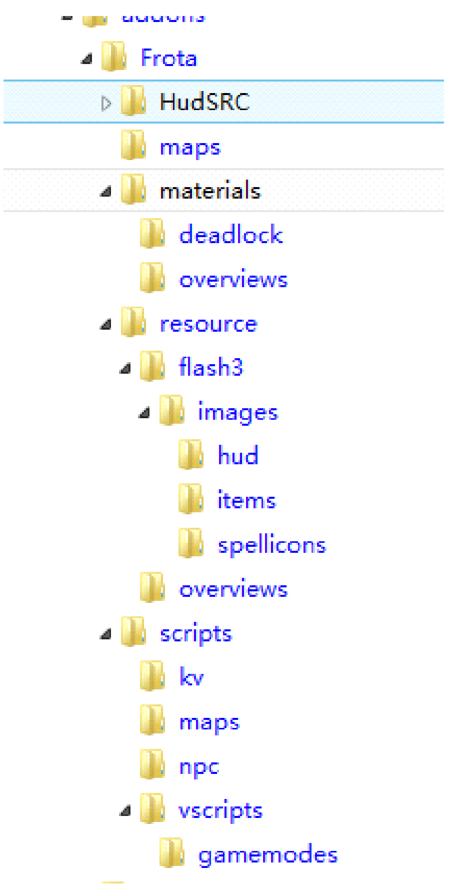
我所知道的关于DOTA2 RPG

2014年1月15日　　17:21

因为我们对于DOTA2 RPG的研究才刚刚开始且本人水平非常有限，因此错漏之处在所难免，希望大家包涵。
由于看此教程的，更多是WE党，因此我也会更多以WE的角度说说我对DOTA2 RPG制作的理解。
此外，本文主要的信息来源为官方frostivus自定模式，和国外ash47大神制作的Frota。

第一、 文件结构

文件架构方面



屏幕剪辑的捕获时间: 2014/1/16 9:26

文件架构方面，以Frota为例，为典型的文件架构，其中，HudSRC是ASH47为他的BP专门使用插件制作的FLASH资源，对此我们暂时先不做研究（好吧，我是不会说我根本不会玩Falsh和AS这种事的。）
其他文件夹，maps文件夹中包含DOTA2所需的BSP地图文件和地图的GNV文件，地图的制作使用的是官方的Hammer Editor，对于HE，本人了解甚浅，大家可以研究研究官方关于DOTA2的HE配置教程：
也可以咨询群中的HE大神：

materials文件夹，其中，overview为必须，定义的是DOTA2地图中的小地图文件，对于如何制作小地图，参考这篇文章：
制作小地图的工具，也可以在那篇文章中下载。
此外，在material文件夹中，也可以定义诸如maps/mapname.txt的文件夹，用以定义地图一些属性，诸如：地图中的战争迷雾，水流的相关信息等等。

resource文件夹，此文件夹中放置的是：1，语言文件，自定义图标(resource/flash3/images/spellicons)，等等。

scripts为MOD中的核心，对于他的说明，我将在下面继续。

此外，MOD对于文件架构的使用可谓是非常自由，全局的文件架构，可以参考解压之后的DOTA2ROOT，在我的理解中，只要放到同样位置的文件，无论是模型，音效，还是英雄图标，粒子特效，都可以不加声明，直接引用，而无需做导入等等繁琐工作。

而说到文件架构，则必须提到目前DOTA2的一个问题，目前的DOTA2，对于addons文件夹中的mod，只能够载入第一个mod，因此，在V社修复此问题之前，请保证你正在操作的mod文件夹，在addons文件夹中，名列第一。

第二、核心进程

接下来将会以Frota和frostivus为例，说说整个custom map的核心，scripts文件夹中的文件，并以此引申。
首先，让我们先关注：scripts/vscripts文件夹中的addon_init，从文件名中就可以看出，一个程序最先导入的应该就是本文件，在Frotra中：

```
function Dynamic_Wrap( mt, name )
    if Convars:GetFloat( 'developer' ) == 1 then
        local function w(...) return mt[name](...) end
        return w
    else
        return mt[name]
    end
end

-- Server side setting of a preset game mode
Convars:RegisterConvar('frota_mode_preset', nil, 'Set to the game mode you want to start
exclusively.', FCVAR_PROTECTED)
Convars:RegisterConvar('frota_ban_modes', nil, 'Set to modes banned on this server', FCVAR_PROTECTED)

-- Load Frota
```

```
require('json')         -- Json Library
require('util')         -- Utilitiy functions
require('frota')        -- Main frota framework
require('gamemodes')    -- Gamemode framework and small gamemodes/addons

-- Include gamemodes
require('gamemodes/tinywars')
require('gamemodes/rvs')
require('gamemodes/warlocks')
require('gamemodes/invokerwars')
require('gamemodes/puckwars')
require('gamemodes/plage')
require('gamemodes/sunstrikewars')
require('gamemodes/kaolinwars')
require('gamemodes/kunkkawars')

-- Include addons
require('gamemodes/fatometer')



print("\n\nDone Loading!\n\n")
```

