第8章-list-map简单使用

1. list

链表是由节点之间通过指针连接而成的链式结构存储结构体，对于链表，c++ 标准库中已经提供了封装好的链表了。

> require:
> #include< list > 包函头文件
> using namespace std; 打开标准命名空间

定义链表，并在首、尾 添加、删除元素。

```
list<int> lst;  //定义链表对象，list后<>中指定节点元素类型

lst.lst.push_front(0);  //头添加
lst.lst.push_back(1);   //尾添加

lst.pop_front();  //删除头节点
lst.pop_back();   //删除尾节点

```

迭代器遍历链表。

```
//begin(): 返回头结点
//end()  : 返回无效的尾节点
list<int>::iterator  ite = lst.begin();  //定义迭代器指向头结点
while (ite != lst.end()) {    //不等于链表的尾节点
    cout << *ite << "    ";   //operator*
    ite++;                    //operator++
}
```

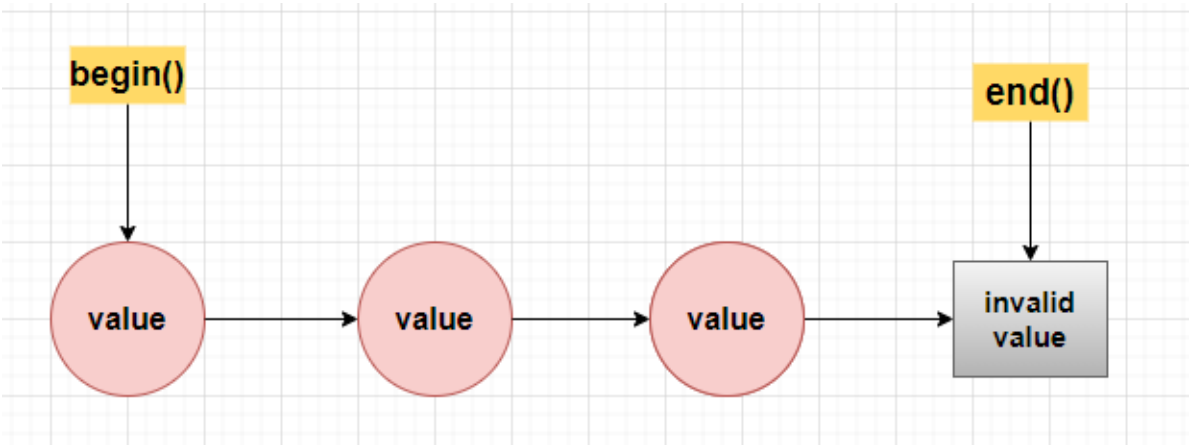
任意位置插入删除。`insert` 、 `erase`。

```
ite2 =  lst.insert(ite1, value);  //在 ite1 指向的位置之前插入元素value，返回插入元素的迭代器

ite2 =  lst.erase(ite1);  //删除 ite1 指向的节点，ite1 将失效不可用，返回删除节点的下一个节点


```

```
ite =  lst.erase(ite);    //多数情况下，用于删除的迭代器也可以承接其返回值，自带++效果
```

获取首尾节点中元素的值。注意：end() 返回的是无效的尾节点，不能对其进行间接引用。

```
lst.front();  //*lst.begin();
lst.back();   //*(--lst.end());
```



使用增强的范围for循环进行遍历链表。

```
for (int v : lst) {
    cout << v << "   ";
}

//加引用 可以修改节点里的值
for (int &v : lst) {
    ...

    v = vlaue;

    ...
}
```

其他常见的函数。

```
lst.empty(); //判断当前链表是否为空（bool类型），空返回true,非空返回false
lst.size();  //获取链表的长度
lst.clear(); //清空链表，empty 为true ,size 为0
```

