Roy

如何从 0 开始创建一个国家？

——国家/国策/民族精神/事件/本地化文件简介

By 已经到了哥伦比亚的 Roy

11. 前言

以下教程仅是我学习代码半年之后的一些个人经验和体会，不可避免的有疏漏之处，可以作为大家自学制作国家相关内容的初步入门教程，但是一些更加进阶的内容，我也没有学会，所以就不说了（ 有需要的可以自己在 wiki 或者找其他教程来学习。不过我认为，大家文案能够想到的东西（大多数也是自己在本体或者其他 mod 里面看到的机制），通过我下面的教程，应该都能够实现。

另外，虽然文案和代码在组内是有区分的，但是我觉得文案还是应该掌握一些基础的代码知识，倒不一定是说都要两开花，只是说如果文案能够懂一些代码的东西，那么对自己文案里面写的东西实现的可行性和难度就会有基本的把握，也就能够提高文案和代码沟通的效率。

12. 需要用到的东西

1.1 钢铁雄心本体（）

1.2 Sublime Text

（不要用 Windows 自带的笔记本，会出问题）

（如果平常也会写其他代码，可以考虑用 Visual Studio Code，里面会对代码的内容进行颜色标记，方便观察，其中甚至有 HOI4 做 mod 的扩展，可以试一下）（不是 Visual Studio，这个是微软很早之前开发的用来编 C/C++的软件，使用起来比较笨重，而且和我们的目的不符）

1.3 任意一个已经做好的国家的代码

（方便 CtrlC CtrlV）

12.1. HOI4 Wiki

英文版 https://hoi4.paradoxwikis.com/Hearts_of_Iron_4_Wiki

中文版 https://hoi4.parawikis.com/wiki/%E9%A6%96%E9%A1%B5

（中文版汉化不完全，而且内容有缺失，不如用英文版）

13. 开始之前的准备

13.1. 区分效果、修正这两个概念

效果：指的是一些可以被执行的，执行完成之后效果立刻出现并且不再影响后续的指令，一般用于国策执行完成之后的效果或者事件选项的效果。

修正：指的是一些对游戏里面的实体进行修改，使其所处状态发生变化的指令，一般用于民族精神或者内阁的特征内容。

这两个的区别主要在适用场合上，而这两个能够做到的事情是有交集的（比如效果和修正都可以让稳定度+10%，国家元首只能用效果改，而消费者工厂只能用修正），因此需要根据文案的目的，选择合适的条目来改。比如如果想要通过一个国策来修改消费者工厂的占用，则必须使用民族精神来改，直接通过国策的效果来改是不可行的。

14. 常用的文件位置

Mod 文件夹下 common/characters/国家代码.txt：国家的人物

（1.11 增加的新机制，把国家元首、政党领袖、内阁、将领合并成一个人物，然后对于这个人物，可以增删他的职责，或者如果人物死去，那么他也会从其他职责上消失，但这也带来了更多的 bug（））

common/country_leader/00_AN_traits.txt：国家元首/内阁特征（整个 mod 使用同一个文件，注意冲突）

common/decisions/国家代码.txt：国家的所有决议

common/decisions/categories/国家代码_decision_categories.txt：国家的所有决议类别（决议必须属于某一个决议类别）

common/dynamic_modifiers/国家代码_ dynamic_modifiers.txt：国家的动态修正（可以使用变量的民族精神）

common/ideas/国家名称.txt：国家的民族精神

common/ideologies/00_ideologies.txt：全局的意识形态文件（整个 mod 使用同一个文件）

common/national_focus/国家名称.txt：国家的国策树

common/unit_leader/00_traits.txt：将领的特征文件（整个 mod 使用同一个文件，注意冲突）

events/国家代码.txt：国家的所有事件

history/countries/国家代码 – 国家名称.txt：国家的开局设置文件

localisation/english/AN_countries_l_english.yml：国家名称的本地化文件（整个 mod 使用同一个文件，注意冲突）

localisation/english/AN_国家代码_l_english.yml：国家具体内容的本地化文件

15. 开始制作一个国家的内容

15.1. 修改国家开局设置

打开任意一个国家的开局设置文件（history/countries/国家代码 – 国家名称.txt），其内容如下：

```
capital = 1 #定义首都位置（使用State ID，也就是包含首都那个胜利点的地块ID，而不是首都本身的province ID）

oob = "SRC_Start" #OOB文件名称，解释见注1

#以下是一些游戏开局时便执行的效果，用于对国家开局状态进行设置，可以任意添加，关于效果内容的解释详见后文

add_ideas = {
    SRC_Siracusa_parliament
    SRC_high_army_loyalty
    SRC_family_conflict
    SRC_unprofessional_army_1
} #开局拥有的民族精神，定义在common/ideas/国家名称.txt

#招募开局的人物，定义在common/characters/国家代码.txt，见注2
recruit_character = SRC_Madam_Sicily
recruit_character = SRC_middle_man_Milner
recruit_character = SRC_Clift_Dorna
recruit_character = SRC_Carnat_Brown
recruit_character = SRC_Leodo_Zener
recruit_character = SRC_Canovis_Jonah
recruit_character = SRC_Familer_Querido
recruit_character = SRC_Infected
recruit_character = SRC_Berito_Hubert

set_research_slots = 2 #设置开局所拥有的科研槽数（整数）
set_stability = 0.5 #设置开局的稳定度（小数）
set_war_support = 0.4 #设置开局的战争支持度（小数）

set_technology = {
    infantry_weapons = 1
    infantry_weapons1 = 1
    tech_recon = 1
    tech_support = 1
} #设置开局时已经拥有的科技

set_politics = {
    ruling_party = democracy
    last_election = "1936.1.1"
    election_frequency = 48
    elections_allowed = no
} #设置开局时的政治情况，包括开局执政党以及其他内容（大多数时候其他内容没什么用（）

set_popularities = {
    traditionalism = 5
    democracy = 45
    authoritarianism = 10
    capitalism = 14
    moderatism = 12
    socialism = 0
    liberalism = 1
    occultism = 6
    extremism = 7
    particular_ideology = 0
} #设置不同党派的支持率（党派名称参见common/ideologies/00_ideologies.txt，见注3）
```