在Frota的addoninit中，初始化了关于游戏模式的变量，并载入了其他lua的内容。

在Frotsivus中也是一样，引入了frostivus和AI：

```
-- This chunk of code forces the reloading of all modules when we reload script.
if g_reloadState == nil then
     g_reloadState = {}
     for k,v in pairs( package.loaded ) do
          g_reloadState[k] = v
     end
else
     for k,v in pairs( package.loaded ) do
          if g_reloadState[k] == nil then
               package.loaded[k] = nil
          end
     end
end

-- A function to re-lookup a function by name every time.
function Dynamic_Wrap( mt, name )
     if Convars:GetFloat( 'developer' ) == 1 then
          local function w(...) return mt[name](...) end
          return w
     else
          return mt[name]
     end
end

-- Going to store off some info when we precache units
UnitPrecacheData = {}

function PrecacheFrostivusUnit( name )
     if not UnitPrecacheData[name] then
          local unit = CreateUnitByName( name, Vec3(0,0,0), true, nil, nil, DOTA_TEAM_BADGUYS )
          if unit then
               UnitPrecacheData[name] = { xp = unit:GetDeathXP() }
               UTIL_RemoveImmediate( unit )
          end
     end
end

require( "frostivus" )
require( "frostivus_logging" )
require( "ai_core" )
```

一个RPG的运行，就是由addon_init开始的。

之后便是整个RPG的核心关键，Frota.lua和frostivus.lua
这两个文件中，注册了整个自定义模式的全体流程，几乎所有的命令和函数。
如果要以WE类比的话，这个就是WE中的触发器。

在Frota.lua中，定义了使用到的所有常量变量，游戏进程中，所有模式都需要用到的诸如applybuild这样的函数：找到

玩家的ID，如果玩家的ID有效，技能组合有效的话，那么就移除当前英雄的所有技能，添加上要添加的技能。
```
function FrotaGameMode:ApplyBuild(hero, build)
    -- Grab playerID
    local playerID = hero:GetPlayerID()
    if not self:IsValidPlayerID(playerID) then
        return
    end

    -- Make sure the build was parsed
    build = build or self.selectedBuilds[playerID].skills

    -- Remove all the skills from our hero
    self:RemoveAllSkills(hero)

    -- Give all the abilities in this build
    for k,v in ipairs(build) do
        -- Add to build
        hero:AddAbility(v)
        self.currentSkillList[hero][k] = v
    end
End
```
这个东西有什么用？除了做OMG以外，这个东西的意义就在于，我们可以完全不使用Hero_custom，单纯使用applybuild，就可以把我们的自定义技能赋给任何英雄。而Frota就是这么干的。
此外，再举个例子，如果有玩过GW2的玩家，或者看过视频的，那么应该对他的1技能三段砍有印象，或者剑灵中也有一些类似的设置。
使用这类函数，就可以模拟出他们的操作，使用一个技能，监听到，改变那个技能，三段之后，回复初始技能。

还有改变玩家当前英雄：changeHero函数
```
function FrotaGameMode:ChangeHero(hero, newHeroName)
    local playerID = hero:GetPlayerID()-- 找到玩家ID
    local ply = Players:GetPlayer(playerID)--赋给PLY本地变量
    if ply then
        -- Grab info
        local exp = hero:GetCurrentXP()--记住当前经验值
        local gold = hero:GetGold()--记住当前金钱

        local slots = {}
        for i=0, 11 do
            local item = hero:GetItemInSlot(i)
            if item then
                -- Workout purchaser
                local purchaser = -1
                if item:GetPurchaser() ~= hero then
                    purchaser = item:GetPurchaser()
                end

                -- Store the item
                slots[i] = {
                    purchaser = purchaser,
                    purchaseTime = item:GetPurchaseTime(),
                    currentCharges = item:GetCurrentCharges(),
                    StacksWithOtherOwners = item:StacksWithOtherOwners(),
                    sort = item:GetAbilityName()
                }

                -- Remove the item
                item:Remove()--记住所有物品
            end
        end

        -- Replace the hero
        local newHero = ply:ReplaceHeroWith(newHeroName, gold, exp)
        self:SetActiveHero(newHero)--置换英雄并设为活动

        -- Validate new hero
        if newHero then
            local blockers = {}

            -- Give items
            for i=0, 11 do
```

```
                local item = slots[i]
                if item then
                    local p = (item.purchaser == -1 and newHero) or item.purchaser
                    local it = CreateItem(item.sort, p, p)
                    it:SetPurchaseTime(item.purchaseTime)
                    it:SetCurrentCharges(item.currentCharges)
                    it:SetStacksWithOtherOwners(item.StacksWithOtherOwners)
                    newHero:AddItem(it)
                else
                    local it = CreateItem('item_blink', newHero, newHero)
                    newHero:AddItem(it)
                    table.insert(blockers, it)--转移所有物品
                end
            end

            -- Remove blocks
            for k,v in pairs(blockers) do
                -- Remove this blocker
                v:Remove()
            end

            -- Return their new hero
            return newHero
        end
    end
End
```

