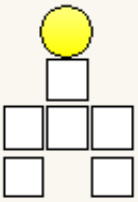

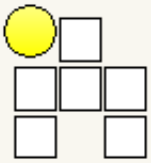

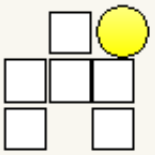

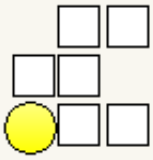

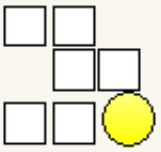

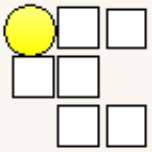

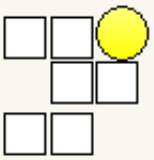

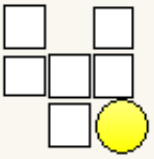

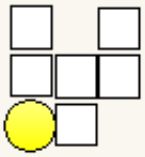

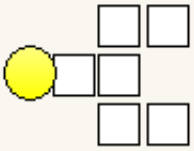

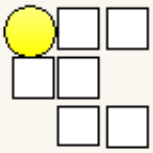

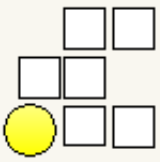

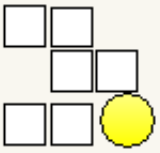

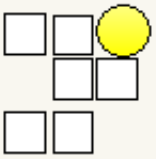

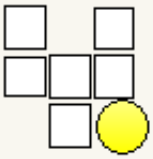

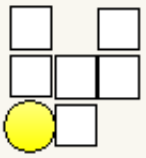

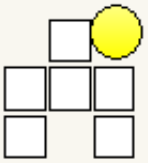

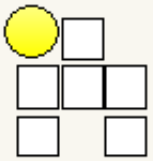

1、上面分别是下一次要改变的方向，是上、左的时候，原始状态的不同情况；右、下类似；
2、以移动后，不擦除原来的障碍物为第一出发点；
3、走到这种状态，是有前提的，这点很重要，也是能处理好的根本原因；不会凭空，走到某种状态；
4、符合行为规范；

注意：关于同类坦克情况，禁止互相跨越的时候，和这个障碍物的判断，还有所不同，这个判断，要完全按照9格处理，否则，会导致无法移动；

子弹和坦克：

```
typedef struct _BULLET {
    int m_bullet_nX;
    int m_bullet_nY;
    DIRECTION m_bullet_direction;
    int m_bullet_speed;
    bool m_bullet_disapper;
    struct _BULLET* m_bullet_next;
    struct _TANK* m_owner_tank;
} BULLET, *PBULLET;

typedef struct _TANK {
    /* Coordinate */
    int m_tank_nX;
    int m_tank_nY;

    /* Move Direction */
    DIRECTION m_tank_direction;

    DIRECTION m_go_direction;

    int m_change_direction_count;

    /* Move Speed */
    int m_tank_speed;

    /* Life circle */
    int m_tank_blood;

    /* Destroy one enemy, got score */
    int m_tank_score;
```

```
31
32      bool m_tank_enermy_flag;
33
34      bool m_is_tank_alive;
35
36      char m_color;
37
38      /* Differ Tank type */
39      TANK_TYPE m_tank_type;
40
41      /* Record bullet info, Not so sure, TBD... */
42      struct _BULLET* m_tank_bullet;
43  } TANK, *PTANK;
```

子弹组织成链表结构；

敌方坦克的移动和发射子弹，主要考虑在单线程下，如何判断——时间差；

另外，在控制坦克移动时，可以使用

```
1
2   #define KEY_DOWN(vk_code)    ((GetAsyncKeyState(vk_code) & 0x8000) ? 1 : 0)
3   #define KEY_UP(vk_code)        ((GetAsyncKeyState(vk_code) & 0x8000) ? 0 : 1)
```

的逻辑，但是，也要注意下，按键去抖；

子弹遇到边界或这敌方子弹，消失的时候，对应要free掉产生时的空间，同时，维持链表结构不被破坏；

子弹遇到敌方坦克，敌方坦克消失，这个处理，可以在相遇的位置，设置一个不同的值；坦克移动的时候，检查，它的值是不是包含这个，包含，证明自己被击中，就要销毁掉它的数据结构，尤其注意子弹的销毁；

关于，存档，如果杀死了一个坦克，这时候保存，要注意消失的是哪个，因此，需要记录下标；不过，后来，设计的时候，每死掉一个坦克，就会自动产生一个，因此，这个也算是一个预留的接口吧；

单线程处理起来，有时候，会感觉到控制坦克时，有时候会很快，有时候，会比较慢，这也体现出了一定的局限性；

另外，最为重要的一点，思想要灵活，各种语言的规范，规矩，互相之间，可以在遵循某些情况的前提下，进行变通和引进；

记录至此；

至于画图，等其他功能，和贪食蛇基本类似，可以拿来主义；

Reginald.S
2017.11.06