



中南大學
CENTRAL SOUTH UNIVERSITY

中南大学课程实践报告

实践名称：计算机程序设计实践

实践题目：贪吃蛇游戏

指导教师：

专 业：

班 级：1234 班

姓 名：R

学 号：

202 年 月 9 日

目录

一 总体设计.....	1
二 详细设计.....	2
三 源代码清单及说明.....	4
四 运行结果.....	17
五 心得体会.....	25

一 总体设计

实践题目

序号： 11
题目名称：贪吃蛇游戏
题目说明：贪吃蛇算法及图形编程库函数

我的设计目标

用 C 语言设计一款贪吃蛇游戏程序。创建一个二维的平面“地图”，地图边界是“墙”，“蛇”在游戏开始后会自动移动，用户通过按键控制“蛇”的下一移动方向，且不能原地掉头。随机在“地图”内生成一个“食物”，不过“食物”不可以生成在“蛇”和“墙”上，当“蛇”“吃到”一个“食物”时，“蛇身”会变长一节，同时再生成一个“食物”。当“蛇头”撞到“墙”或“蛇身”时，则“蛇”“死亡”，宣告游戏结束。

在游戏操作过程中，用户通过按键控制“蛇”的下一移动方向，也可以按键来实现暂停和退出游戏，当暂停时，按任意键继续。

游戏还设置了实时记分系统，每当“蛇”“吃到”一个“食物”，就会让分数加一，并设置记录历史最高分的功能。该系统会不断更新当前分数，实时地显示于“地图”的下方。每当一次游戏判定结束时，会告诉用户他的最终得分是多少分，如果用户此次游戏最终得分小于、等于、大于历史最高分时，会有不同的提示语。正常关闭游戏后再次打开游戏时，最高分数不会清零。

同时，在游戏程序的初始化界面，注明了游戏操作，按键说明，以及可以通过用户输入数字“1”、“2”、“3”来调整蛇的移动速度，这样可以达到调整游戏难度的目的。在游戏的结束界面，会显示用户本局游戏的最终得分和于历史最高得分的比较，并让用户按键选择，是重新开始游戏或是退出游戏。

二 详细设计

- 定义结构体变量 `head`，用来表示“蛇头”，“蛇头”中含有的属性是蛇头的二维坐标，蛇身的长度。
- 定义二维结构体数组 `body`，用来表示“蛇身”，含有每个蛇身的二维坐标。
- 定义全局变量二维数组 `maps`，用来划分游戏“地图”，便于设置“墙”，“蛇头”，“蛇身”，“食物”等不同的状态。
- 定义全局变量 `speed`，用来表示“蛇”自动移动时经历的那个循环周期的时长，所以 `speed` 越小，“蛇”移动速度越快，游戏难度越大。
- 定义全局变量 `grade`，用来实时表示当前游戏中的得分的数值。
- 定义全局变量 `best`，用来存储历史最高纪录的数值。
- 定义函数 `HideCursor`，用来隐藏光标，使程序界面更加美观。需要头文件 `<windows.h>` 来实现。
- 定义函数 `JumpTo(int,int)`，用来跳转光标，便于打印“墙”、“蛇”、“食物”对应的字符，甚至整个程序的背景颜色，以及打印一些文字提示，便于美化作品和提升人机交互。需要头文件 `<windows.h>` 来实现。
- 定义函数 `color(int)`，用来实现通过输入十进制代码输出各种颜色，美化作品。需要头文件 `<windows.h>` 来实现。
- 定义函数 `InitMap()`，用来在游戏中的那个界面生成“墙”，并将其坐标状态设置为“墙”，即使用循环结构把“墙”的位置打印字符以显示，同时也在“地图”之下显示分数系统和游戏作者。
- 定义函数 `InitSnake()`，用来在游戏中的那个界面生成“蛇”，给结构体变量 `head` 赋值初始化蛇头的位置为地图左上角且“蛇头朝右”，蛇身的长度为 2 节，并将其坐标状态设置为“蛇头”或“蛇身”。
- 定义函数 `InitFood()`，用来在游戏中的那个界面生成“食物”，且“食物”为随机生成，且其状态位置必须不能与“墙”、“蛇”等重合。调用 `rand()` 函数，同理将该坐标状态设置为“食物”，并打印其字符。
- 定义函数 `PrintSnake(int)`，用来打印“蛇”的字符。设置一个参数 `flag`，用来表示是打印新的“蛇头”还是用空格消除“旧蛇尾”。因为“蛇身”是沿着“蛇头”移动的，所以“蛇”的移动可用打印新的“蛇头”和用空格覆盖旧的“蛇身”最后一节来视觉体现。
- 定义函数 `AutoMove(int,int)`，用来实现“蛇”的移动。首先进入一个死循环，来实现蛇的自动移动功能，每循环一次为一个时间周期，通过再装入一个自变量自减的循环结构，改变其中自变量的数值大小即可控制循环周期的长度，进而实现“蛇”移动速度的改变，再进而实现游戏难度的调整。若在此循环周期内有新的来自键盘的输入，则跳出循环，回到函数 `Game()` 函数中，读取输入的键值实现相应的功能。
- 定义函数 `Move(int,int)`，主要是和 `PrintSnake(int)` 配套使用，用来提供该打印哪个坐标为“蛇”的“新脖颈”，该用空格消除哪个坐标为“蛇”的“旧尾巴”。最终视觉实现“蛇”的移动。
- 定义函数 `BeforeGame()`，用在生成在刚运行程序打开 `cmd` 窗口时的那个界面。首先通过打印空格调整整个背景色，视觉上使 `cmd` 窗口不再是黑色，提升视

觉效果。然后打印出游戏标题和游戏操作介绍，提供一个可以让玩家自由选择 3 种游戏难度的一个交互界面。最后，为了提升游戏体验，所以令用户按下难度选择时直接开始游戏，所以要调用一个清屏的 `cmd` 命令，然后初始化历史最高分、地图、蛇、食物，最后调用下一个函数 `Game()` 正式执行游戏主体。

- 定义函数 `Game()`，用来实现在游戏中通过读取用户按键的键值进行各种操作，如改变“蛇”的移动方向、暂停、退出游戏。先初始化蛇的移动方向，这里设置先向右移动比较合理。随后读取键值，判断是否是有效键值，并进入一个 `switch` 语句，根据不同的键值实现相应的功能。用户输入方向键“↑、↓、←、→”时，实现“蛇”改变移动方向的功能；用户输入空格键时，实现游戏暂停并可以按任意键继续的功能，用户输入 `ESC` 键时，实现退出游戏，即关闭窗口功能。
- 定义函数 `AfterGame(int,int)`，用来在蛇的自动移动的过程中，判断游戏是否结束，若“蛇头”即将移动到的坐标的状态为“食物”，则“蛇身长度”加一；若“蛇头”即将移动到的坐标的状态为“墙”或“蛇身”，则执行清屏，并输出显示游戏结束界面，在游戏的结束界面，会显示用户本局游戏的最终得分和于历史最高得分的比较，并让用户按键选择，是重新开始游戏还是退出游戏。
- 定义函数 `ReadGrade()`，用来读取相应文件中记录的变量的数值，来实现历史最高得分的保存。若当前文件夹中没有这个文件，则创建该文件。这里我给这个文件命名为 `TheSnake_HighLight_ByR.txt`。
- 定义函数 `WriteGrade()`，用来将新的历史最高记录赋值给相应文件中对应变量，来实现最高得分记录值的更新。这里我给这个文件命名为 `TheSnake_HighLight_ByR.txt`。
- 主函数：相对比较简单，需要为随机生成“食物”设置随机数生成点，命名窗口和设置窗口大小等等，最后只需要调用 `BeforeGame()` 即可。