而在frostivus.lua中，非常具有参考价值的则是关于刷兵的程序编写：

```
function FrostivusGameMode:_spawnWave( unitData )
     if unitData.nUnitsSpawnedThisRound == 0 then
          print( "Started spawning " .. unitData.pszLabel )
     end

     --根据传递过来的unitdata，找到本次unit的SPAWNER的位置。
     local spawner = Entities:FindByName( nil, unitData.pszSpawnerName )



     --下面这是一条测试指令，如果说，刷怪点设置有误，就会显示错误信息
     if not spawner then
          Msg( "Failed to find spawner named \'" .. unitData.pszSpawnerName .. "\' for \'" ..
          unitData.pszNPCClassName .."\' \n" )
          return
     end

     for iUnit= 1,unitData.nUnitsPerSpawn do
          --判断是不是要刷一个精英怪 bIsChampion
          local bIsChampion = RollPercentage( unitData.nChampionChance )

          -- You can define a different NPC to spawn when a champion spawns, otherwise it will use an
          upgraded version of the same class
          --定义要刷怪的NPC的名字，pszNPCtoSpawn
          local pszNPCtoSpawn = unitData.pszNPCClassName
          if bIsChampion and unitData.pszChampionNPCClassName ~= "" then
               pszNPCtoSpawn  = unitData.pszChampionNPCClassName
          end

          --定义刷怪点 vSpawnLocation
          local vSpawnLocation = spawner:GetOrigin()
          if not unitData.bDontOffsetSpawn then
               vSpawnLocation = vSpawnLocation + RandomVector( 200 )  --加一个200范围的刷怪范围，避免
               卡怪。
          end

          --刷怪！ 刷怪的命令，是DOTA2的API CreateUnitByName（要刷怪的NPC的名字，刷怪的地点，寻找空
          地，阵营（DOTA2里面三个阵营 DOTA_TEAM_GOODGUYS/BADGUYS/NEUTRAL））
          --hscript CreateUnitByName( string name, vector origin, bool findOpenSpot, hscript, hscript,
          int team)
          --多参考https://developer.valvesoftware.com/wiki/List_of_Dota_2_Script_Functions上面关于
          DOTA2 API的内容
          local unit = CreateUnitByName( pszNPCtoSpawn, vSpawnLocation, true, nil, nil,
          DOTA_TEAM_BADGUYS )
          if unit then
               if unit:IsCreature() then
                    if bIsChampion then
                         --刷个BOSS
                         unit:CreatureLevelUp( ( unitData.nChampionLevel - 1 ) ) -- Difficulty is
                         handled in the OnNPCSpawn callback
                         unit:SetChampion( true )
                         local nParticle = ParticleManager:CreateParticle( "heavens_halberd",
                         PATTACH_ABSORIGIN_FOLLOW, pCreature )
                         ParticleManager:ReleaseParticleIndex( nParticle )
                         unit:SetModelScale( 1.1, 0 )
                         unitData.nChampionMax = unitData.nChampionMax - 1
                         if unitData.nChampionMax <= 0 then
                              unitData.nChampionChance = 0
                         end
                    elseif unitData.nCreatureLevel > 0 then
                         unit:CreatureLevelUp( ( unitData.nCreatureLevel - 1 ) ) -- Difficulty is
                         handled in the OnNPCSpawn callback
                    end

                    if unitData.bFaceSouth then
                         unit:SetForwardVector( Vec3( 0, -1, 0 ) )
                    end

                    if unitData.bBonusRound and unitData.flBonusTime > 0.0 then
                         unit:AddNewModifier( unit, nil, "modifier_kill", { duration =
                         unitData.flBonusTime } )
                    end
                    self.nGoldFromRound = self.nGoldFromRound + unit:GetGoldBounty()
                    self.nXPFromRound = self.nXPFromRound + unit:GetDeathXP()
               end

               if not unitData.bDontGiveGoal then
                    unit:SetInitialGoalEntity( Entities:FindByName( nil, unitData.pszWaypointName ) )
               end

               unitData.nUnitsSpawnedThisRound = unitData.nUnitsSpawnedThisRound + 1
               unitData.nUnitsCurrentlyAlive = unitData.nUnitsCurrentlyAlive + 1

               for iEnemyData=1,#self.vEnemiesRemaining do
                    local enemyDataTable = self.vEnemiesRemaining[iEnemyData]
                    if enemyDataTable.hEnemy == unit then
                         enemyDataTable.bCoreRoundEnemy = true
                         enemyDataTable.unitData = unitData

                         -- this is a core enemy, so XP is ok
                         unit:SetDeathXP( enemyDataTable.nDesiredXP )
                    end
               end
          end

          -- We spawned as many units as need to, this round of spawning is over.
          if unitData.nUnitsSpawnedThisRound >= unitData.nTotalUnitsToSpawnForRound then
               return
          end
     end
end
```
当然，frostivus的这个刷兵做得比一般刷兵要复杂一些，因为他的刷兵还要在其中随机刷出一个小BOSS什么的。
而如果要说有什么比刷兵过程更具有参考价值的话，那就是，frostivus.lua是如何一步一步调用到最后的刷兵过程的，他之前的判断条件是什么？何时刷兵？何时不刷兵？每次刷多少又是如何控制的？
大家可以以函数名一步一步向上搜索。