注 0：使用#，可以在代码文件中任意添加注释内容，系统不会读取本行内#号后面的

内容。

注 1：OOB 文件指的是所有国家开局拥有的部队模板文件，在 history/units/国家代码_Start.txt，在国家开局设置文件中只需要填写文件名即可（不加.txt）。OOB 文件的内容详解参见 https://hoi4.paradoxwikis.com/Division_modding。

注 2：人物的定义文件在 common/characters/国家代码.txt，其中具体的内容见下文。

注 3：意识形态的定义文件在 common/ideologies/00_ideologies.txt，其中对于国家新增内容有用的是 ideologies 这个括号下面的那一段，代表意识形态的第一级，可以用在这里设置开局各种意识形态的支持率比例，也可以用在后续国策/事件中增减意识形态的支持率。而在每一个意识形态的括号下面，types 下面的每一项是意识形态的第二级，主要是用在设置党派领袖具体的意识形态。

人物文件的内容如下：

```
characters = {
    SRC_Carnat_Brown = {          #人物ID
        name = SRC_Carnat_Brown #人物姓名在本地化文件中对应的代码，参见注1
        portraits = {             #人物画像
            civilian = {          #人物在内政界面（内阁或者国家领袖）的画像
                large = "" #gfx/leaders/SRC/Carnat_Brown.dds
                small = "" #gfx/leaders/SRC/Carnat_Brown_small.dds
                #大图用于国家领袖，小图用于内阁成员
            }
            army = { #人物在军事界面（选择将领）的画像
                large = "" #gfx/leaders/SRC/Carnat_Brown.dds
            }
        }
        #以下是对于这个人物担任不同职责的定义
        country_leader = { #国家领袖（党派领袖）
            ideology = paternalistic_dictatorship #意识形态，使用二级意识形态名称
            traits = {  }                         #国家领袖的特征，参见注2
            expire = "1965.1.1.1"                 #过期日期，一般没用
            id = -1
            desc = SRC_Carnat_Brown_desc #作为国家领袖的描述在本地化文件中的代码
        }
```

```
        corps_commander = { #陆军上将
            traits = { politically_connected trait_reckless trickster }
            #将领特征，参见注3
            skill = 5                #将领等级
            attack_skill = 6         #攻击等级
            defense_skill = 2        #防御等级
            planning_skill = 3       #计划等级
            logistics_skill = 3      #后勤等级
        }
        advisor = { #政治顾问，可以作为内阁成员/三军总长/总司令/理论家
            slot = political_advisor         #政治顾问的所属槽位
            idea_token = SRC_Carnat_Brown    #政治顾问的ID
            #标为同一ID的政治顾问不能同时出现在两个职位上

            cost = 100                       #聘用花费的政治点数
            removal_cost = 0                 #撤职花费的政治点数（设为-1则不可移除）

            allowed = {
                tag = SRC
            } #开局时对这一条进行检测，如果成立则这个国家可以选择该人物作为政治顾问

            visible = {
                tag = SRC
            } #决定该人物是否可见

            available = {
                tag = SRC
            } #决定该人物是否可以选择

            traits = { military_organizer } #政治顾问的特征
        }
    }
}
```

注 1：本地化文件在 localisation 文件夹中，文件后缀为 yml，其中 english 放的一般是 mod 中独有的文本，而 replace 放的是在本体中沿用的内容。本地化文件示例如下图所示：

```
l_english:

#国家领导人/将领/内阁
 SRC_Carnat_Brown:0 "卡尔纳特·布朗"
 SRC_Carnat_Brown_desc:0 ""
 SRC_Canovis_Jonah:0 "卡诺维斯·约拿"
 SRC_Canovis_Jonah_desc:0 ""
```

首行必须为 **l_english:**，然后每一行的格式是：<对应代码>:0 “内容”，每一行前必须有

一个空格。同时，除了这里引号（引号也是英文的引号）内部的内容和#后面的注释内容之外，所有的代码中不能有中文符号。以及最重要的一点是：**本地化文件必须要用 UTF-8 with BOM 格式进行编码，否则游戏无法识别！**