三 源代码清单及说明

源代码如下：

（为了便于阅读，所以从 VSCode 带格式粘贴以显示颜色，颜色主题是 VSCode 的默认浅色主题）

头文件

```
#include <stdio.h>
#include <Windows.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
```

说明：需要的头文件。

宏定义

```
#define UP 72 //方向键：上
#define DOWN 80 //方向键：下
#define LEFT 75 //方向键：左
#define RIGHT 77 //方向键：右
#define SPACE 32 //暂停
#define ESC 27 //退出
#define ENTER 13 //确认

#define HANG 34 //游戏区行数
#define LIE 70 //游戏区列数

#define AIR 0 //标记状态为空气
#define WALL 1 //标记状态为墙
#define FOOD 2 //标记状态为食物
#define HEAD 3 //标记状态为蛇头
#define BODY 4 //标记状态为蛇身
```

说明：宏定义便于值的使用和修改。第一组宏定义为游戏需要按键的键值。第二组宏定义是界面大小，便于输出打印各种符号或文字。第三组是便于给不同坐标赋予不同的状态。

定义结构体

```
struct Snake
{
    int lenth; //蛇身长度
    int x; //蛇头横坐标
    int y; //蛇头纵坐标
}head;

struct Body
```

```
{  
    int x; //蛇身横坐标  
    int y; //蛇身纵坐标  
}body[HANG*LIE];
```

说明：结构体 **SnakeHead** 存储当前蛇身的长度以及蛇头的位置坐标。结构体 **SnakeBody** 存储着该段蛇身的位置坐标，并开辟足以存储蛇身的结构体数组，故数组大小只需要为 **HANG*LIE**。

定义全局变量

```
int maps[HANG][LIE];  
int best, grade, speed;
```

说明：二维数组 **maps**，用来表示游戏地图坐标。变量 **speed**，用来表示“蛇”自动移动时经历的那个循环周期的时长。变量 **grade**，用来实时表示当前游戏中的得分的数值。变量 **best**，用来存储历史最高纪录的数值。

函数声明

```
void HideCursor();  
  
void JumpTo(int x, int y);  
  
void color(int c);  
  
void InitMap();  
  
void InitSnake();  
  
void InitFood();  
  
void PrintSnake(int flag);  
  
void AutoMove(int x, int y);  
  
void Move(int x, int y);  
  
void BeforeGame();  
  
void Game();  
  
void AfterGame(int x, int y);  
  
void ReadGrade();  
  
void WriteGrade();
```

说明：该程序用到的所有函数的声明。

主函数

```
int main()
{
    best = 0, grade = 0;
    srand((unsigned int)time(NULL)); //为 rand()随机生成“食物”设置随机数生成点
    system("title 贪吃蛇游戏-                "); //窗口命名
    system("mode con cols=140 lines=35"); //设置窗口大小
    HideCursor();
    BeforeGame();
    return 0;
}
```

说明：主函数相对比较简单，基本只需要调用 BeforeGame()即可。

定义函数 HideCursor

```
void HideCursor()
{
    CONSOLE_CURSOR_INFO curInfo;
    curInfo.dwSize = 1;
    curInfo.bVisible = FALSE;
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorInfo(handle, &curInfo);
}
```

说明：该函数用来隐藏光标，使程序界面更加美观。需要头文件<windows.h>来实现。直接套用即可。

定义函数 JumpTo(int,int)

```
void JumpTo(int x, int y)
{
    COORD c;
    c.X = x;
    c.Y = y;
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(handle, c);
}
```

说明：该函数用来跳转光标到指定坐标（对应的行和列），便于下一步 printf 的使用，使程序界面更加美观。需要头文件<windows.h>来实现。直接套用即可。

定义函数 color(int)

```
void color(int c) //参数 c 为一个表示相应颜色的十进制数
{
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), c);
}
```

说明：用于调整下一步的打印颜色。需要头文件<windows.h>来实现。直接套用

即可。

定义函数 InitMap()

```
void InitMap()
{
    color(120); //背景色设置
    for (int i = 0; i < HANG; i++)
    {
        for (int j = 0; j < LIE; j++)
        {
            if (j == 0 || j == LIE - 1)
            {
                maps[i][j] = WALL; //纵向的墙
                JumpTo(2 * j, i);
                printf("■");
            }
            else if (i == 0 || i == HANG - 1)
            {
                maps[i][j] = WALL; //横向的墙
                printf("■");
            }
            else //背景
            {
                maps[i][j] = AIR;
                printf(" ");
            }
        }
    }
    color(112); //文字提示
    JumpTo(2, HANG);
    printf("当前得分:%d 分", grade);
    JumpTo(LIE / 2, HANG);
    printf("历史最佳:%d 分", best);
    JumpTo(2 * LIE - 24, HANG);
    printf("作者:          R  ");
}
```

说明：初始化游戏中的那个界面，在某坐标打印字符的同时标记其相应状态，为的是蛇每一步移动的判断该发生什么反应。

定义函数 InitSnake()

```
void InitSnake()
{
    head.lenth = 2; //蛇身长度的初始化
    head.x = LIE / 2; //蛇头位置的横坐标的初始化
```

```

head.y = HANG / 2; //蛇头位置的纵坐标的初始化

body[0].x = head.x - 1; //初始的那2节蛇身坐标的初始化
body[0].y = head.y;
body[1].x = head.x - 2;
body[1].y = head.y;

maps[head.y][head.x] = HEAD;
maps[body[0].y][body[0].x] = BODY;
maps[body[1].y][body[1].x] = BODY;
}

```

说明：初始化蛇头坐标、蛇身长度、开始时每节蛇身的坐标。同理，在某坐标打印字符的同时标记其相应状态。

定义函数 InitFood()

```

void InitFood()
{
    int i, j;
    do
    {
        i = rand() % HANG; //随机生成食物的横纵坐标
        j = rand() % LIE;

    } while (maps[i][j] != AIR); //判断生成食物的位置是否为空，只有空位置才能生成
    maps[i][j] = FOOD;
    color(116); //打印食物字符
    JumpTo(2 * j, i);
    printf("★");
}

```

说明：初始化食物坐标。但要注意，判断生成食物的位置是否为空，只有空位置才能生成食物。同理，在某坐标打印字符的同时标记其相应状态。

定义函数 PrintSnake(int)

```

void PrintSnake(int flag)
{
    if (flag == 1) //打印新蛇头蛇
    {
        color(114);
        JumpTo(2 * head.x, head.y);
        printf("◆");
        for (int i = 0; i < head.lenth; i++)
        {
            JumpTo(2 * body[i].x, body[i].y);
            printf("◆");
        }
    }
}

```

```

    }
    else //打印空格覆盖原蛇身最后一节
    {
        if (body[head.lenth - 1].x != 0)
        {
            JumpTo(2 * body[head.lenth - 1].x, body[head.lenth - 1].y);
            printf(" ");
        }
    }
}