此外在lua的编写过程中，对于函数的定义是相当自由的。大家可以现在lua里面写一些内容，后来发现不够用了，再到后面再加一些内容，是没问题的。

第三、技能英雄物品编辑器
DOTA MOD中关于技能物品编辑器的设定，全在scripts/npc中。

1、herolist.txt

在herolist中，定义的是要在自定义地图中显示的英雄，例如frostivus：

```
//
// <key>      <value>
// Hero              currently on/off, will be # of instances (-1 = infinite)
//

"CustomHeroList"
{
     "npc_dota_hero_axe"                "1"
     "npc_dota_hero_windrunner"         "1"
     "npc_dota_hero_omniknight"         "1"
     "npc_dota_hero_nevermore"          "1"    // Shadow Fiend
     "npc_dota_hero_shadow_shaman"             "1"
     "npc_dota_hero_juggernaut"         "1"
     "npc_dota_hero_crystal_maiden"            "1"
     "npc_dota_hero_drow_ranger"        "1"
     "npc_dota_hero_venomancer"         "1"
     "npc_dota_hero_sven"               "1"
     "npc_dota_hero_jakiro"                    "1"
     "npc_dota_hero_magnataur"          "1"
     "npc_dota_hero_legion_commander"   "1"
     "npc_dota_hero_lina"               "1"
     "npc_dota_hero_puck"               "0"
     "npc_dota_hero_tidehunter"         "0"
     "npc_dota_hero_sand_king"          "1"
     "npc_dota_hero_kunkka"                    "0"
     "npc_dota_hero_elder_titan"        "1"
     "npc_dota_hero_storm_spirit"              "1"
     "npc_dota_hero_queenofpain"        "1"
     "npc_dota_hero_witch_doctor"       "1"
     "npc_dota_hero_templar_assassin"   "1"
}
```

要注意，在herolist.txt中，即使你在npc_hero_custom中定义了例如npc_dota_hero_templar_assassin_holdout来override原本的圣堂刺客，但是，在HeroList中，你依然要使用圣堂刺客的原名，否则不会正确显示。

此文件若放空，则在英雄选择界面，不会有任何英雄显示出来，当然，英雄选择不是必须，例如frota中就覆盖了英雄选择的过程，直接赋予玩家一个axe，这是后话。

2、npc_abilities_custom

这个文件夹中，定义的是所有在mod中要使用的技能，包括英雄的，普通单位的。

而对于技能，下面举几个例子;

```
"lesser_nightcrawler_pounce"
	{
		// General
		//-------------------------------------------------------------------------------------------------------------
		"AbilityName"					"lesser_nightcrawler_pounce"
		"AbilityBehavior"				"DOTA_ABILITY_BEHAVIOR_NO_TARGET"
		"AbilityUnitDamageType"			"DAMAGE_TYPE_MAGICAL"
		"AbilityTextureName"			"slark_pounce"

		// Time
		//-------------------------------------------------------------------------------------------------------------
		"AbilityCooldown"				"4.0"

		// Cost
		//-------------------------------------------------------------------------------------------------------------
		"AbilityManaCost"				"0"

		// Special
		//-------------------------------------------------------------------------------------------------------------
		"AbilitySpecial"
		{
			"01"
			{
				"var_type"					"FIELD_INTEGER"
				"pounce_distance"			"700"
		}
		"02"
		{
			"var_type"				"FIELD_FLOAT"
			"pounce_speed"			"933.33"
		}
		"03"
		{
			"var_type"				"FIELD_FLOAT"
			"pounce_acceleration"	"7000.0"
		}
		"04"
		{
			"var_type"				"FIELD_INTEGER"
			"pounce_radius"			"95"
		}
		"05"
		{
			"var_type"				"FIELD_INTEGER"
			"pounce_damage"				"35 50 65 80"
		}
		"06"
		{
			"var_type"				"FIELD_FLOAT"
			"leash_duration"		"3.5"
		}
		"07"
		{
			"var_type"				"FIELD_INTEGER"
			"leash_radius"			"325"
		}
	}
}
```

