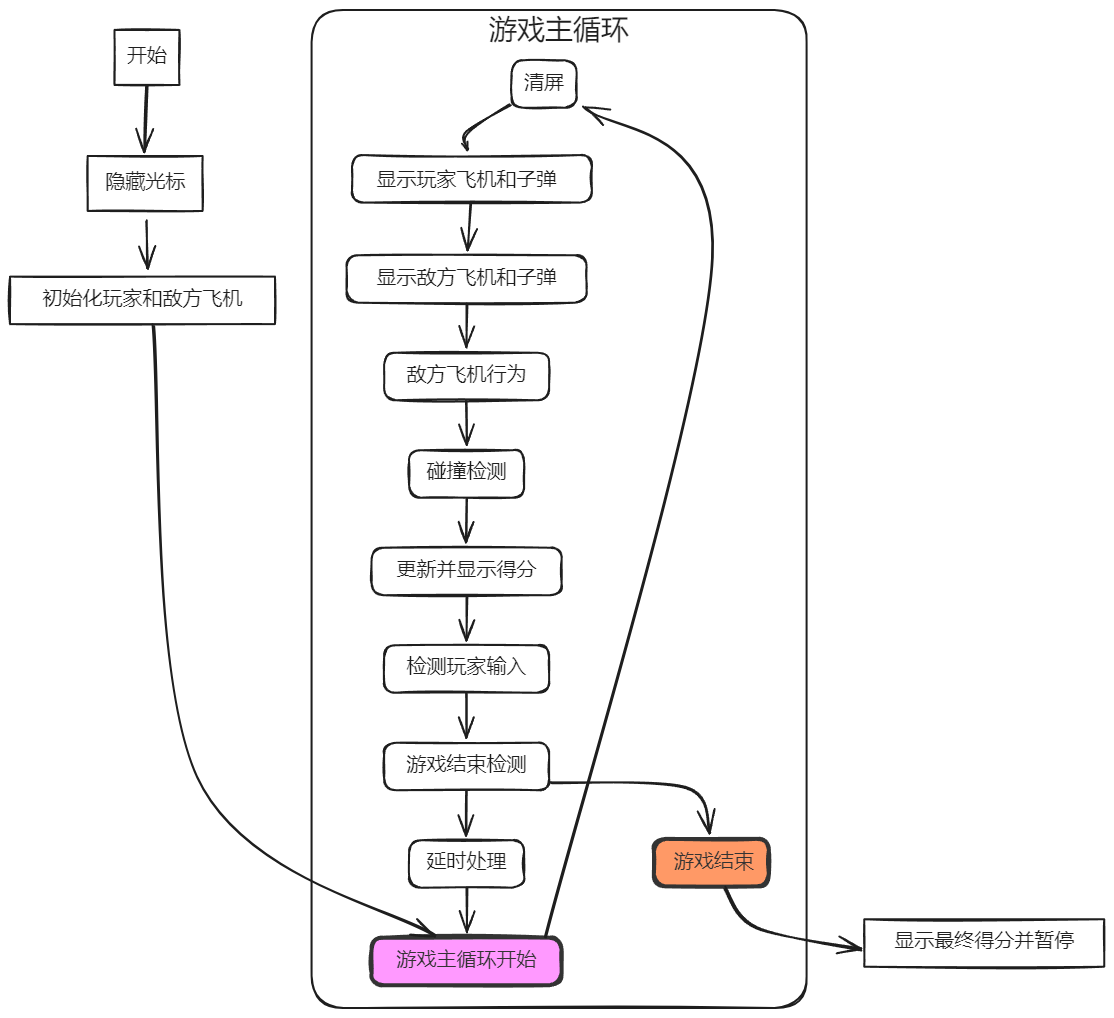
飞机大战的游戏设计

**设计思想：**

1. **玩家控制的飞机**：玩家可以通过键盘控制自己的飞机移动和射击。
2. **敌方飞机**：敌方飞机会自动生成并随机移动，玩家需要射击它们来获得分数。
3. **子弹机制**：玩家和敌方飞机都可以发射子弹，子弹可以击中飞机导致得分变化。
4. **碰撞检测**：通过检查矩形是否重叠来检测飞机与飞机、子弹与飞机之间的碰撞。
5. **游戏结束条件**：玩家分数为零或与敌方飞机碰撞会导致游戏结束。