注 2：国家领袖/政治顾问的特征在 common/country_leader/00_AN_traits.txt 中，其内容如下图所示：

```
leader_traits = {
    opposite_of_the_family = {          #特征名称，也是本地化文件中的代码
        random = no                     #该特征是否能够被新创建的人物获得

        #以下是该特征所赋予的修正，关于修正的内容在民族精神部分会讲的更详细一些
        stability_factor = 0.05
        political_power_gain = -0.1
        democracy_drift = 0.07

        ai_will_do = {                  #决定ai是否会选择拥有该特征的政治顾问
            factor = 1
        }
    }
}
```

注 3：将领的特征参见 common/unit_leader/00_traits.txt，如果实在无法找到对应的特征，可以在 localisation/replace/r_traits_l_english.yml 这个文件中搜索中文名，就能找到对应的特征的英文 ID。创建新将领特征的方法参见 https://hoi4.paradoxwikis.com/Character_modding#Unit_leader_traits

16. 开始编写国家的国策

打开任意一个国家的国策树文件（common/national_focus/国家名称.txt），其内容如下：

```
focus_tree = {
    id = Siracusa_start_focus #国策树的ID

    country = {               #决定这一国策树被赋予哪一个国家，见注1
        factor = 0
        modifier = {
            add = 20
            tag = SRC
        }
    }

    default = no              #是否为默认国策树
    continuous_focus_position = { x = 20 y = 1800 } #连续性国策位置

    focus = {                 #国策具体内容，见下文
        ......
    }

}
```

注 1：每一个国家的国策树由上图所示的一个至多个 focus_tree 块组成，但是同一时刻一个国家的全部国策只能有一个 focus_tree 块中的内容。那么，游戏读取时，它就会把所有的 focus_tree 块全部读取，然后决定每一个国家究竟使用哪一个 focus_tree 块时，它会根据你在这一个块当中填写的内容来决定。factor 后面的数字代表默认值，然后此处 modifier 的内容就是使得这个 focus_tree 块对于国家代码为 SRC（叙拉古）的国家来说加 20 分。那么，如果没有其他特殊情况，那么对于叙拉古，这个 focus_tree 块有 20 分，其他 focus_tree 块只有 0 分，所以它会将这个 focus_tree 块中的内容加载作为叙拉古的国策树。

在每一个 focus_tree 中，除了图中出现的这些内容外，最重要的就是每一个国策具体内容。每一个国策的具体内容就在上图所示的 focus 块中，示例如下：

```
#议题：基建建设
focus = {
    id = SRC_agenda_infrastructure  #国策ID，见注1
    icon = GFX_focus_unfinshied     #国策图标
    x = 10                          #国策位置，见注2
    y = 1

    cost = 1                        #国策时长（以7天为单位）

    ai_will_do = {                  #AI执行国策概率，见注3
        factor = 2
    }

    allow_branch = {                #国策显示的前提条件，见注4
        NOT = {
            has_country_flag = {
                flag = SRC_consultative_conference_explosion
                value > 0
            }
        }
    }

    prerequisite = {                #前置国策，见注5
        focus = SRC_agenda_economy_advancement
    }

    mutually_exclusive = {          #互斥国策，见注6
        focus = SRC_advocate_east_threat
    }

    available = {}                  #国策的其他前置条件，见注7

    cancel_if_invalid = yes         #国策在不满足前置条件时是否取消
    continue_if_invalid = no        #国策在不满足前置条件时是否继续
    available_if_capitulated = no   #国策在投降后是否还可以执行

    select_effect = {               #国策在点选之后立刻执行的效果
        ......                      #使用effect（效果）
    }
```

注 1：国策 ID 也用在本地化文件中，国策名对应的代码就是 ID 本身（本例中为 SRC_agenda_infrastructure），国策的描述对应的代码是 ID 后加上“_desc”（本例中为

SRC_agenda_infrastructure_desc）。国策最好使用国家代码作为前缀。

注 2：国策的位置使用的是平面直角坐标系，左上角为（0,0），x 越大越靠右，y 越大越靠下。根据我的经验，相邻的两个国策，x 的数值应当差 2，而 y 的数值应当差 1。

注 3：此处设定的是 AI 选择此国策的概率，同上述国策树的分配一样，有基础数值，同时在不同的条件下该数值可以修改，数字越大，AI 就更有可能执行这一国策。

注 4：此处描述两种切换国策树显示的方法：

16.1. 如果你希望在执行完一段国策之后，出现一些新的国策，并且此前的国策全部消失

（如 TNO），那么就可以把两段国策树放在不同的 focus_tree 块中，然后通过改变 country 中的数值来确定初始国策树。在初始国策树执行完成后，在最后一个国策的效果中加上 load_focus_tree = <第二段国策树的 ID>。这样，在完成最后一个国策后，就会载入全新的国策树。