其中，最重要的内容是：BaseClass的内容，也就是定义这个技能的父类。
之后，对比下他的父亲：小鱼人的跳：

```
"slark_pounce"
	{
		// General
		//-------------------------------------------------------------------------------------------------------------
		"ID"					"5495"														// unique ID number for this ability.  Do not change this once established or it will invalidate collected stats.
		"AbilityBehavior"				"DOTA_ABILITY_BEHAVIOR_NO_TARGET"
		"AbilityUnitDamageType"			"DAMAGE_TYPE_MAGICAL"

		// Time
		//-------------------------------------------------------------------------------------------------------------
		"AbilityCooldown"				"20.0 16.0 12.0 8.0"

		// Cost
		//-------------------------------------------------------------------------------------------------------------
		"AbilityManaCost"				"75 75 75 75"

		// Special
		//-------------------------------------------------------------------------------------------------------------
		"AbilitySpecial"
		{
			"01"
			{
				"var_type"				"FIELD_INTEGER"
				"pounce_distance"		"700"
```

```
                }
                "02"
                {
                    "var_type"                          "FIELD_FLOAT"
                    "pounce_speed"                      "933.33"
                }
                "03"
                {
                    "var_type"                          "FIELD_FLOAT"
                    "pounce_acceleration"        "7000.0"
                }
                "04"
                {
                    "var_type"                          "FIELD_INTEGER"
                    "pounce_radius"                     "95"
                }
                "05"
                {
                    "var_type"                          "FIELD_INTEGER"
                    "pounce_damage"                     "70 140 210 280"
                }
                "06"
                {
                    "var_type"                          "FIELD_FLOAT"
                    "leash_duration"             "3.5"
                }
                "07"
                {
                    "var_type"                          "FIELD_INTEGER"
                    "leash_radius"                      "325"
                }
            }
        }
```

应该就可以看出对于自定义技能的最基本应用：改变数值。

第二个例子，来自cyborgmatt创作的invoer_meat_ball。

```
// Invoker Wars: Meat Ball
//
=========================================================================================
===============
"invoker_wars_chaos_meteor"
{
    // General
    //-------------------------------------------------------------------------------------------------------------
    "AbilityBehavior"                               "DOTA_ABILITY_BEHAVIOR_POINT |
    DOTA_ABILITY_BEHAVIOR_IGNORE_BACKSWING"
    "AbilityUnitDamageType"                         "DAMAGE_TYPE_MAGICAL"
    "BaseClass"                                             "invoker_chaos_meteor"
    "AbilityTextureName"                    "invoker_wars_chaos_meteor"

    // Stats
    //-------------------------------------------------------------------------------------------------------------
    "AbilityCastRange"                              "600"
    "AbilityCastPoint"                              "0"
    "AbilityCooldown"                               "30"
    "AbilityManaCost"                               "0"

    // Stats
    //-------------------------------------------------------------------------------------------------------------
    "AbilityModifierSupportValue"    "0.0"    // Mainly about damage

    // Special
    //-------------------------------------------------------------------------------------------------------------
    "AbilitySpecial"
    {
```

```
			"01"
			{
				"var_type"					"FIELD_FLOAT"
				"land_time"					"1.3"
			}
			"02"
			{
				"var_type"					"FIELD_INTEGER"
				"travel_distance"			"465 615 780 930"
			}
			"03"
			{
				"var_type"					"FIELD_INTEGER"
				"travel_speed"				"150"
			}
			"04"
			{
				"var_type"					"FIELD_INTEGER"
				"area_of_effect"		"250"
			}
			"05"
			{
				"var_type"					"FIELD_FLOAT"
				"damage_interval"			"0.5"
			}
			"06"
			{
				"var_type"					"FIELD_INTEGER"
				"vision_distance"			"500"
			}
			"07"
			{
				"var_type"					"FIELD_FLOAT"
				"end_vision_duration"	"3.0"
			}
			"08"
			{
				"var_type"					"FIELD_FLOAT"
				"main_damage"				"50 70 90 110"
			}
			"09"
			{
				"var_type"					"FIELD_FLOAT"
				"burn_duration"				"2.0"
			}
			"10"
			{
				"var_type"					"FIELD_FLOAT"
				"burn_dps"					"5 10 15 20"
			}
		}
	}
```