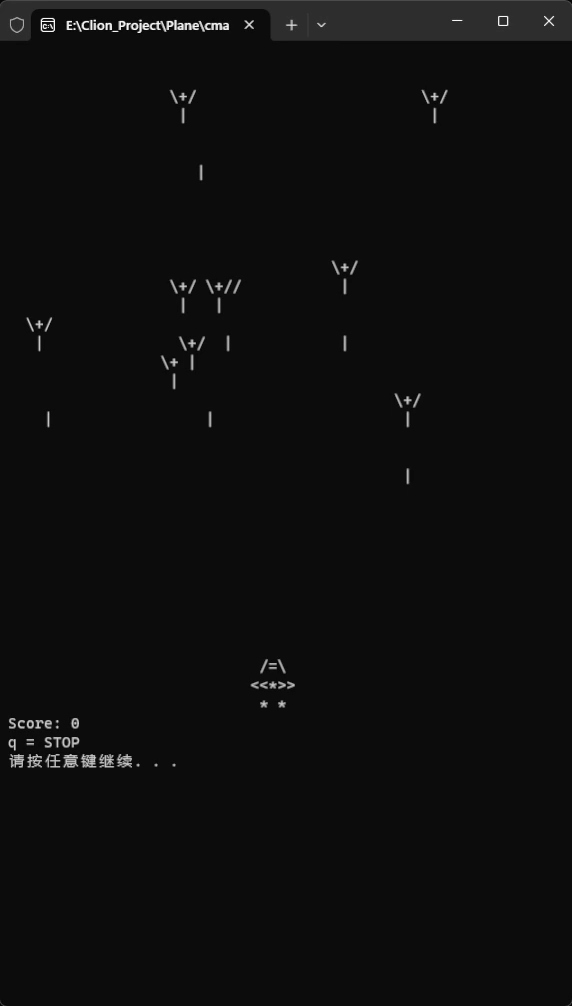
**流程图：**

****

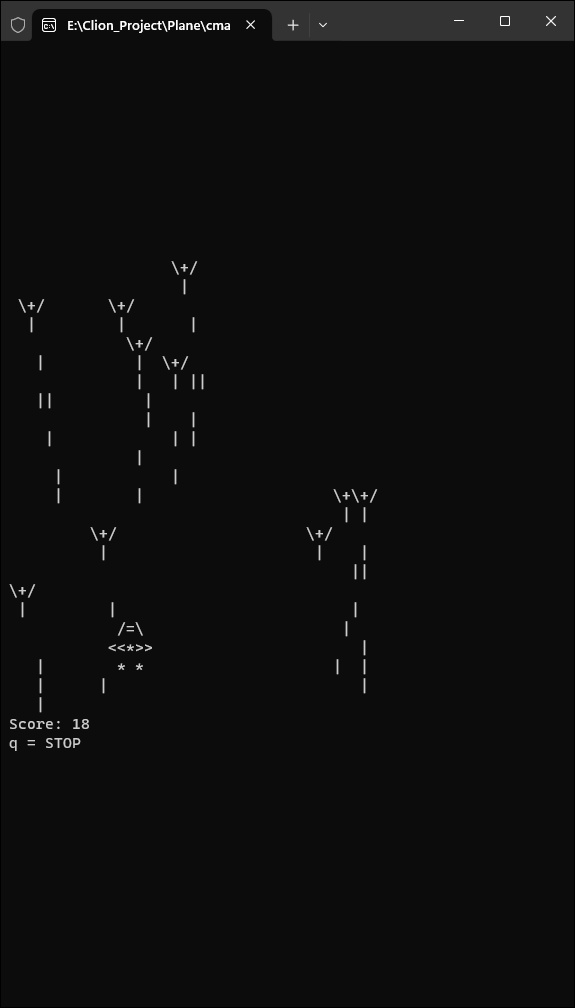
**代码分析（注释）：**

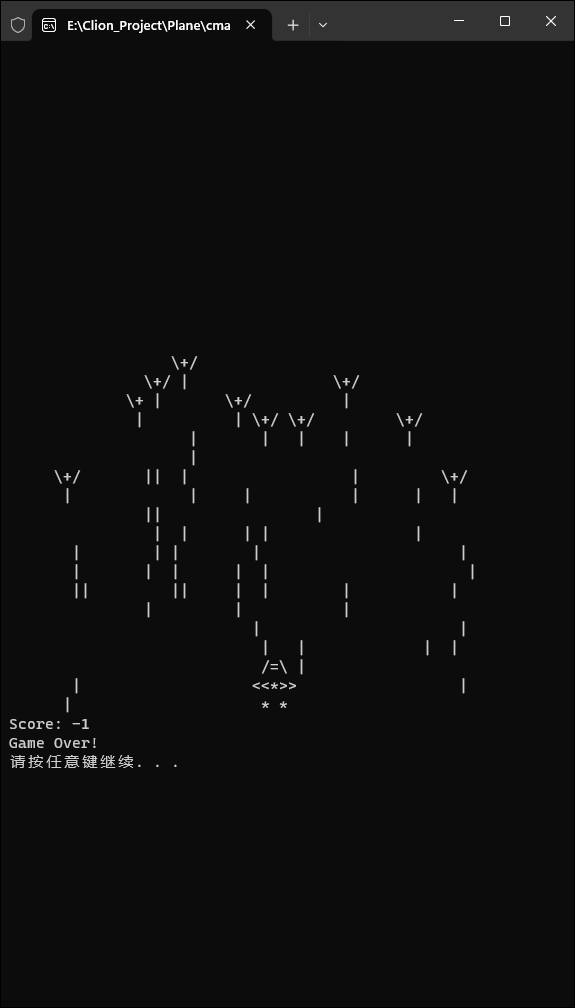
|  |
| --- |
| *//main.cpp*  #include *<iostream>*  #include *<windows.h>*  #include *<vector>*  #include *<ctime>*  #include *<cstdlib>*  #include *<algorithm>*  #include *"Plane.h"*  **using** **namespace** **std**;  *// 定义我方飞机的形状*  **const** char\* myPlaneShape[3] = {  " /=**\\** ",  "<<\*>>",  " \* \* "  };  *// 定义敌方飞机的形状*  **const** char\* enemyPlaneShape[2] = {  "**\\**+/",  " | ",  };  *// 控制台光标移动函数*  void gotoXY(int x, int y) {  COORD pos = {short(x), short(y)};  HANDLE hOut = GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleCursorPosition(hOut, pos);  }  *// 隐藏光标*  void HideCursor() {  CONSOLE\_CURSOR\_INFO cursor\_info = {1, 0};  SetConsoleCursorInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &cursor\_info);  }  *// 清屏函数*  void clearScreen() {  system("cls");  }  *// 显示飞机*  void displayPlane(**const** Plane& plane, **const** char\* shape[3]) {  **for** (int i = 0; i < 3; ++i) {  gotoXY(plane.pos.x, plane.pos.y + i);  cout << shape[i];  }  }  *// 显示子弹*  void displayBullets(**const** vector<Bullet>& bullets) {  **for** (**const** **auto**& bullet : bullets) {  gotoXY(bullet.pos.x, bullet.pos.y);  cout << '|';  }  }  *// 检查两个矩形是否重叠*  bool isCollision(Position p1, int w1, int h1, Position p2, int w2, int h2) {  **return** p1.x < p2.x + w2 &&  p1.x + w1 > p2.x &&  p1.y < p2.y + h2 &&  p1.y + h1 > p2.y;  }  *// 游戏主函数*  int main() {  srand(time(0));  HideCursor();  Plane myPlane(WIDTH / 2, HEIGHT - 3);  vector<Plane> enemyPlanes;  int score = 0;  **const** int initialEnemyCount = 10;  **const** int enemyBulletSpeed = 2; *// 控制敌方子弹速度*  int i = 0 , speed = 4;  *// 创建敌方飞机*  **for** (int i = 0; i < initialEnemyCount; ++i) {  int x = rand() % (WIDTH - 5);  int y = rand() % (HEIGHT / 2);  enemyPlanes.emplace\_back(x, y);  }  **while** (true) {  