16.2. 如果你希望在新国策树出现后，仍然保留原有的国策

我自己的方法就需要使用到这里说的 allow_branch 条件。allow_branch 这个块中需要写特定的条件，只有满足这些条件这个国策才会出现，否则就会隐藏起来（但是位置会空着）。这样的问题是，这个条件游戏只会在开始的时候检查一遍，所以为了实现我们的目的，如果想要在游戏进行中也让它检查国策是否该出现，那就需要在特定的位置加上 mark_focus_tree_layout_dirty = yes 这一效果。以叙拉古的国策为例，叙拉古的国策分为几个阶段，只有完成上一阶段的国策下一阶段的国策才会出现。因此，下一阶段的国策的 allow_branch 当中就需要一个特定的 flag，然后在上一阶段最后的国策中设置这个 flag，并且加上 mark_focus_tree_layout_dirty = yes 这一效果。这样，下一阶段的国策就会自然出现，而上一阶段的国策也不会消失。

注 5：如果需要多个前置国策中满足一个即可，那么就可以写：

```
prerequisite = {
    focus = focus_id_1
    focus = focus_id_2
}
```

如果需要多个前置国策并且全部满足，那么就可以写：

```
prerequisite = {
    focus = focus_id_1
}
```

prerequisite = {

focus = focus_id_2

}

注 6：如果需要多个互斥国策，则需要把这些国策写在同一个 mutually_exclusive 块当中。

注 7：这里需要填写条件，条件的具体内容参见

https://hoi4.paradoxwikis.com/Conditions

注 8：跳过国策的条件写在 bypass = {} 里面。

国策的重要内容在于效果，下面简要描述一些常用的效果：（全部的效果具体内容参见 https://hoi4.paradoxwikis.com/Effect ）

16.3. 相关代码段描述

16.3.1. 将特定的人物设定为该意识形态的领导人

promote_character：

使用方式为

<人物 ID> = {

promote_character = yes

}

16.3.2. 招募/退休特定的人物

recruit_character/retire_character：

退休指令会使其失去所有位置（包括政党领导人，内阁成员，上将等）

格式为

recruit_character = <人物 ID>

16.3.3. 在 game.log 中输出特定的内容，用于 debug

log：

格式为

log = “”（英文双引号，引号内和本地化文件使用方法一致）

16.3.4. 在某一个国家中设定一个标记（flag）

set_country_flag：

可以用于标记一些特定的状态（完成某一阶段国策，国家改革尚未完成等），格式如下：

set_country_flag = {

flag = <名称>（不要与其他内容冲突）

days = <特定天数>（指定 flag 的维持天数，非必要）

value = 1

}

16.3.5. 在指定时间内触发国家事件

country_event：

格式如下：

country_event = {

id = <事件 ID>

days = <数字>（在特定天数后触发）

random_hours = <数字>（在触发时间到达后，还会有一段时间之后才触发，最多还有这里的数字这么多小时，非必要）

random_days = <数字>（同上，只不过单位为日，非必要）

}

16.3.6. 加政治点数/指挥点数

add_political_power/add_command_power：

格式为 add_political_power = <数字>（整数，可用负数）

16.3.7. 加稳定度/战争支持度

add_stability/add_war_support：

格式为

add_stability = <数字>（小数，可以用负数）

16.3.8. 加三军经验

army_experience/navy_experience/air_experience：

格式为：

army_experience = <数字>（整数，可用负数）

16.3.9. 增加特定意识形态的支持度

add_popularity：

格式为：

add_popularity = {

ideology = <意识形态名称>（为第一级意识形态名称）

popularity = <数字>（小数，可用负数）

}

16.3.10. 设置关于政治的相关内容（选举）

set_politics：

格式为

set_politics = {

ruling_party = <意识形态名称>（为第一级意识形态名称）

election_allowed = <yes/no>（是否允许选举，非必要）

last_election = <"年.月.日">（上次选举时间，非必要）

election_frequency = <月数>（选举频率，非必要）

long_name = <本地化文件代码>（党派长名称，非必要）

name = <本地化文件代码>（党派名称，非必要）

}

16.3.11. 改变和特定国家的关系

add_relation_modifier：

格式为

<国家代码>（国家 A） = {

add_relation_modifier = {

target = <国家代码>（国家 B）

modifier = <modifier 代码>

}

}

这样，国家 A 对国家 B 的关系就会有对应的 modifier，这一 modifier 的定义在 common/opinion_modifiers/<国家代码>_opinion_modifiers.txt，示例如下：

```
opinion_modifiers = {
    LAT_Laterano_conflict_SRC = {        #modifier代码，见注1
        value = -40                      #关系修正的数值
    }
    LAT_peace_with_Siracusa_SRC = {
        value = 25
    }
}
```

注 1：关系修正的本地化文件可以写在对应国家的本地化文件中，modifier 代码也就是它在本地化文件中的代码。

16.3.12. 让目前国家对特定国家宣战

declare_war_on：

格式为：

declare_war_on = {

target = <国家代码>（被宣战国家的代码）

type = <宣战借口代码>（对应的文件在 common/wargoals 文件夹中）

}

16.3.13. 增加火星建筑（不一定局限于工厂）

add_offsite_building：

格式为

add_offsite_building = {

type = <建筑代码>（建筑代码可以在

https://hoi4.paradoxwikis.com/Building_modding#Types 中找到，是表格中的 internal name 这一列，具体的定义在 common/buildings/00_buildings.txt）

level = <数字>（整数，可以用负数）

}

16.3.14. 没有任何效果，只是在国策效果中显示解锁特定决议/决议类别的提示。

unlock_desicion_tooltip/unlock_decision_category_tooltip：

格式为：

unlock_decision_tooltip = <decision 名称>（决议相关内容见下文）

HOI4 的决议是否解锁是需要在决议的 available 条件中填写，而没有办法通过国策效果直接解锁，因此如果一个国策完成后能够解锁一定的决议，则需要把完成国策写在对应决议的条件中，然后在国策的效果中加上这一条指令。