对比卡尔的原版技能：

```
"invoker_chaos_meteor"
	{
		// General
		//-------------------------------------------------------------------------------------------------------------
		"ID"					"5385"
						// unique ID number for this ability.  Do not change this once established or it will invalidate collected stats.
		"AbilityBehavior"				"DOTA_ABILITY_BEHAVIOR_POINT | DOTA_ABILITY_BEHAVIOR_HIDDEN | DOTA_ABILITY_BEHAVIOR_NOT_LEARNABLE | DOTA_ABILITY_BEHAVIOR_IGNORE_BACKSWING"
```

```
"MaxLevel"                                      "1"
"HotKeyOverride"                         "D"
"AbilityUnitDamageType"                  "DAMAGE_TYPE_MAGICAL"

// Stats
//------------------------------------------------------------------------------------------
-------------------
"AbilityCastRange"                       "700"
"AbilityCastPoint"                       "0"
"AbilityCooldown"                        "55"
"AbilityManaCost"                        "200"

// Stats
//------------------------------------------------------------------------------------------
-------------------
"AbilityModifierSupportValue"     "0.0"  // Mainly about damage

// Special
//------------------------------------------------------------------------------------------
-------------------
"AbilitySpecial"
{
     "01"
     {
          "var_type"                                "FIELD_FLOAT"
          "land_time"                               "1.3"
     }
     "02"
     {
          "var_type"                                "FIELD_INTEGER"
          "travel_distance"                  "465 615 780 930 1095 1245 1410"
          "levelkey"                                "wexlevel"
     }
     "03"
     {
          "var_type"                                "FIELD_INTEGER"
          "travel_speed"                            "300"
     }
     "04"
     {
          "var_type"                                "FIELD_INTEGER"
          "area_of_effect"                   "275"
     }
     "05"
     {
          "var_type"                                "FIELD_FLOAT"
          "damage_interval"                  "0.5"
     }
     "06"
     {
          "var_type"                                "FIELD_INTEGER"
          "vision_distance"                  "500"
     }
     "07"
     {
          "var_type"                                "FIELD_FLOAT"
          "end_vision_duration"              "3.0"
     }
     "08"
     {
          "var_type"                                "FIELD_FLOAT"
          "main_damage"                      "57.5 75 92.5 110 127.5 145 162.5"
          "levelkey"                                "exortlevel"
     }
     "09"
     {
          "var_type"                                "FIELD_FLOAT"
          "burn_duration"                           "3.0"
     }
     "10"
     {
          "var_type"                                "FIELD_FLOAT"
          "burn_dps"                                "11.5 15 18.5 22 25.5 29 32.5"
          "levelkey"                                "exortlevel"
     }
```

```
          }
      }
```

对于这个技能，应该主要关注以下几点：
1、如何自定义一个英雄图标？
2、如何使用另一个技能的等级来决定本技能的等级？
3、关于DOTA_ABILITY_BEHAVIOR_HIDDEN的应用。