clearScreen();  i = (i+1) % 5;  *// 显示我方飞机和子弹*  displayPlane(myPlane, myPlaneShape);  myPlane.moveBullets();  displayBullets(myPlane.bullets);  *// 显示敌方飞机和子弹*  **for** (**auto**& enemy : enemyPlanes) {  *// 敌方飞机自动向前（向下）移动*  enemy.move(0, 1.0/speed);  *// 随机左右移动*  double dx = ((rand() % 3) - 1.0)/speed;  enemy.move(dx, 0);  *// 敌方飞机发射子弹*  **if** (rand() % 10 == 0 ) {  enemy.shoot(true);  }  displayPlane(enemy, enemyPlaneShape);  enemy.moveBullets(true, enemyBulletSpeed);  displayBullets(enemy.bullets);  *// 如果敌方飞机到达最底部，则重新生成位置*  **if** (enemy.pos.y >= HEIGHT - 3) {  enemy.pos.y = rand() % (HEIGHT / 2);  enemy.pos.x = rand() % (WIDTH - 5);  }  }  *// 检查我方飞机和敌方飞机之间的碰撞*  **for** (**const** **auto**& enemy : enemyPlanes) {  **if** (isCollision(myPlane.pos, 5, 3, enemy.pos, 3, 2)) {  gotoXY(0, HEIGHT);  cout << "Score: " << score << endl;  cout << "Game Over!" << endl;  system("pause");  **return** 0;  }  }  *// 检查我方子弹与敌方飞机之间的碰撞*  **for** (**auto** it = myPlane.bullets.begin(); it != myPlane.bullets.end();) {  bool hit = false;  **for** (**auto** et = enemyPlanes.begin(); et != enemyPlanes.end();) {  **if** (isCollision(it->pos, 1, 1, et->pos, 5, 3)) {  it = myPlane.bullets.erase(it);  enemyPlanes.erase(et);  score += 1;  hit = true;  *// 补充新的敌方飞机*  int x = rand() % (WIDTH - 5);  int y = rand() % (HEIGHT / 2);  enemyPlanes.emplace\_back(x, y);  **break**;  } **else** {  ++et;  }  }  **if** (!hit) {  ++it;  }  }  *// 检查敌方子弹与我方飞机之间的碰撞*  **for** (**auto**& enemy : enemyPlanes) {  **for** (**auto** it = enemy.bullets.begin(); it != enemy.bullets.end();) {  **if** (isCollision(it->pos, 1, 1, myPlane.pos, 5, 3)) {  it = enemy.bullets.erase(it);  score -= 1;  **if** (score < 0) {  gotoXY(0, HEIGHT);  cout << "Score: " << score << endl;  cout << "Game Over!" << endl;  system("pause");  **return** 0;  }  } **else** {  ++it;  }  }  }  *// 显示得分*  gotoXY(0, HEIGHT);  cout << "Score: " << score << endl <<"q = STOP" << endl;  *// 控制我方飞机移动*  **if** (GetAsyncKeyState('W') & 0x8000) myPlane.move(0, -1);  **if** (GetAsyncKeyState('S') & 0x8000) myPlane.move(0, 1);  **if** (GetAsyncKeyState('A') & 0x8000) myPlane.move(-1, 0);  **if** (GetAsyncKeyState('D') & 0x8000) myPlane.move(1, 0);  **if** (GetAsyncKeyState(VK\_SPACE) & 0x8000) myPlane.shoot(false);  **if** (GetAsyncKeyState('Q') & 0x8000) {  system("pause");  }  Sleep(30); *// 减少延迟，提升流畅度*  }  }  *//plane.h*  #ifndef PLANE\_PLANE\_H  #define PLANE\_PLANE\_H  #include *<vector>*  #include *<algorithm>*  **using** **namespace** **std**;  *// 定义战场的大小*  **const** int HEIGHT = 35;  **const** int WIDTH = 55;  *// 飞机位置结构*  **struct** **Position** {  double x, y;  };  *// 子弹类*  **class** **Bullet** {  **public**:  Position pos;  int speedCounter;  Bullet(int x, int y) : pos{x, y}, speedCounter(0) {}  };  *// 飞机类*  **class** **Plane** {  **public**:  Position pos;  vector<Bullet> bullets;  Plane(double x, double y) : pos{x, y} {}  void move(double dx, double dy);  void shoot(bool isEnemy = false);  void moveBullets(bool isEnemy = false, int speed = 1);  };  #endif *//PLANE\_PLANE\_H*  *//plane.cpp*  #include *"Plane.h"*  void Plane::move(double dx, double dy) {  pos.x += dx;  pos.y += dy;  **if** (pos.x < 0) pos.x = 0;  **if** (pos.x >= WIDTH - 5) pos.x = WIDTH - 5;  **if** (pos.y < 0) pos.y = 0;  **if** (pos.y >= HEIGHT - 3) pos.y = HEIGHT - 3;  }  void Plane::shoot(bool isEnemy) {  **if** (isEnemy) {  bullets.emplace\_back(pos.x + 2, pos.y + 3);  } **else** {  bullets.emplace\_back(pos.x + 2, pos.y - 1);  }  }  void Plane::moveBullets(bool isEnemy, int speed) {  **for** (**auto**& bullet : bullets) {  bullet.speedCounter++;  **if** (bullet.speedCounter >= speed) {  bullet.pos.y += (isEnemy ? 0.8 : -1);  bullet.speedCounter = 0;  }  }  bullets.erase(remove\_if(bullets.begin(), bullets.end(), [](Bullet b) { **return** b.pos.y < 0 || b.pos.y >= HEIGHT; }), bullets.end());  } |

**效果展示**

****

****

****

****