16.3.15. 增加科研槽数

add_research_slot：

格式为：

add_research_slot = <数字>（整数，可用负数）

16.3.16. 使某一领域的科技研究获得加速

add_tech_bonus：

格式为：

add_tech_bonus = {

bonus = <数字>（小数，代表研究速度能够增加多少）

uses = <数字>（整数，代表总共可以使用几次）

category = <科技类别>（可以在 common/technology_tags/*.txt 中找到）

}

16.3.17. 增加新的民族精神

add_ideas：

格式为：

add_ideas = <民族精神代码>或者

add_ideas = {

<民族精神代码>

<民族精神代码>

……（可以填写多个民族精神代码）

}

民族精神的详细解释见后文。

16.3.17.1. 增加一个仅生效一段时间的民族精神

add_timed_idea：

格式为：

add_timed_idea = { （注意 idea 后面没有加 s）

idea = <民族精神代码>

days = <数字>（生效日数）

}

16.3.17.2. 增加一个民族精神的同时删除另一个

如果两个民族精神在本地化文件中的名称一致，则显示为该民族精神获得修正；如果名称不一致，则显示为更换为新的民族精神，并且显示两个民族精神之间的不同。

swap_ideas：

格式为：

swap_ideas = {

remove_idea = <民族精神代码>（被去除的民族精神，注意 idea 后面没有 s）

add_idea = <民族精神代码>（增添的民族精神，注意 idea 后面没有 s）

}

16.3.17.3. 去除已经存在的民族精神

remove_ideas：

格式为：

remove_ideas = <民族精神代码>或者

remove_ideas = {

<民族精神代码>

<民族精神代码>

……（可以填写多个民族精神代码）

}

16.3.18. 改变库存的装备数量

add_equipment_to_stockpile：

格式为：

add_equipment_to_stockpile = {

type = <装备名称>（装备名称见 common/units/equipment/*.txt，一般使用原型代码，毕竟大多数时候不需要特殊指定是哪一代）

amount = <数字>（为了避免 bug，一般需要检查是否有足够的装备）

producer = <国家代码>（指定装备的生产国，非必要）

variant_name = “型号代码”（适用于坦克和舰船等玩家指定的名称，非必要）

}

16.3.19. 在某地开始建设建筑

add_building_construction：

格式为：

<state 编码> = {（此处可以使用 random_owned_state，表示在本国控制的任意 state 中随机选择一个，适用于国策加工厂）

type = <建筑代码>

level = <数字>（整数）

instant_build = <yes/no>（建筑是否立刻获得，亦或是增添进建筑队列中）

province = <province 编码>（取决于建筑类型决定是否必要）或者

province = {

all_provinces = <yes/no>（是否对该 state 中的所有 province 都生效）

limit_to_coastal = <yes/no>（是否只适用于临海的 province）

limit_to_naval_base = <yes/no>（是否只适用于已经有海军基地的 province）

limit_to_border = <yes/no>（是否只适用于边界 province）

limit_to_victory_point = <yes/no>（是否只适用于有胜利点的 province）或者

limit_to_victory_point >/< <数字>（只适用于胜利点大于一定数值的 province）

level = <数字>（只适用于指定建筑等级已经超过该数值的 province）

id = <province 编码>

}

}

除此之外，还有一些可以在效果当中使用的语句：

16.3.20. 如果需要判断条件来决定某些效果是否生效，则使用这一语句

if/else_if/else：

格式为：

if = {

limit = {

<条件>

}

<效果>

}

else_if = {（注意，这里中间使用的是_而不是空格）

limit = {

<条件>

}

<效果>

}

……（还可以接多个 else_if 语句）

else = {

<效果>

}

16.3.21. 有的效果并不希望让玩家看到（比如变量、flag、事件等）

hidden_effect：

可以把不想被看到的效果包含在一个 hidden_effect 块中，这样效果仍然会生效，但是

玩家不会看到。

16.3.22. 随机效果

有时会希望事件的效果有一定的随机性，这时需要使用 random 和 random_list 这两个效果。格式为：

random = {

chance = <数字>（整数，0~100，数字越大代表可能性越高）

<效果>

}（这样就有百分之 chance 的几率效果会生效，其余时候效果不会生效）

random_list = {

<数字> = {（代表对应效果发生的概率）

<效果>

}

……（可以增加多个块，代表不同概率下究竟哪一个效果会生效，数字之和不必等于 100）

}

16.4. 编写国家的民族精神

国家民族精神分为两种，一种是常见的显示在界面中的民族精神，另一种则是隐藏起来的民族精神。

16.4.1. 民族精神简介

民族精神文件（common/ideas/国家名称.txt）的内容如下图所示：

```
ideas = {
    country = {
        SRC_family_conflict = {          #民族精神ID
            picture = GFX_SRC_family_conflict_idea  #配图
            name = SRC_family_conflict  #民族精神本地化文件代码，注1

            allowed = {                  #允许生效的条件
                original_tag = SRC
            }

            modifier = {                 #提供的修正，见注2
                consumer_goods_factor = 0.05
                stability_factor = -0.1
            }

            targeted_modifier = {        #针对特定国家的修正，见注3
                tag = KRT
                attack_bonus_against = 0.1
            }

            research_bonus = {           #特定科技的研究速度，见注4
                dd_tech = 0.1
            }
```

```
            equipment_bonus = {          #装备数据的修正，见注5
                infantry_equipment = {
                    soft_attack = 0.1
                }
            }
        }
        #此后还可以有多个民族精神
    }
}
```