```

说明：分两块，由一个变量 **flag** 控制开关。分别是打印蛇头和打印两个空格以让蛇身最后一节消失。

定义函数 AutoMove(int,int)

```

void AutoMove(int x, int y)
{
    while (1)
    {
        int flag = 0;
        int time = speed;
        //当time自减为0时跳出while, 所以time越小, 蛇移动速度越快。因此speed越小, 蛇移动速度越快
        while (time--)
        {
            //调用头文件<conio.h>中kbhit()函数, 检查当前是否有键盘输入, 若有则返回一个非0值, 否则返回0
            if (kbhit() != 0)
            {
                flag = 1;
            }
        }
        if (flag == 0)
        {
            AfterGame(x, y); //先判断到达该位置后, 是否得分与游戏结束
            Move(x, y); //再移动蛇
        }
        if (flag == 1)
        {
            break; //通过break跳出while(1), 这里返回的是函数Game()读取这个键值
        }
    }
}

```

说明：含参函数。在函数 **Game()** 中将按不同的方向键产生不同的参数，传递给函数 **AutoMove(int,int)**，并由函数 **AutoMove(int,int)** 判断“蛇”是否可能需要改变移动方向。在此函数中，先设置一个死循环，来实现蛇的自动移动功能，每循环一次为一个时间周期，通过再装入一个自变量自减的循环结构，改变其中自变量

的数值大小即可控制循环周期的长度，进而实现“蛇”移动速度的改变，再进而实现游戏难度的调整。若在此循环周期内有新的来自键盘的输入，则跳出循环，认定“蛇”可能需要改变移动方向，并交给函数 `Move(int,int)` 处理按键效果。

定义函数 `Move(int,int)`

```
void Move(int x, int y)
{
    PrintSnake(0); //先消除当前所显示蛇的蛇身最后一节
    maps[body[head.lenth - 1].y][body[head.lenth - 1].x] = AIR; //蛇移动后蛇尾重新标记为空
    maps[head.y][head.x] = BODY; //蛇移动后蛇头的位置变为蛇身第一节
    //重新修改各个坐标
    for (int i = head.lenth - 1; i > 0; i--)
    {
        body[i].x = body[i - 1].x;
        body[i].y = body[i - 1].y;
    }
    body[0].x = head.x;
    body[0].y = head.y;
    head.x = head.x + x;
    head.y = head.y + y;
    PrintSnake(1); //再打印出新的蛇头
}
```

说明：`x` 表示蛇移动后的横坐标相对于当前蛇的横坐标的变化。`y` 表示蛇移动后的纵坐标相对于当前蛇的纵坐标的变化。并结合函数 `AutoMove(int,int)`，所以 `Move(0, -1)` 表示新“蛇”头横坐标+0，纵坐标-1，所以是向上移动；`Move(0, 1)` 表示新“蛇”头横坐标+0，纵坐标+1，所以是向下移动；`Move(-1, 0)` 表示新“蛇”头横坐标-1，纵坐标+0，所以是向左移动；`Move(1, 0)` 表示新“蛇”头横坐标+1，纵坐标+0，所以是向右移动。

定义函数 `BeforeGame()`

```
void BeforeGame()
{
    //做出游戏前的界面
    color(112);
    for(int i = 0; i < HANG + 1; i++)
        for(int j = 0; j < LIE; j++)
            printf(" ");
    JumpTo(LIE - 9, 6);
    printf("贪吃蛇游戏");
    JumpTo(LIE - 8, HANG - 18);
    printf("游戏操作");
    JumpTo(LIE - 26, HANG - 16);
    printf("吃一颗星得1分 蛇头撞墙或撞到身体则游戏结束");
}
```

```

JumpTo(LIE - 26 , HANG - 14);

printf("方向键↑↓↔控制蛇的移动 空格键 SPACE 暂停游戏");

JumpTo(LIE - 24 , HANG - 12);

printf("请按以下数字键开始游戏...退出游戏请按 ESC 键...");

JumpTo(LIE - 12 , HANG - 10);

printf("1——新手模式");

JumpTo(LIE - 12 , HANG - 8);

printf("2——老手模式");

JumpTo(LIE - 12 , HANG - 6);

printf("3—— 模式");

JumpTo(2 * LIE - 24 , HANG - 1);

printf("作者: ");

//实现游戏前的界面的功能

int n;

while(1)

{
    n = getch();

    if (n == ESC)

    {

        system("cls");

        JumpTo(LIE - 6 , HANG / 2);

        printf("游 戏 结 束 ! \a");

        JumpTo(0, HANG - 1);

        exit(0);

    }

    else if (n == '1' || n == '2' || n == '3' )

    {

        switch(n) //选择蛇的移动速度

        {

            case '1': speed = 4096; break;

            case '2': speed = 2048; break;

            case '3': speed = 1024; break;

        }

        system("cls");

        ReadGrade(); //从文件读取最高分

        InitMap(); //初始化地图

        InitSnake(); //初始化蛇

        PrintSnake(1);

        InitFood(); //初始化食物

        Game(); //开始游戏;

    }

    else

    {

        printf("\a"); //响一声铃提醒玩家

```

```

    }
}
}

```

说明：分两块。先打印出游戏前选择难度的那个界面。再读取键值实现难度选择等等功能。

定义函数 Game()

```

void Game()
{
    int n; //初始伪键值 (n)
    int tmp = RIGHT; //初始移动方向 (tmp)
    AutoMove(1, 0); //游戏刚开始时蛇向右跑
    while (1)
    {
        n = getch();
        //游戏中防止其他键捣乱
        if(n != UP && n != DOWN && n != RIGHT && n != LEFT && n != SPACE && n != ESC)
        {
            n = tmp;
        }
        switch (n)
        {
            case UP: //当按了向上键时
                if (tmp != DOWN) //蛇正在移动的方向是向下时不能改变蛇的方向，即防止蛇出现“原地调头”
                {
                    AutoMove(0, -1); //向上移动
                    tmp = UP; //记录当前蛇的移动方向
                }
                else
                {
                    n = tmp; //按键“无效”，还是向下移动
                    AutoMove(0, 1);
                }
                break;
            case DOWN:
                if (tmp != UP)
                {
                    AutoMove(0, 1);
                    tmp = DOWN;
                }
                else
                {
                    n = tmp;
                    AutoMove(0, -1);
                }
            // ... (other cases for LEFT and RIGHT)
        }
    }
}

```

```

    }
    break;
case LEFT:
    if (tmp != RIGHT)
    {
        AutoMove(-1, 0);
        tmp = LEFT;
    }
    else
    {
        n = tmp;
        AutoMove(1, 0);
    }
    break;
case RIGHT:
    if (tmp != LEFT)
    {
        AutoMove(1, 0);
        tmp = RIGHT;
    }
    else
    {
        n = tmp;
        AutoMove(-1, 0);
    }
    break;
case SPACE:
    system("pause>nul"); //暂停后按任意键继续
    break;
case ESC:
    system("cls"); //清空屏幕并生成游戏结束画面
    color(112);
    JumpTo(LIE - 6, HANG / 2);
    printf("游戏结束!\a");
    JumpTo(0, HANG - 1);
    exit(0);
}
}
}