2. map

map 为映射表，每一个元素称之为键值对（pair），分为键值（key） 和 实值(value)，键值不能重复，所有元素都会根据元素的键值自动被排序。

> require:
> #include< map > 包函头文件
> using namespace std; 打开标准命名空间

定义map，添加、遍历、删除

```
//格式: map<key,value> mm;
map<char, int> mm;

//格式: mm[key] = value;
mm['b'] = 1;   //使用[] 添加元素
mm['d'] = 2;
mm['a'] = 3;
mm['c'] = 4;

//使用函数插入元素，参数为键值对pair
mm.insert(pair<char, int>('e',5));

//迭代器遍历map
map<char, int>::iterator ite = mm.begin();
while (ite != mm.end()) {
    //first 对应键值，second 对应实值
    cout << ite->first << "-" << ite->second <<"    ";
    ite++;
}
//自动按照键值进行排序
//a-3    b-1    c-4    d-2    e-5

mm['c'] = 40;  //由于键值 'c' 已经存在，所以此处为修改实值
//a-3    b-1    c-40    d-2    e-5


//删除 ite 所指向的元素，返回删除元素的下一个，
map<char, int>::iterator ite2 = mm.erase(ite);

//同链表erase一样自带++效果
ite = mm.erase(ite);
```

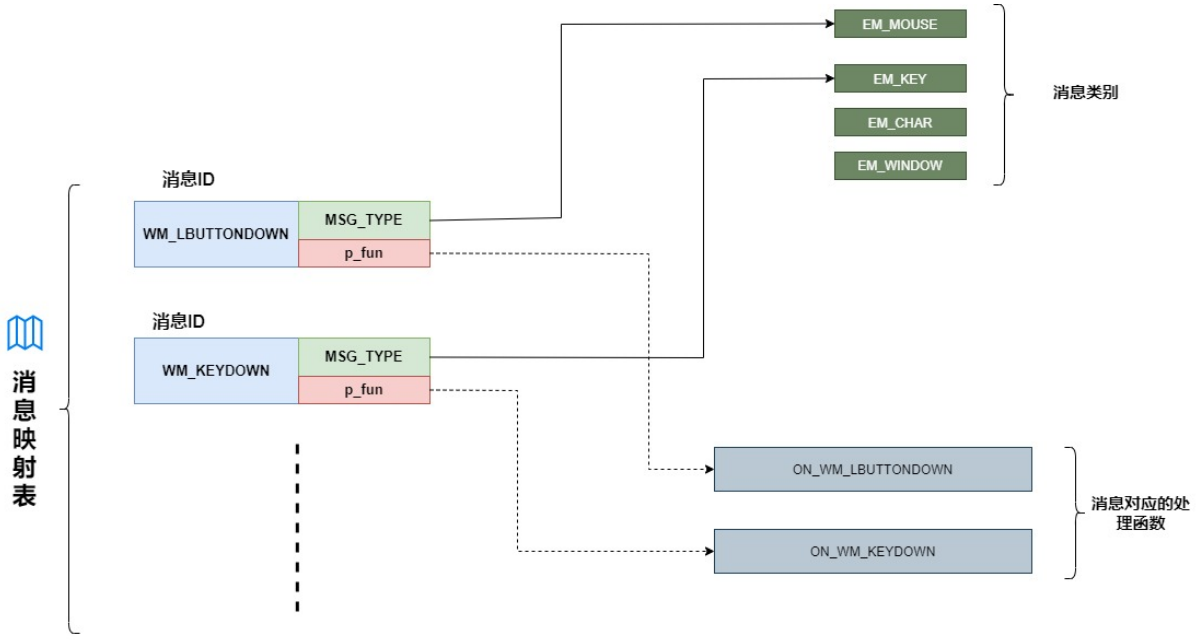
使用增强的范围for循环遍历

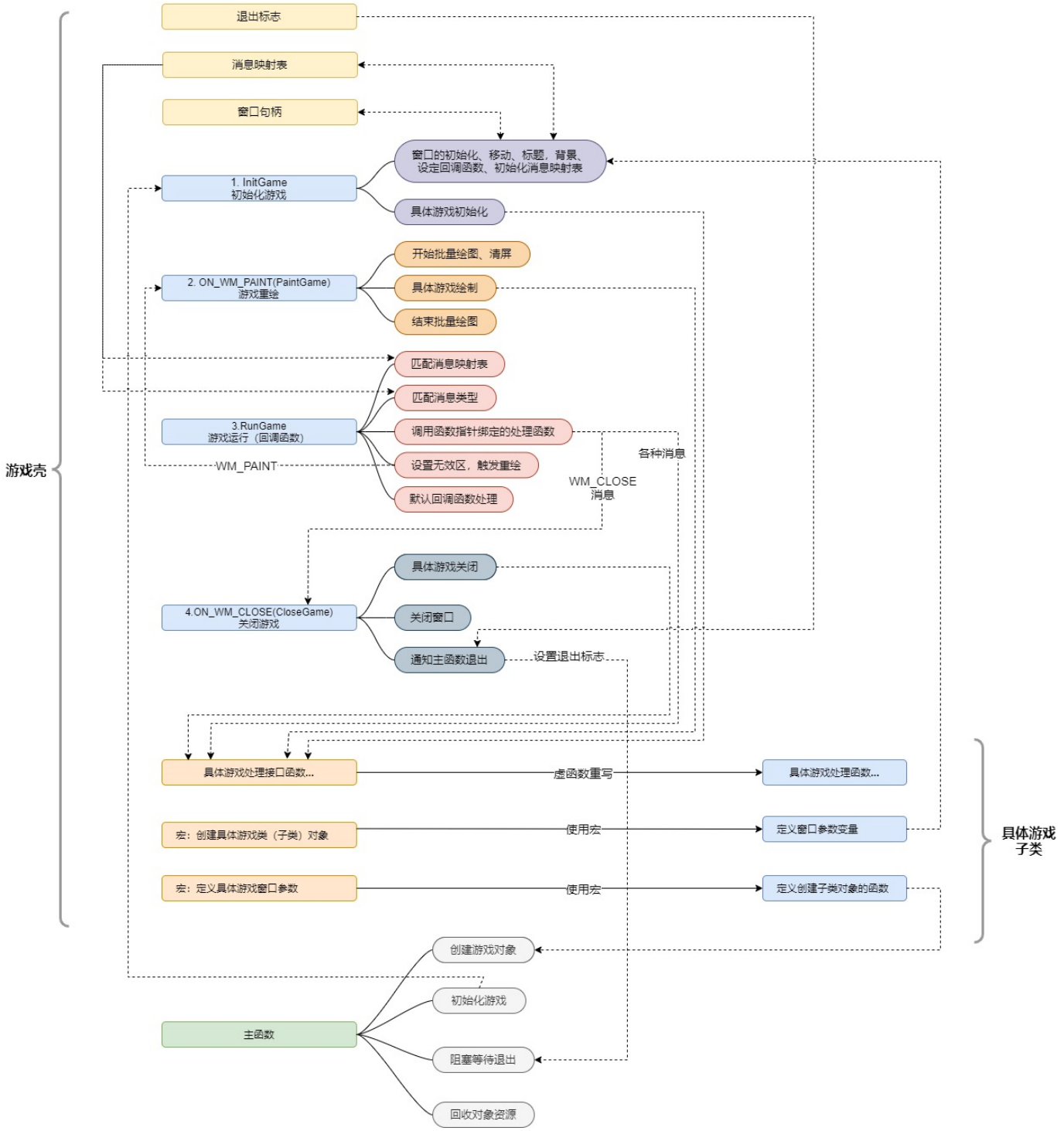
```cpp
for (pair<char, int> pr : mm) {
    cout << pr.first << "-" << pr.second << "    ";
}

//使用引用，可以修改实值
for (pair<const char, int> &pr : mm) {  //注意：因键值不能修改，故const不能省略否则报错
    ...

    if (pr.first == key) {
        pr.second = value;
    }

    ...
}
```

count 统计：

```cpp
map<char, int> mm;

//输出键值对应的实值，键值不存在，则自动添加
cout << mm['f'] << endl;

//统计某个键值出现的次数，map键值唯一，可以用来判断某个键值是否存在
int num = mm.count('f');
```