再之后，则是一个完完全全彻头彻尾的自定义技能的例子：

```
      // Ability: Summon Undead
      //
      ===================================================================================================
      ================
      "creature_summon_undead"
      {
            // General
            //-------------------------------------------------------------------------------------------------
            --------------------
            "BaseClass"                                    "ability_datadriven"
            //使用一个空技能作为技能类
            "AbilityBehavior"                        "DOTA_ABILITY_BEHAVIOR_NO_TARGET | 
            DOTA_ABILITY_BEHAVIOR_CHANNELLED | DOTA_ABILITY_BEHAVIOR_DONT_RESUME_ATTACK"
            //定义技能类型
            "AbilityTextureName"                     "undying_soul_rip"
            //自定义技能图标//这里的图标调用的是官方图标//上面卡尔调用的是自定义图标，他们都是不需要经过
            声明，就可以直接使用的。
            "precache"
            {
                  "particlefile"                           "particles/generic_aoe_persistent_circle_
                  1.pcf"
                  "particlefile"
                        "particles/econ/generic/generic_aoe_explosion_sphere_1.pcf"
                  "soundfile"
                        "scripts/game_sounds/ability_summon_undead.txt"
            }
            //定义预载入的粒子特效，音效

            // Casting
            //-------------------------------------------------------------------------------------------------
            --------------------
            "AbilityCastRange"                       "0"
            "AbilityCastPoint"                       "0"
            "AbilityChannelTime"               "2.2 2.1 2.0 1.9"
            "AbilityCastAnimation"                   "ACT_DOTA_VICTORY"
            //定义技能属性，技能动画

            // Time
            //-------------------------------------------------------------------------------------------------
            --------------------
            "AbilityCooldown"                        "10.0 10.0 10.0 10.0"
            //定义冷却时间

            // Cost
            //-------------------------------------------------------------------------------------------------
            --------------------
            "AbilityManaCost"                        "100 100 100 100"
            //定义技能的蓝耗

            "OnSpellStart"
            {
                  "ApplyModifier"
                  {
                        "Target"                           "CASTER"
                        "ModifierName"                     "channel_started"//这个channel started的
                        modifier，在下面有定义，具体应用可以理解为，如何在一个释放技能的单位上，绑上特定的
                        粒子特效。
                  }

                  "FireSound"
```

```
		{
			"EffectName"				"Ability.SummonUndead"
			"Target"						"CASTER"
		}
	}

	"OnChannelSucceeded"
	{
		"SpawnUnit"
		{
			--当技能释放成功，便召唤单位:
			"UnitName"				"npc_dota_creature_berserk_zombie"
			"UnitCount"				"%number_of_zombies"
			"UnitLimit"				"32"
			"SpawnRadius"		"175"
			"Target"				"CASTER"
			"OnSpawn"
			{
				"AttachEffect"
				{
					"EffectName"		"generic_aoe_explosion_sphere_1"
					"EffectAttachType"	"follow_origin"
					"Target"					"TARGET"

					"ControlPoints"
					{
						"00"		"0 0 0"
						"01"		"50 100 0"
						"02"		"4 10 .5"
						"03"		"20 200 0"
						"04"		"0 0 0"
						"05"		"0 0 0"
					}
				}

				"FireSound"
				{
					"EffectName"		"Ability.SummonUndeadSuccess"
					"Target"					"CASTER"
				}
			}
		}
	}

	"OnChannelFinish"
	{
		"RemoveModifier"
		{
			"Target"						"CASTER"
			"ModifierName"					"channel_started"
		}
	}

	// Modifiers
	//-----------------------------------------------------------------------------------------------------------------------
	"Modifiers"
	{
		"channel_started"
		{
			"OnCreated"
			//在创造这个Modifier的时候，创造一个粒子特效，绑定到Caster的Follow_origin上
			{
				"AttachEffect"
				{
					"EffectName"		"generic_aoe_persist_summon_1"
					"EffectAttachType"	"follow_origin"
					"Target"					"CASTER"
				}
			}
		}

		"summoned_units"
		{
```

```
                "AttachEffect"
                {
                    "EffectName"        "leshrac_split_earth"
                    "EffectAttachType"  "follow_origin"
                    "Target"                "TARGET"

                    "ControlPoints"
                    {
                        "00"         "0 0 0"
                        "01"         "200 0 0"
                        "02"         "0 0 0"
                    }
                }
            }
        }

        // Special
        //--------------------------------------------------------------------------------------------------------------
        "AbilitySpecial"
        {
            "01"
            {
                "var_type"                      "FIELD_INTEGER"
                "number_of_zombies"         "8"
            }
        }
    }
```

3，关于自定义物品npc_items_custom

第一个例子来自frostivus的大净化：greater clarity

// Greater Clarity
//
===
"item_greater_clarity"
{

// General
//---
"ID"   "1004"
//首先，关于ID，每个物品从目前对于items_custom的研究来说，他的ID并非必须，那究竟哪些地方是需要ID的，哪些地方是不需要ID的，还需要进行归类研究，找出他们的本质区别。
"AbilityBehavior"   "DOTA_ABILITY_BEHAVIOR_UNIT_TARGET | DOTA_ABILITY_BEHAVIOR_IMMEDIATE | DOTA_ABILITY_BEHAVIOR_DONT_RESUME_ATTACK"
//AbilityBehavior方面，这个与英雄技能方面并无二致
"AbilityUnitTargetTeam"   "DOTA_UNIT_TARGET_TEAM_FRIENDLY"
"AbilityUnitTargetType"   "DOTA_UNIT_TARGET_HERO"
//定义只能对友方英雄使用，当然，这个字段对于英雄技能，同样有效。
"Model"   "models/props_gameplay/salve_blue.mdl"
//这个Model定义的是物品被丢到地上之后的模型，这个模型可以是自定义模型，也可以是官方模型，使用方法是一样的。
"BaseClass"   "item_datadriven"
//基础类，和上面的Summon Undead一样，使用的是item类的空技能。
"AbilityTextureName"   "item_greater_clarity"
//定义物品图标
"ItemKillable"   "0"
//定义物品能否被丢地上A掉~在WE里面一般是用给物品加一个非常高的血量实现的。