注 1：民族精神在本地化文件中有两个代码：name 后面对应的代码，在本地化文件中对应的是民族精神的名字；name 后面对应的代码加上“_desc”，在本地化文件中对应的是民族精神的描述。

注 2：民族精神提供的对其他方面的修正，可用的修正内容参见下文。

注 3：针对特定国家的修正，具体内容参见

https://hoi4.paradoxwikis.com/Modifiers#Targeted_modifiers

注 4：对特定科技研究速度的修正，正值代表研究速度加速。此处使用的是科技类别，详见 common/technology_tags/里面的文件

注 5：对不同装备的修正，其中使用到的一般都是装备原型，需要根据装备文件的内容进行修改。

常用的修正内容简介如下：（所有修正的内容参见

https://hoi4.paradoxwikis.com/Modifiers）

16.4.2. 修正参数

1. research_speed_factor：研究速度修正，输入小数，正数代表研究速度加快。
2. political_power_cost：每日政治点数消耗修正，输入小数，正数代表消耗增加。
3. political_power_gain：每日政治点数获取修正，输入小数，正数代表获取增加。
4. stability_factor：稳定度修正，输入小数，修正为百分比。（战争支持度同）
5. stability_weekly：每周稳定度增长修正，输入小数。（战争支持度同）
6. <ideology>_drift：意识形态每日新增支持率，需要使用具体的意识形态代码来代替<ideology>，输入小数。
7. production_factory_efficiency_gain_factor：工厂效率增长修正，输入小数。
8. production_factory_max_efficiency_factor：工厂最大效率修正，输入小数。
9. production_factory_start_efficiency_factor：工厂起始效率修正，输入小数。
10. consumer_goods_factor：消费品所需工厂，输入小数。
11. industrial_capacity_factory：军用工厂生产效率，输入小数。
12. line_change_production_efficiency_factor：生产线转换时效率保留，输入小数。
13. production_speed_buildings_factor：建设速度，输入小数。
14. production_speed_<building>_factor：不同类别建筑的建设速度，需要使用建筑的代码来代替<building>，输入小数。
15. training_time_factor：海军和陆军部队训练时间，输入小数。
16. training_time_army_factor：陆军部队训练时间，输入小数，正数代表训练时间延长。
17. army_attack_factor：陆军攻击修正，输入小数。（防御更换为 defence）
18. army_core_attack_factor：陆军在核心领土上的攻击修正，输入小数。（防御更换

为 defence）

19. army_morale：陆军恢复修正，输入整数。
20. army_morale_factor：陆军恢复比例修正，输入小数，转换为百分比。
21. army_org：陆军组织度修正，输入整数。
22. army_org_factor：陆军组织度比例修正，输入小数，转换为百分比。
23. army_org_regain：陆军组织度恢复修正，输入小数。

16.5. 隐藏民族精神

有时需要增加某些短期的修正，但同时又不希望写一个完整的民族精神，就可以使用隐藏的民族精神。隐藏民族精神如下图所示：

```
ideas = {
    hidden_ideas = {
        #议题：基建建设国策效果
        SRC_agenda_infrastructure_construction_idea = {
            modifier = {
                production_speed_infrastructure_factor = 0.25
                production_speed_rail_way_factor = 0.6
            }
        }
    }
}
```

隐藏民族精神定义在 ideas 块中建立的 hidden_ideas 块中，一般不需要其他内容，只需要写明修正的具体内容即可。本地化文件可以没有，因为隐藏的民族精神不会显示，但是如果隐藏的民族精神在一段时间后失效，则玩家会看到民族精神失效的信息，如果没有本地化文件，则系统会随机选取一个名称（英文的随机人名），难免会对玩家造成困惑，因此本地化文件中最好还是加上隐藏民族精神的名称（描述可以没有，毕竟也没有显示的机会）。

17. 编写事件

事件文件在单独的文件夹中（events/国家代码.txt），其中的内容如图所示：

```
add_namespace = focus_krt                #名称空间，见注1

country_event = {
    id = focus_krt.18                    #事件ID
    title = focus_krt.18.t               #事件标题
    desc = focus_krt.18.d                #事件描述，见注2
    picture = GFX_report_event_Evening_of_river

    is_triggered_only = yes              #是否只能通过效果触发，见注3
    fire_only_once = yes                 #是否只能触发一次
```

```
option = {                          #选项1
    name = focus_krt.18.a           #选项1名称
    ai_chance = {                   #AI选择选项1的概率，见注4
        base = 10
    }
    #其下为效果
    custom_effect_tooltip = focus_krt.18.tt.1
    hidden_effect = {
        random_list = {
            50 = {
                country_event = { id = focus_krt.19 days
            }
            50 = {
                country_event = { id = focus_krt.20 days
            }
        }
    }
}

option = {                          #选项2（同上）
    name = focus_krt.18.b
    ai_chance = {                   #AI选择选项2的概率
        base = 30
    }
    custom_effect_tooltip = focus_krt.18.tt.2
    hidden_effect = {
        random_list = {
            80 = {
                country_event = { id = focus_krt.19 days
            }
            20 = {
                country_event = { id = focus_krt.21 days
            }
        }
    }
}
```

