can_swap設為true；因為上一回合已經產生了這一回合的方塊種類及顏色（next_color及next_figure已經在上一回合就有指定值），所以在第548、549行就將next_color、next_figure指定給這回合的color、figure，接著再輸出方塊，並且再次產生新的next_color、next_figure。

```
void handleEvents();
```

```
572 void Tetris::handleEvents()
573 {
574     SDL_Event e;
575     while (SDL_PollEvent(&e))
576     {
577         switch (e.type)
578         {
579         case SDL_QUIT:
580             running = false;
581             break;
582         case SDL_KEYDOWN:
583             switch (e.key.keysym.sym)
584             {
585             case SDLK_UP:           //當方向鍵輸入「上」，旋轉方塊。
586                 rotate = true;
587                 break;
588             case SDLK_LEFT:         //當方向鍵輸入「左」，左移方塊。
589                 dx = -1;
590                 break;
591             case SDLK_RIGHT:         //當方向鍵輸入「右」，右移方塊。
592                 dx = 1;
594             case SDLK_z:            //當鍵盤輸入「z」，執行hold。
595                 if (can_swap == true){
596                     tempR = srcR;
597                     can_swap = false;
598                     if (swap == false){
599                         swap = true;
600                         for (int i=0 ; i<4 ; ++i){
601                             temp_figure[i] = figures[kind][i];
602                         }
603                         temp_kind = kind;
604                         temp_color = color;
605                         nextTetrimino();
606                         can_swap = false;
607                     } else {
608                         int temp;
609                         for (int i=0 ; i<4 ; ++i){
610                             temp_figure[i] = figures[kind][i];
611                         }
612                         swapnextTetrimino();
613                         temp = kind;
614                         kind = temp_kind;
615                         temp_kind = temp;
616                         temp = color;
617                         color = temp_color;
618                         temp_color = temp;
619                     }
620                 }
621                 break;
622             default:
623                 break;
624             }
625         default:
626             break;
627         }
628     }
629     const Uint8* state = SDL_GetKeyboardState(NULL);
630     if (state[SDL_SCANCODE_DOWN])       //當輸入方向鍵「下」，加速方塊掉落。
631         delay = 50;
632     if (state[SDL_SCANCODE_SPACE]){     //當鍵盤輸入「空白鍵」，方塊直接掉落。
633         if (delay != 0){
634             SDL_Delay(100);
635             delay = 0;
636         }
637     }
638 }
```

- 實現鍵盤操作。

```
void gameplay();
```

```
662 void Tetris::gameplay()
663 {
664
665     //backup
666     for (int i = 0; i < 4; i++)
667         backup[i] = items[i];
668     //move
669     if (dx)
670     {
671         for (int i = 0; i < 4; i++)
672         {
673             items[i].x += dx;
674         }
675         if (!isvalid())     //若此動作是不符合規定的 (isvalid()回傳false) 則將方塊保持在原來的狀態
676             for (int i = 0; i < 4; i++)
677                 items[i] = backup[i];
678     }
679
680     //rotate
681     if (rotate)
682     {
683         Point p = items[2]; // center of rotation
684         for (int i = 0; i < 4; i++)
685         {
686             int x = items[i].y - p.y;
687             int y = items[i].x - p.x;
688             items[i].x = p.x - x;
689             items[i].y = p.y + y;
690         }
691         if (!isvalid())
692             for (int i = 0; i < 4; i++)
693                 items[i] = backup[i];
694     }
```

- 先將方塊原先的位置儲存在backup，再依照要移動的方向去改變每一塊方塊的位置，接著用 isvalid（）去判斷次移動是否有違規（遇到其他方塊或者超出邊界），若 isvalid（ ） 為 false 則將方塊的位置保持在原來的位置（用backup覆蓋）。
- 將第三個方塊作為旋轉的中心，並坐旋轉；若 isvalid（ ） 為 false 則將方塊保持在原來的位置。

```
697     //當時間差大於dalay，方塊會下落一格。
698     if (currentTime - startTime > delay)
699     {
700         for (int i = 0; i < 4; i++)
701             backup[i] = items[i];
702
703         for (int i = 0; i < 4; i++){
704             items[i].y++;
705         }
706
707         //當isvalid()回傳false時，將方塊的位置固定在遊戲的畫面上。
708         if (!isvalid())
709         {
710             for (int i = 0; i < 4; i++)
711                 field[backup[i].y][backup[i].x] = color;
712             if (delay == 0){
713                 delay = gamevel;
714             }
715             nextTetrimino();
716         }
717
718         startTime = currentTime;
719     }
720
721     //check lines
722     //判斷遊戲是否結束
723     for (int j=0 ; j<Cols ; ++j){
724         if (field[0][j] != 0){
725             running = false;
726         }
```

- 當跑過的時間超過 delay 時，會將方塊的位置向下移動一格；若 isvalid （ ）為 false 時（代表方塊下方碰到其他方塊），則將方塊載入背景畫面中。
- 若最上方有方塊碰到邊界時，遊戲結束（running 為 false）。

```
    //判斷是否有一橫列都被方塊填滿
    int k = Lines - 1;
    for (int i = k; i > 0; i--)
    {
        int count = 0;
        for (int j = 0; j < Cols; j++)
        {
            if (field[i][j]){
                count++;
            }
            field[k][j] = field[i][j];
        }

        if (count == Cols){
            have_rows = true;
            time++;
        } else {
            k--;
            have_rows = false;
        }
    }

    dx = 0;
    rotate = false;
    if (delay == 0){
        delay = 0;
    } else {
        delay = gamevel;
    }
}
```