```

说明：用来实现在游戏中通过读取用户按键的键值进行各种操作，如改变“蛇”的移动方向、暂停、退出游戏。先初始化蛇的移动方向，这里设置先向右移动比较合理。随后读取键值，为防止其他按键捣乱产生不必要的程序漏洞，所以要判断是否是有效键值，并进入一个 **switch** 语句，根据不同的键值实现相应的功能。用户输入方向键“↑、↓、←、→”时，实现“蛇”改变移动方向的功能。这里

一定要考虑到，“蛇”不能出现“原地调头”的现象发生，其逻辑就是：不能在“蛇”朝一个方向移动时，按相反的方向键并执行其效果，因此需要再定义一个变量 tmp，来记录“蛇”当前移动的方向。

定义函数 AfterGame(int,int)

```
void AfterGame(int x, int y)
{
    //若蛇头即将到达的位置是食物，则：蛇身长度加一，得分加一，随机生成一个食物
    if (maps[head.y + y][head.x + x] == FOOD)
    {
        head.lenth++; //蛇身长度加一
        grade += 1; //得分加一并实时更新
        color(112);
        JumpTo(2, HANG);
        printf("当前得分:%d分", grade);
        InitFood(); //随机生成食物
    }

    //若蛇头即将到达的位置是墙或者蛇身，则游戏结束
    else if (maps[head.y + y][head.x + x] == WALL || maps[head.y + y][head.x + x] == BODY)
    {
        system("cls");
        color(112);
        JumpTo(LIE - 6, 6);
        printf("游戏结束!\a"); //响一声铃提醒玩家
        JumpTo(2 * LIE - 24, HANG - 1);
        printf("作者:");
        JumpTo(LIE - 18, HANG / 2);
        if (grade > best)
        {
            printf("您已经刷新了最高记录! 您的分数是%d分", grade);
            WriteGrade();
        }
        else if (grade == best)
        {
            printf("您已经追平了最高纪录! 您的分数是%d分", grade);
        }
        else
        {
            printf("您的分数是%d分! 还差%d分才能追平记录", grade, best - grade);
        }

        int n; //读取键值看是再来一局还是退出游戏
        while (1)
        {
            JumpTo(LIE - 20, HANG - 6);
```



```

printf("重新开始游戏请按 ENTER 键...退出游戏请按 ESC 键...");

n = getch();
if (n == ESC)
{
    JumpTo(0, HANG - 1);
    exit(0);
}
else if (n == ENTER)
{
    system("cls");
    main();
}
else
{
    printf("\a"); //响一声铃提醒玩家
}
}
}

```

说明：分两块：一是蛇头即将到达的位置是食物，二是蛇头即将到达的位置是墙或蛇身。若“蛇头”即将移动到的坐标的状态为“食物”，则“蛇身长度”加一。若“蛇头”即将移动到的坐标的状态为“墙”或“蛇身”，则执行清屏，并输出显示游戏结束界面，在游戏的结束界面，会显示用户本局游戏的最终得分和于历史最高得分的比较，并让用户按键选择，是重新开始游戏还是退出游戏。

定义函数 ReadGrade()

```

void ReadGrade()
{
    FILE* pf = fopen("TheSnake_HighLight_ByR .txt", "r");
    if (pf == NULL)
    {
        pf = fopen("TheSnake_HighLight_ByR .txt", "w");
        fwrite(&best, sizeof(int), 1, pf);
    }
    fseek(pf, 0, SEEK_SET);
    fread(&best, sizeof(int), 1, pf);
    fclose(pf);
    pf = NULL;
}

```

说明：用来读取文件 TheSnake_HighLight_ByR txt 中记录的变量的数值，来实现历史最高得分的保存。若当前文件夹中没有这个文件，则创建该文件。属于锦上添花，与游戏主题逻辑关系其实不大。

定义函数 WriteGrade()

```
void WriteGrade()
{
    FILE* pf = fopen("TheSnake_HighLight_ByR .txt", "w");
    if (pf == NULL)
    {
        printf("\a");
        exit(0);
    }
    fwrite(&grade, sizeof(int), 1, pf);
    fclose(pf);
    pf = NULL;
}
```

说明：用来将新的历史最高记录赋值给文件 TheSnake_HighLight_ByR .txt 中对应变量，从而实现最高得分记录值的更新。属于锦上添花，与游戏主题逻辑关系其实不大。

四 运行结果

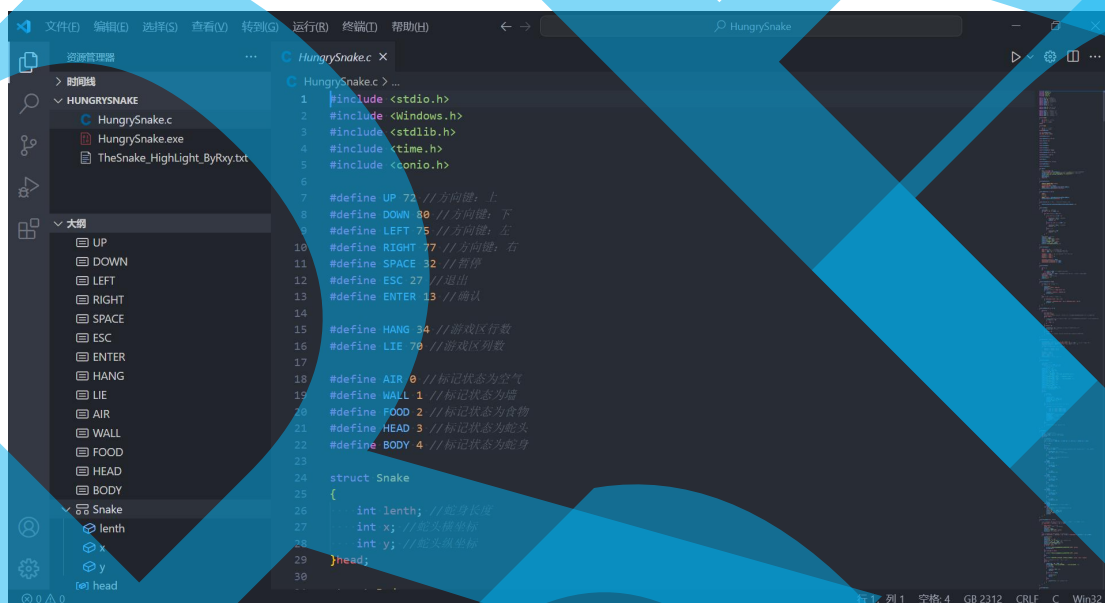


图 1.1 打开开发工具（Visual Studio Code）的界面(1)

图 1.2 打开开发工具（Visual Studio Code）的界面(2)