注 1：事件的 ID 必须由这些 namespace 组成前缀，在文件刚开始时就需要定义这些 namespace。

注 2：事件的标题、描述和选项的名称就是其在本地化文件中的代码。

注 3：事件有两种类型，一种是自然的，满足条件后自动触发的事件；另一种则是只能通过特定效果触发的事件（如国策、决议等）。根据以往的经验，大多数事件属于第二类，因此此处展示的是这类事件的一般格式。如果希望事件独立触发，则可以参考 https://hoi4.paradoxwikis.com/Event_modding 中的内容，通过 trigger 来决定事件是否能够发生，然后通过 mean_time_to_happen 来控制在条件满足时事件何时被触发。

注 4：此处各个选项的概率之和不一定为 100。

每一个选项还可以加一个 trigger 块，只有满足其中的所有条件该选项才能出现并被玩家选择。

18. 编写决议

决议文件在 common/decisions 文件夹当中。每一个决议一定属于特定的类别，而类别在 common/decisions/categories 文件夹当中。决议类别文件的内容如图所示：

```
SRC_military_construction = {                    #决议类别名称，见注1
    icon = saf_anti_colonialist_crusade          #图标

    allowed = {                                  #游戏开局时判断是否存在
        original_tag = SRC
    }
    visible = {                                  #判断是否可见，见注2
        OR = {
            is_debug = yes
            AND = {
                has_completed_focus = SRC_military_prospect
                NOT = {
                    has_country_flag = {
                        flag = SRC_military_finished_recycling
                        value > 0
                    }
                }
            }
        }
```

注 1：决议类别和民族精神一样，在本地化文件中有分别对应标题（与 ID 一致）和描述（ID_desc）的代码，都需要写上，特别是描述相关的代码。

注 2：allowed 当中的条件仅在开局和读档时检测，一般只填写判断国家的条件。visible 决定决议类别是否可见，可以通过这个条件来决定决议是否被解锁。一般希望能够加上 is_debug = yes 条件，这样在 debug 模式中就能够看到所有的决议，而不需要被游戏本身的条件所限制。

决议的具体内容如图所示：

```
SRC_military_construction = {                    #决议类别ID
    SRC_government_pay_construction = {          #决议本身ID，见注1
        icon = generic_oppression

        allowed = {                              #开局判断条件
            tag = SRC
        }
        visible = {}                             #决议可见的条件
        available = {                            #决议可以被选择的条件
            has_completed_focus = SRC_military_prospect
        }

        cost = 80                                #消耗政治点数
        days_remove = 30                         #点选后移除天数

        fire_only_once = no                      #是否只能被点选一次

        select_effect = {}                       #点选后立刻生效的效果

        remove_effect = {                        #移除时生效的效果
            subtract_from_variable = {
                SRC.SRC_military_construction_foreign_infiltra
                tooltip = SRC_government_pay_construction_tt
            }
            clamp_variable = {
                var = SRC.SRC_military_construction_foreign_in
                min = 0
                max = 1
```

注 1：本地化文件中 ID 对应的就是决议题目的代码。

决议通过 visible 条件确定是否可见，也可以通过 available 条件确定是否可被点选。点选后的移除天数对于点选后立刻生效的决议可以不写，同样也不能有 remove_effect 和 modifier。

18.1. 变量机制简介

18.1.1. 变量的定义

就如同一般的编程语言一样，钢铁雄心 4 当中也有变量的存在，变量是分别定义在不同的范围内，比如变量可以定义在游戏/国家/state 当中。如果定义在国家中，则变量

名为<国家代码>.<变量名称>，如果定义在 state 当中，则是<state ID>.<变量名称>。

定义变量时需要使用 set_variable 效果，格式为

set_variable = {<变量名> = <变量名/数值>} 或者

set_variable = {

var = <变量名>

value = <变量名/数值>

}

这样就可以定义前面的变量，并且将其他变量或者具体的数值赋予这个变量。

对于变量，我们可以使用其他效果对变量的值进行修改，包括 add_to_variable（加），subtract_from_variable（减），multiply_variable（乘），divide_variable（除）和 modulo_variable（取余数）。格式为

add_to_variable = {<变量名> = <变量名/数值>} 或者

add_to_variable = {

var = <变量名>

value = <变量名/数值>

}

这样，我们就会把后面变量的数值或者输入的数值与前面的变量数值相加，并且将这个数值存储在前面的变量中。

同时，我们也可以将变量限制在一定范围内（比如 0~1 之内），使用 clamp_variable 效果。格式为

clamp_variable = {

var = <变量名>

min = <变量名/数值>

max = <变量名/数值>

}

这样，如果 var 对应的变量数值小于 min 对应的数值，那么 var 对应的变量数值就会被改为 min 对应的数值；如果 var 对应的变量数值大于 max 对应的数值，那么 var 对应的变量数值就会被改为 max 对应的数值。min 和 max 两个至少需要存在一个，但不一定需要两个都存在。