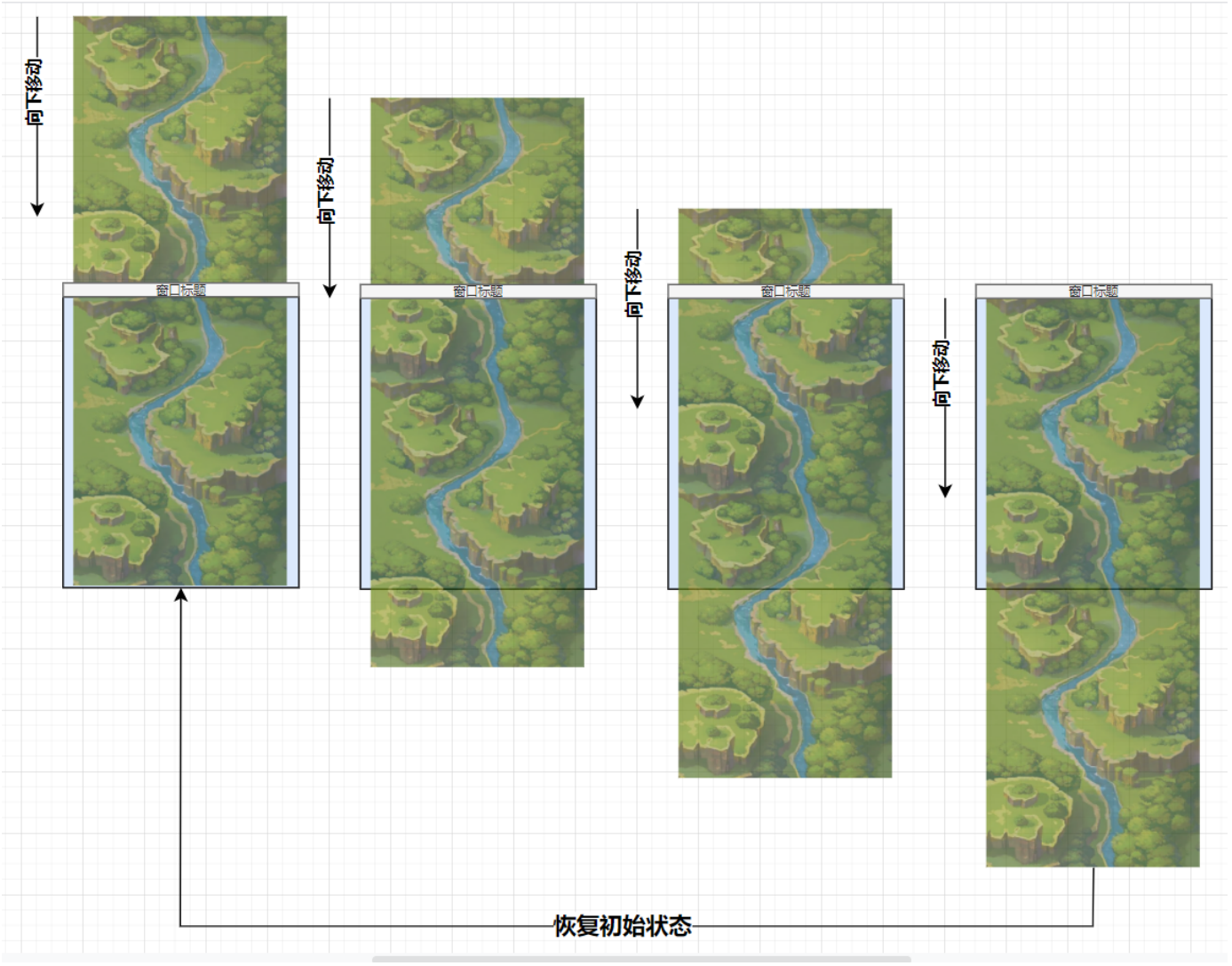
3. 游戏壳消息映射表示意图





消息映射表

4. 飞机大战背景图移动示意图



5. 飞机大战去白边原理图

原图 <-----大小一致------> 屏蔽图

原图需要去掉的部分为白色，屏蔽图对应位置为黑色

原图需要保留部分（机身），屏蔽图对应位置为白色

需要去掉部分

背景图颜色： 1 0 0 1

屏蔽图（黑色） 0 0 0 0

位或

1 0 0 1

原图（白色） 1 1 1 1

位与

最终得到颜色为**背景色** 1 0 0 1

需要保留部分

背景图颜色： 1 0 0 1

屏蔽图（白色） 1 1 1 1

位或

1 1 1 1

原图（飞机颜色） 1 1 0 0

位与

最终得到颜色为**飞机色** 1 1 0 0

结论：

黑色 位或 任何颜色 = 任何颜色

黑色 位与 任何颜色 = 黑色

结论：

白色 位与 任何颜色 = 任何颜色

白色 位或 任何颜色 = 白色