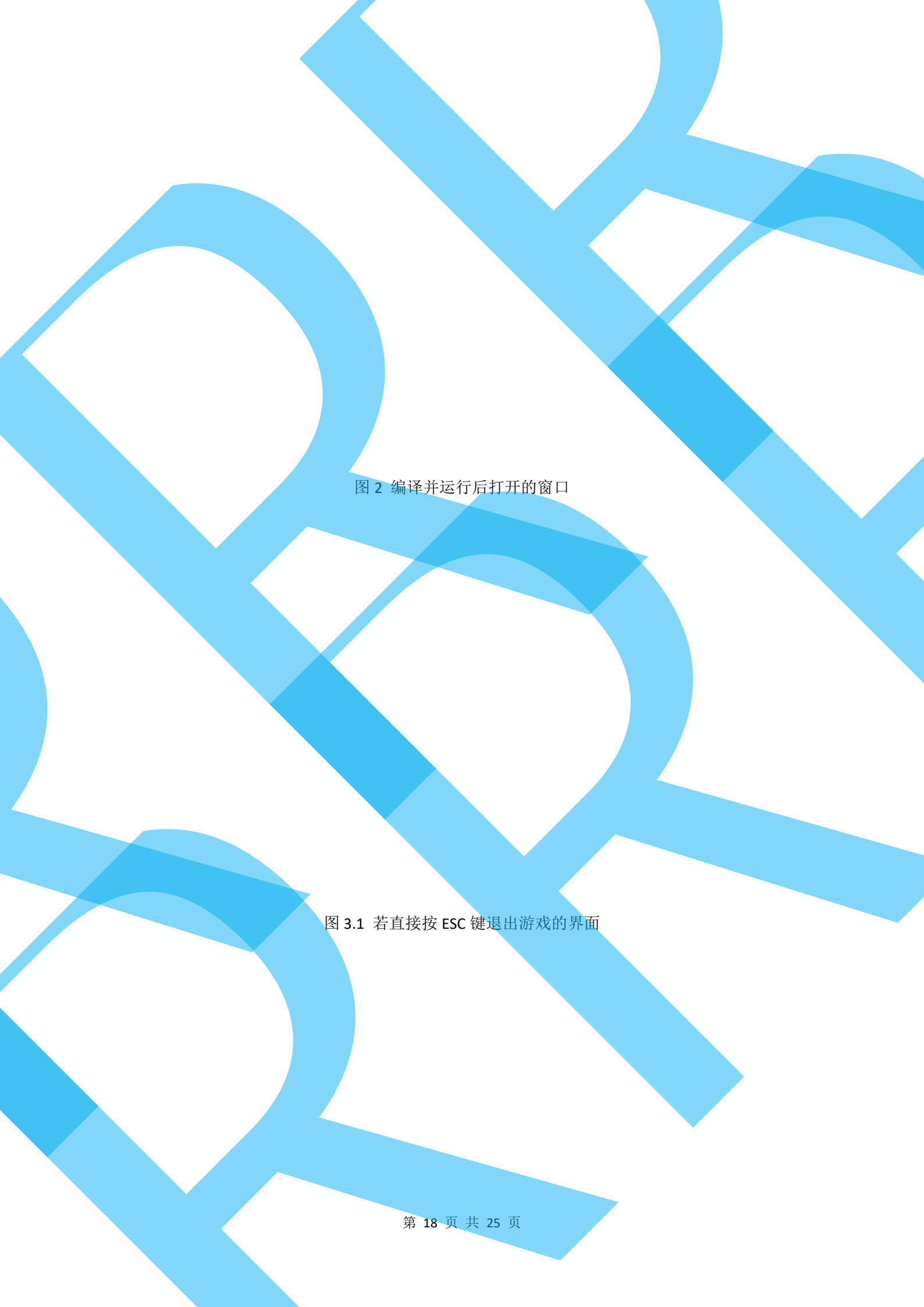


图 2 编译并运行后打开的窗口

图 3.1 若直接按 ESC 键退出游戏的界面

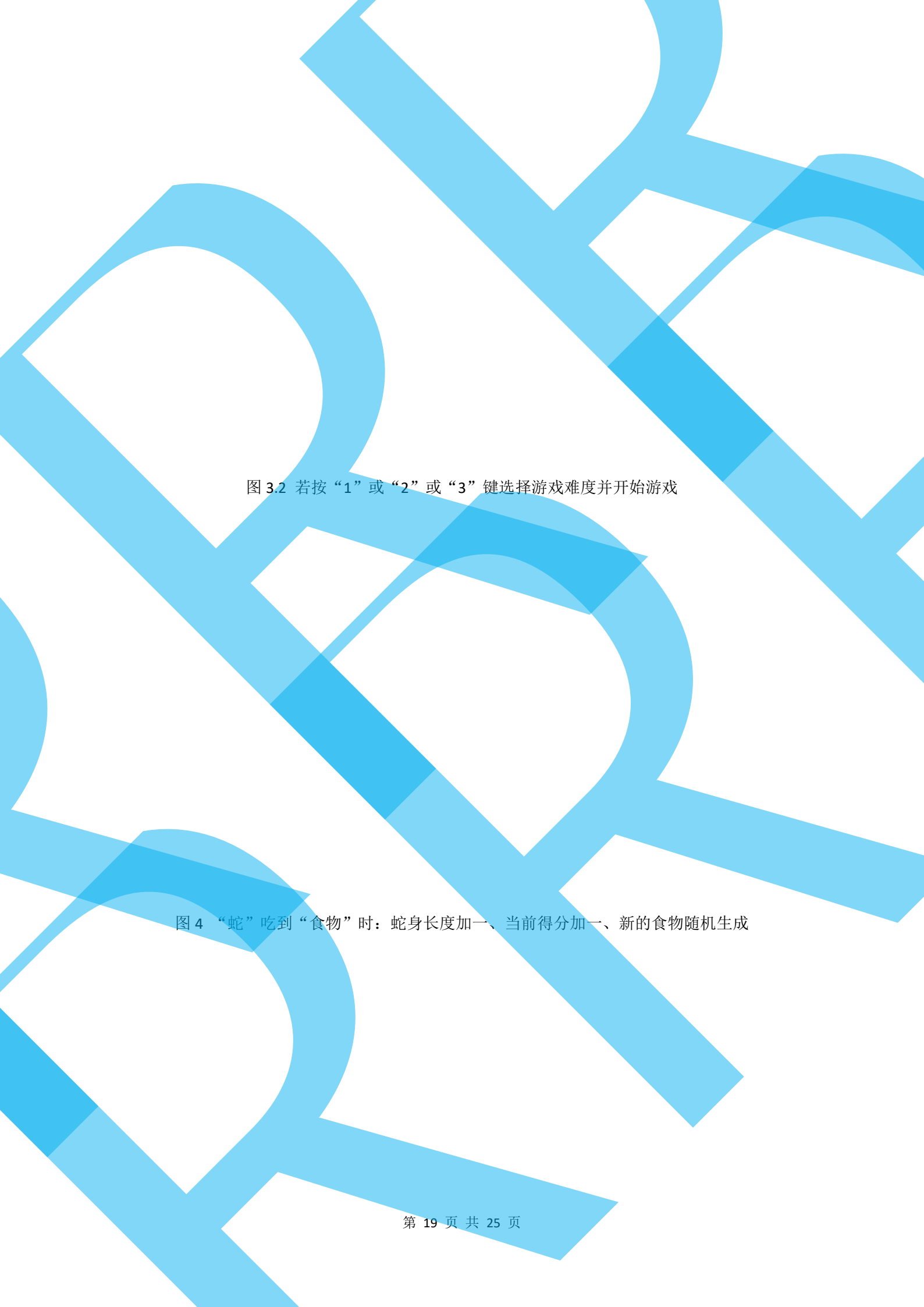
The background of the page is a complex, abstract pattern of overlapping, semi-transparent blue shapes. These shapes include various geometric forms like circles, polygons, and thick, curved lines that create a sense of depth and movement. The colors range from a light sky blue to a deeper cerulean blue.