另外，如果需要对变量进行取整，则可以使用 round_variable = <变量名>，变量数值会被四舍五入。

在判断条件时，可以对变量进行判断，使用 check_variable 条件，格式如下：

check_variable = {

var = <变量名>

value = <变量名/数值>

compare = <比较类别>

}

可用的比较类别包括 less_than（小于），less_than_or_equals（小于等于），greater_than（大于），greater_than_or_equals（大于等于），equals（等于）和 not_equals（不等于）。

在本地化文件中，可以使用[?<变量名>]，使得在游戏中显示变量的数值。同时，如果希望能够对这个数值的显示进行修改，可以使用[?<变量名>|<修饰字符>]，包括颜色（如 Y 表示黄色），百分比（%），小数位数（数字）。

另外，有时我们需要一些比较复杂的公式计算，而这就需要进行多种计算的复合。按照此前的做法，我们应该设置多个变量来存储这些中间数值，但是我们也可以通过临时变量进行计算，只需要将此前效果中的 variable 改编为 temp_variable 即可。这些临时变量在执行完一个效果块之后就会自动清除，因此不需要担心变量名重复的问题。

最后，除了我们自己定义的变量外，游戏中还有很多系统内自己定义的变量。这些变量也可以和其他变量一样使用，具体内容参见

https://hoi4.paradoxwikis.com/Variables#Game_Variables。

18.1.2. 变量的应用

在国家的国策中，经常能够见到通过国策对同一个民族精神不断进行修正，从而消除一些开局 debuff 的影响。如果国策的执行顺序是确定的，那么在写国策效果时就可以直接使用 swap_ideas，还算是比较简单。但是，如果有两列国策，这两列国策都能对民族精神进行修正，则会面临很麻烦的情况，比如国策分别是 a1，a2，a3 和 b1，b2，b3，这两列国策可以分别按各自的顺序执行，那么在写 a2 的国策效果时，由于根本不清楚 b 系列国策的执行情况，因此如果按照民族精神来写，那么就需要对各种不同的情况进行分类判断，代码的复杂程度会极大提升。因此，我们需要使用新的方法来解决这个问题。动态修正作为一种机制，可以被理解为是可以使用变量的民族精神。动态修正文件在 common/dynamic_modifiers/<国家代码>_dynamic_modifiers.txt，定义方式类似于民族精神，但是随后只能使用修正，而且

修正后面的数值可以使用变量名。随后，在特定的时候可以使用 add_dynamic_modifier 效果添加这一动态修正，然后只需要在写国策时把对应的变量数值进行调整即可。当然，由于变量的调整通常不会显示效果提示，因此效果提示需要通过 custom_effect_tooltip 来指明。

除此之外，如果你希望在国家中通过一些其他机制来推动游戏的进行（通常是在决议中），那么变量也将会是经常用到的机制。

建筑代码

#省份建筑（上层-基础槽）

infrastructure = <int> #基建

air_base = <int> #空军基地

anti_air_building = <int> #防空炮

radar_station = <int> #雷达

#省份建筑（下层-建筑槽）

industrial_complex = <int> #民用工厂

arms_factory = <int> #军用工厂

dockyard = <int> #船坞

fuel_silo = <int> #油罐

synthetic_refinery = <int> #炼油厂

rocket_site = <int> #火箭基地

nuclear_reactor = <int> #核反应堆

#地块建筑

naval_base = <int> #港口

bunker = <int> #陆上堡垒

coastal_bunker = <int> #沿海堡垒

#区域等级

megalopolis #特大城市

metropolis #大都市

large_city #大都市

city #城市

large_town #城镇

town #乡镇

rural #农村

pastoral #田庄

small_island #小岛

enclave #飞地

tiny_island #特小岛

wasteland #荒地

#省份文件示范

state={

id=976 #省份 id

name="STATE_976" #省份本地化名称

state_category= metropolis #省份区域等级

history={ #历史分组

owner = KRT #省份所有者

add_core_of = KRT #省份核心

victory_points = { #胜利点分组

17306 50 #具体胜利点地块-胜利点值

}

buildings = {#建筑分组

infrastructure = 5

industrial_complex = 4

arms_factory = 3

17306 = { #地块建筑分组

bunker = 10

}

}

}

provinces={ #省份地块内容

17306

}

manpower= 386000 #省份人数

buildings_max_level_factor=1.000 #省份最高建筑等级（不用搞）

local_supplies=0.000 #新版本的后勤机制（？）

}

#常用项目

##省份建筑

buildings = {

infrastructure = 5

industrial_complex = 4

arms_factory = 3

}

#胜利点

victory_points = { #胜利点分组

17306 50 #具体胜利点地块-胜利点值

}

#其他

#在记录文川的本地化时，对于建筑可以记住这个口诀

#军民基-船炼罐