- 由最上方開始檢查，用count記錄地 i 橫列多少個格子已經被方塊填滿，當count和Cols的數目相同則表示此橫列已經全部被方塊填滿（have_row為true）。

```
void clean();
```

```
void Tetris::clean()
{
    //Free loaded images
    scoretext.free();
    leveltext.free();
    nexttext.free();
    gPromptTextTexture1.free();
    gPromptTextTexture2.free();
    gInputTextTexture.free();
    for (int i=0 ; i<5 ; ++i){
        playername[i].free();
        playerscore[i].free();
    }

    //Free global font
    TTF_CloseFont( gFont );
    gFont = NULL;

    //Free the music
    Mix_FreeMusic( gMusic );
    gMusic = NULL;

    //Destroy window
    SDL_DestroyRenderer( render );
    SDL_DestroyTexture(blocks);
    SDL_DestroyTexture(scoreboard);
    SDL_DestroyTexture(gameover);
    SDL_DestroyWindow( window );
    window = NULL;
    render = NULL;

    //Quit SDL subsystems
    IMG_Quit();
    SDL_Quit();
    TTF_Quit();
    Mix_Quit();
}
```

- 清除遊戲載入的圖片、視窗、音樂還有SDL套件。

- 以下為主程式。

```
int main(int argc, char* argv[])
```

```
347 int main(int argc, char* argv[])
348 {
349     //Start up SDL and create window
350     srand(time(0));
351     Tetris* tetris = new Tetris();
352     const char* title = "Tetris";
353 
354     if(tetris->init(title) != true)
355     {
356         printf( "Failed to initialize!\n" );
357     }
358     else
359     {
360         //Load media
361         if( !tetris->loadMedia() )
362         {
363             printf( "Failed to load media!\n" );
364         }
365         else
366         {
367             int score = 0;
368             int level = 1;
369 
370             //Play the music
371             Mix_PlayMusic( gMusic, -1 );
372             while (tetris -> isrunning())
373             {
374                 tetris->setCurrentTime(SDL_GetTicks());
375                 tetris->handleEvents();
376                 tetris->gameplay();
379 
380             //關閉遊戲
381             if (tetris -> isrunning() == false){
382                 Mix_HaltMusic();
383                 SDL_SetRenderDrawColor(render, 0xFF, 0xFF, 0xFF, 0xFF);
384                 SDL_RenderClear( render );
385                 SDL_Surface* loadSurf = IMG_Load("gameover.png");
386                 gameover = SDL_CreateTextureFromSurface(render, loadSurf);
387                 SDL_FreeSurface(loadSurf);
388                 SDL_RenderCopy(render, gameover, NULL, NULL);
389                 SDL_RenderPresent(render);
390             }
```

- 載入圖片、視窗以及音樂。當isrunning（）為true，持續執行遊戲；當isrunning（）為 false 時則終止遊戲並且將畫面改成gameover.png。

```
392             SDL_Delay(3000);
393             int rank = tetris -> loadLeaderboard();
394             if (rank != -1){
395                 //Render the prompt
396                 SDL_Color textColor = { 0, 0, 0, 0xFF };
397                 string s = "You are the " + to_string(rank+1) + " place";
398                 gPromptTextTexture1.loadFromRenderedText( s, textColor, render );
399                 gPromptTextTexture2.loadFromRenderedText( "Please enter your name:", textColor, render );
400                 handleuserinput();
401                 tetris -> save_leaderboard(rank);
402             }
403 
404             for (int i=0 ; i<5 ; ++i){
405                 SDL_Color textColor = { 0, 0, 0, 0xFF };
406                 playername[i].loadFromRenderedText( name[i], textColor, render );
407                 playerscore[i].loadFromRenderedText( to_string(scores[i]), textColor, render );
408             }
409 
410             handleScoreBoard();
411         }
412     }
413 
414     //Free resources and close SDL
415     tetris->clean();
416 
417     return 0;
418 }
```

- 當rank不等於-1（玩家的分數在前五名），就會請玩家輸入名字在記分板上做紀錄。

- 其他函數

```
int countscore(int time)
```

```
145 int countscore(int time){
146     //time表示在一回合消除幾行
147     if (time == 4){
148         return 800;
149     }
150 
151     if (time == 3){
152         return 400;
153     }
154 
155     if (time == 2){
156         return 200;
157     }
158 
159     if (time == 1){
160         return 100;
161     }
162 }
163 
164 int Tetris::loadLeaderboard() {
165     int n = -1;
166     fstream file;
167     file.open("rank.txt");
168     for (int i = 0; i < 5; i++) {
169         file >> name[i] >> scores[i];
170     }
171     for (int i=0 ; i<4 ; ++i){
172         for (int j=0 ; j<4-i ; ++j){
173             if (scores[j] < scores[j+1]){
174                 int temp = scores[j];
175                 scores[j] = scores[j+1];
176                 scores[j+1] = temp;
177                 string s = name[j];
178                 name[j] = name[j+1];
179                 name[j+1] = s;
```

- 計算分數。

組員分工：

組員名稱	組員學號	負責事項
金家逸	B10502076	程式設計、Debug
林桓鈺	B10502013	書面報告、Debug
陳竣瑋	B10502056	簡報、影片、美工、Debug