图 3.2 若按“1”或“2”或“3”键选择游戏难度并开始游戏

图 4 “蛇”吃到“食物”时：蛇身长度加一、当前得分加一、新的食物随机生成

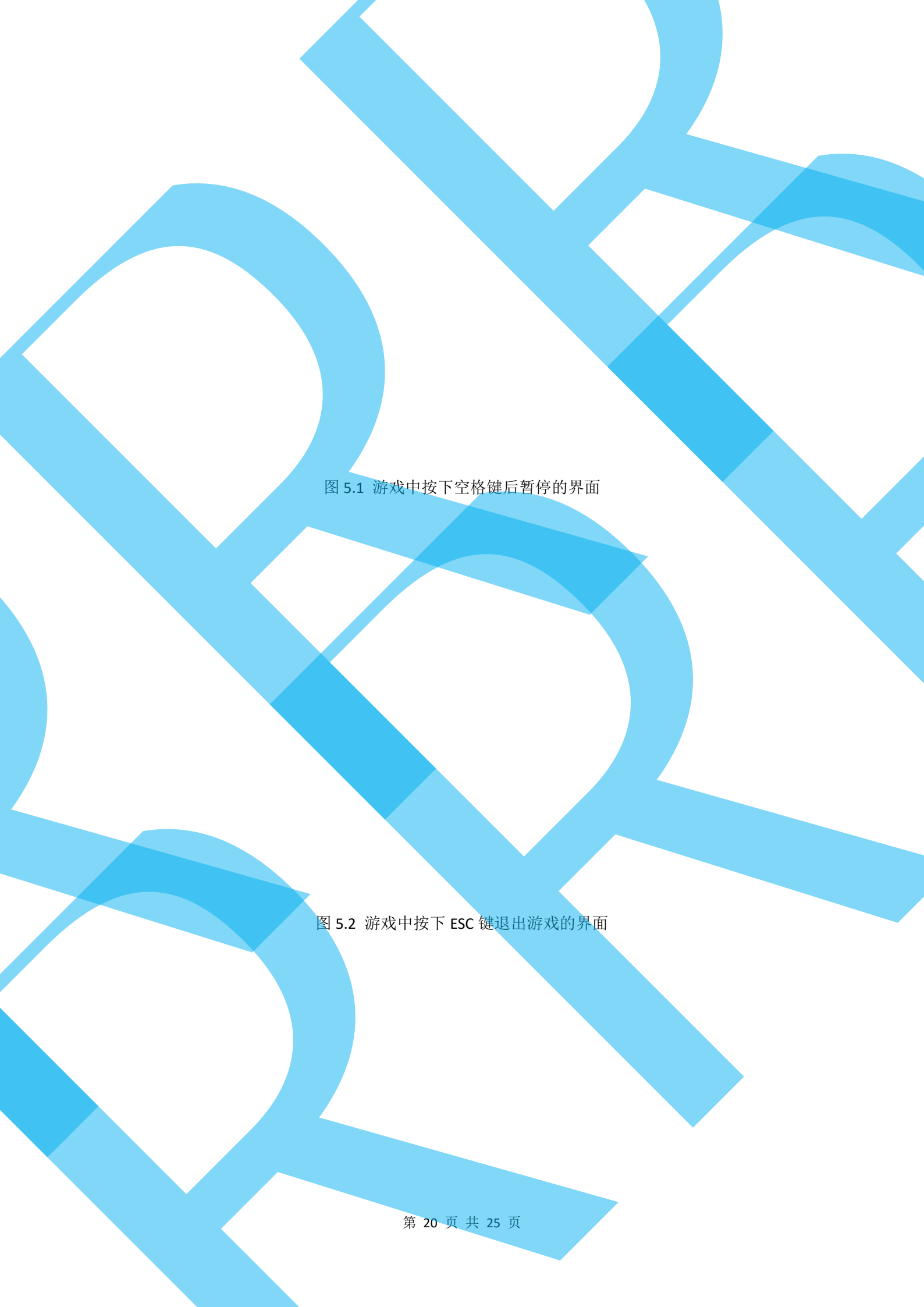
The background of the entire page is a complex, abstract pattern of overlapping, semi-transparent blue shapes. These shapes include thick lines, curves, and polygons that create a sense of depth and movement. The colors range from a light sky blue to a deeper cerulean blue.

图 5.1 游戏中按下空格键后暂停的界面

图 5.2 游戏中按下 ESC 键退出游戏的界面

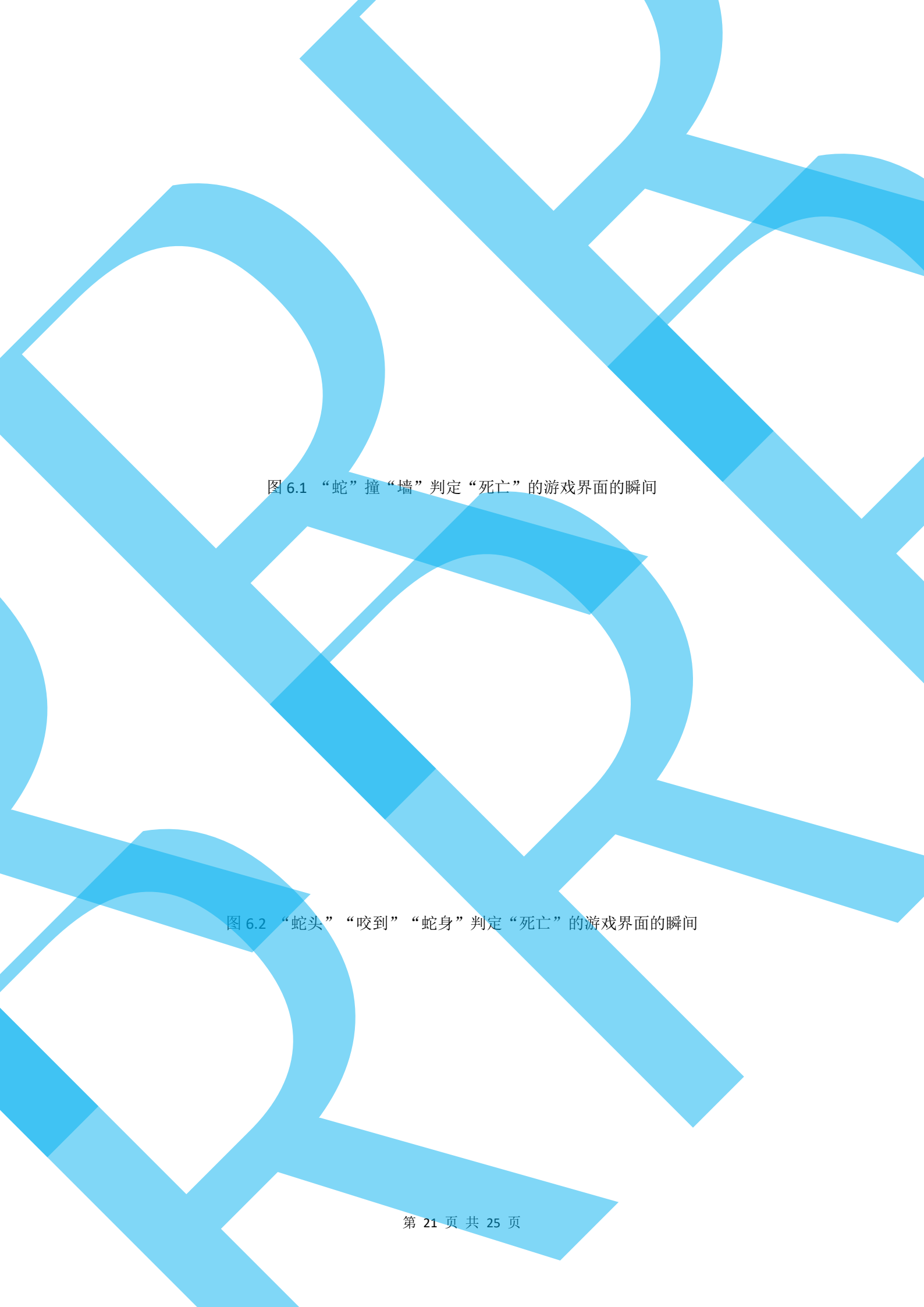
The background of the entire page is a complex, abstract pattern of overlapping, semi-transparent blue shapes. These shapes include various polygons, circles, and curved lines, creating a dynamic and layered visual effect. The colors range from a light sky blue to a deeper cerulean blue.

图 6.1 “蛇”撞“墙”判定“死亡”的游戏界面的瞬间

图 6.2 “蛇头”“咬到”“蛇身”判定“死亡”的游戏界面的瞬间

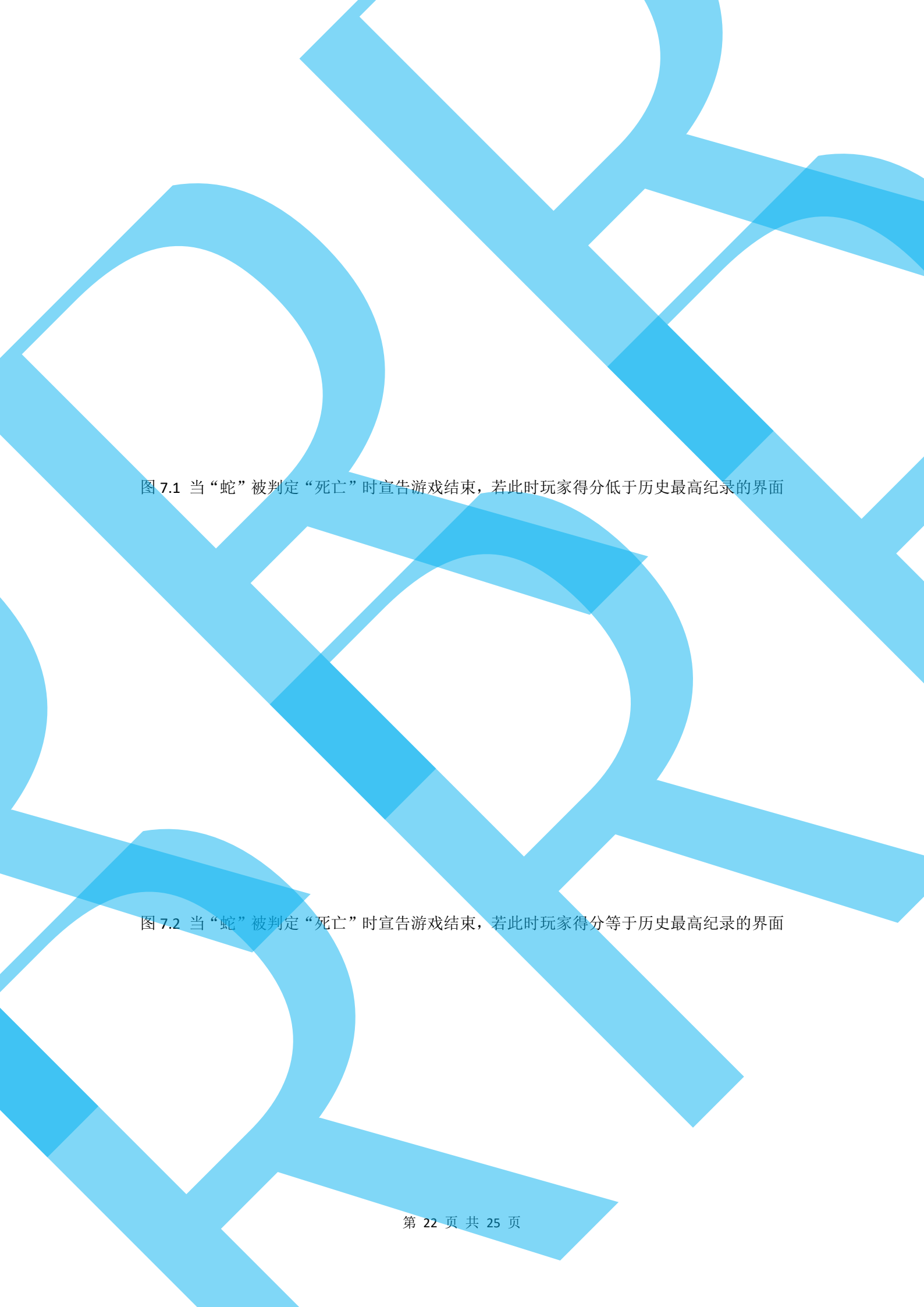
The background of the entire page is a complex, abstract pattern of overlapping, semi-transparent blue shapes. These shapes include various geometric forms like circles, polygons, and thick, curved lines that intersect to create a dynamic, layered visual effect. The shades of blue range from a light, airy cyan to a more saturated, vibrant blue.

图 7.1 当“蛇”被判定“死亡”时宣告游戏结束，若此时玩家得分低于历史最高纪录的界面

图 7.2 当“蛇”被判定“死亡”时宣告游戏结束，若此时玩家得分等于历史最高纪录的界面

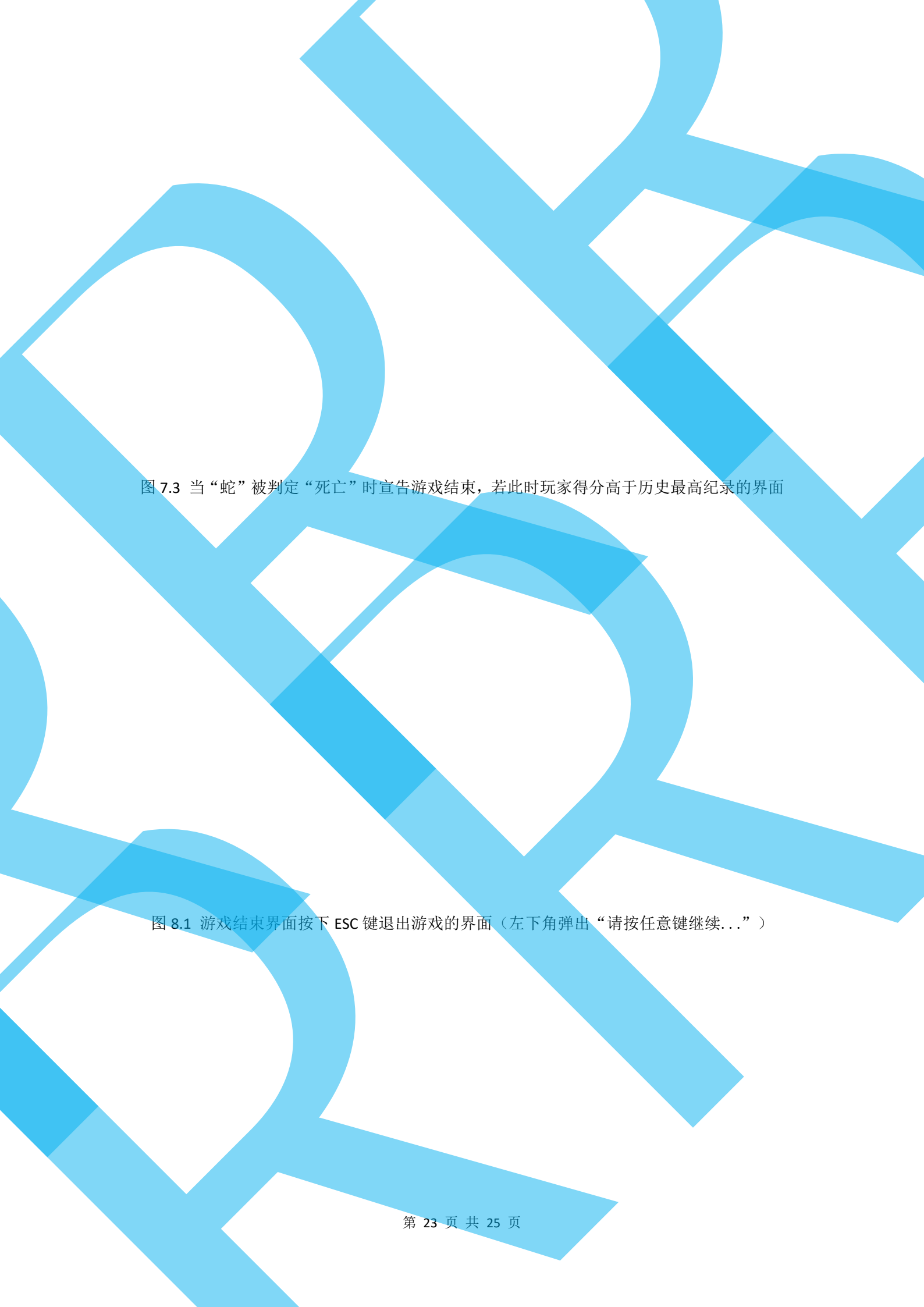
The background of the entire page is a complex, abstract pattern of overlapping, semi-transparent blue shapes. These shapes include various geometric forms like circles, polygons, and lines, creating a dynamic and modern visual texture.

图 7.3 当“蛇”被判定“死亡”时宣告游戏结束，若此时玩家得分高于历史最高纪录的界面

图 8.1 游戏结束界面按下 ESC 键退出游戏的界面（左下角弹出“请按任意键继续...”）

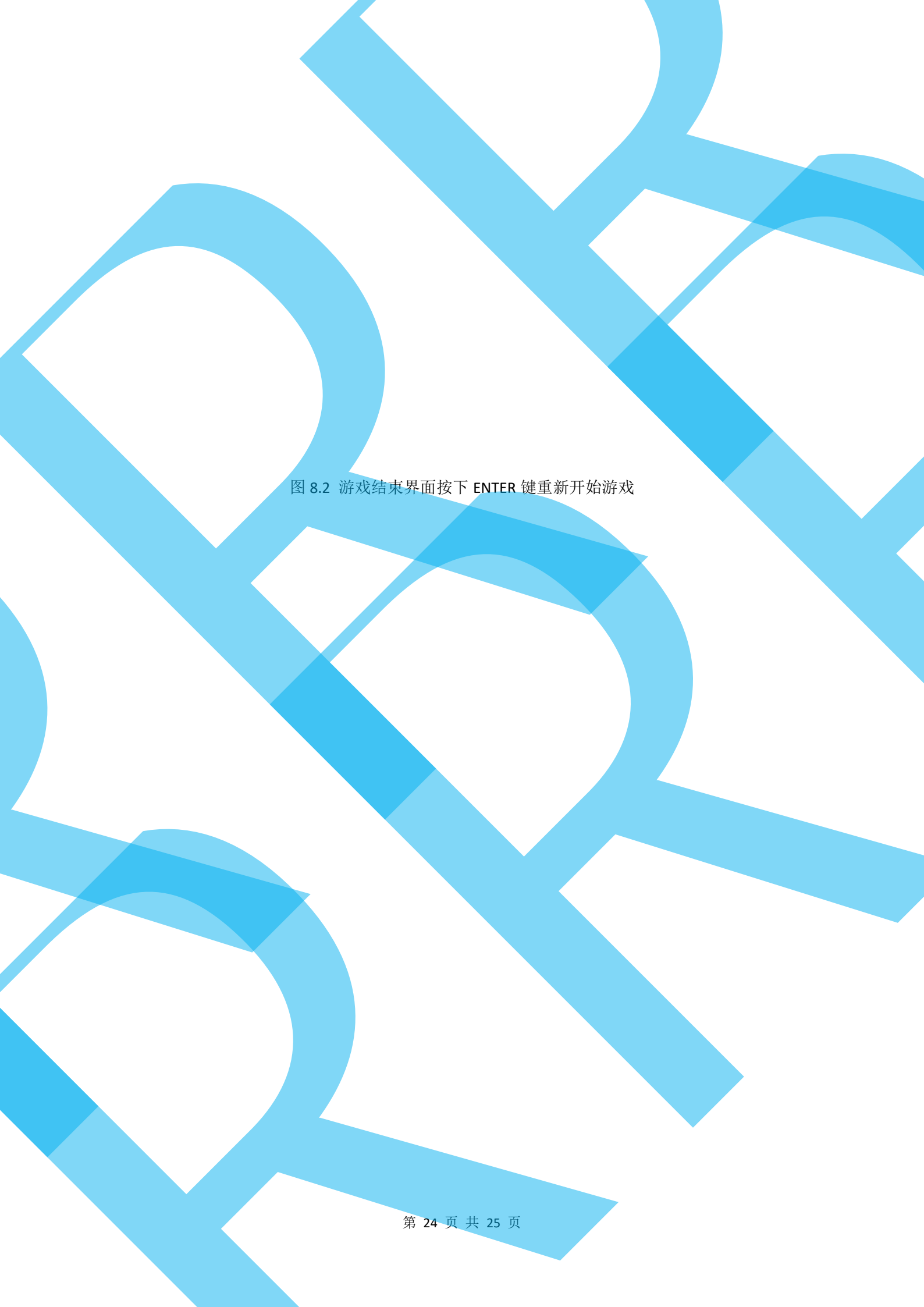
The background of the page is a complex, abstract pattern of overlapping, semi-transparent blue lines and shapes. These shapes resemble stylized, thick strokes of a brush or a series of interconnected geometric forms, creating a sense of movement and depth. The colors range from a light sky blue to a deeper cerulean blue.

图 8.2 游戏结束界面按下 ENTER 键重新开始游戏

五 心得体会