// Stats
//---
"AbilityCastRange"   "100"
"AbilityCastPoint"   "0.0"

// Item Info
//---
"ItemCost"   "90"

```
"ItemShopTags"                                  "consumable"
"ItemQuality"                                   "consumable"
"ItemStackable"                                 "1"
"ItemShareability"                              "ITEM_FULLY_SHAREABLE_STACKING"
"ItemPermanent"                                         "0"
"ItemInitialCharges"                    "1"
"SideShop"                                              "1"
//这是在物品上独有的，物品花费，归属类型，能否村粗，能否共享，能否在边路商店购买，等等。
//对于一个单纯完全自定义的技能，则需要完全定义他的所有属性，否则，如同上面的英雄技能一样，对
于和父类一样的属性，不用重复定义，能够完全继承。

"OnSpellStart"
{
	"ApplyModifier"
	{
		"Target" "CURSOR_TARGET"
		"ModifierName" "modifier_item_greater_clarity"
		//和英雄技能一样，以ApplyModifier的方式，赋予持续类技能效果，具体Modifier，查看
		Modifiers中的定义：
	}
	"FireSound"
	{
		"Target" "UNIT"
		"EffectName" "DOTA_Item.ClarityPotion.Activate"
	}
	"SpendCharge"
	{}
}

"Modifiers"
{
	"modifier_item_greater_clarity"
	{
		"TextureName" "item_greater_clarity"
		//这是状态栏图标
		"EffectName" "healing_clarity"
		//这是状态的动画
		"EffectAttachType" "follow_origin"
		//动画绑定点
		"Duration" "%buff_duration"
		//持续时间
		"Properties"
		{
			"MODIFIER_PROPERTY_MANA_REGEN_CONSTANT" "%mana_per_tick"
			//MODIFIER_PROPERTY_MANA_REGEN_CONSTANT,请到V社官方网站查看MODIFIER大全，
			或者，对于你想使用的技能属性，也可以参考类似的技能来编写。
		}
		"OnTakeDamage"
		{
			"RemoveModifier"
			{
				"Target" "UNIT"
				"ModifierName" "modifier_item_greater_clarity"
			}
		}//定义受到攻击伤害之后移除回复Modifier
	}
}

// Special
//-------------------------------------------------------------------------------------------------------------
"AbilitySpecial"
```

```
		{
			"01"
			{
				"var_type"						"FIELD_INTEGER"
				"buff_duration"				"10"
			}
			"02"
			{
				"var_type"						"FIELD_INTEGER"
				"total_mana"					"150"
				//这个Total_mana，在技能中并没有使用过，也一样可以写出来，方便以后的修改参考。
			}
			"03"
			{
				"var_type"						"FIELD_INTEGER"
				"mana_per_tick"					"15"  // %total_mana / %buff_duration
			}
		}
		//定义了在Modifier中使用的变量数值。归出来定义，可以方便修改。
	}
```

接下来让我们介绍一下DOTA2中关于物品合成：

```
	// Recipe: Arcane Boots
	//
=========================================================================================================
	"item_recipe_arcane_boots2"
	{

		"BaseClass"							"item_recipe_arcane_boots"

		// Item Info
		//-----------------------------------------------------------------------------------------------------
		"ItemCost"							"0"
		"ItemShopTags"						""

		// Recipe
		//-----------------------------------------------------------------------------------------------------
		"ItemRecipe"						"1"
		"ItemResult"						"item_arcane_boots2"
		"ItemRequirements"
		{
			"01"							"item_energy_booster;item_arcane_boots"
		}
	}
	//如果玩过frostivus的话，应该就会知道，大秘法鞋，是由一个秘法鞋+一个蓝球合成的，为何又来一个卷轴？
	//那就要涉及到DOTA2的物品合成系统了，我们来看一个真正需要卷轴合成的物品，夜叉：
		// Recipe: Yasha
		"item_recipe_yasha"
		{
			// General
			//-------------------------------------------------------------------------------------------------
			"ID"							"169"							// unique ID number for this item.  Do not change this once established or it will invalidate collected stats.

			// Item Info
			//-------------------------------------------------------------------------------------------------
			"ItemCost"						"600"
			"ItemShopTags"					""

			// Recipe
		//-------------------------------------------------------------------------------------------
		------------------------------
		"ItemRecipe"					"1"
		"ItemResult"					"item_yasha"
		"ItemRequirements"
		{
			"01"
				"item_blade_of_alacrity;item_boots_of_elves"
		}
	}
```
通过对比，那么我想，非常容易就可以发现，他们的定义，在本质上，几乎是完全一样的。
唯一的区别就是，夜叉的卷轴有价格，大秘法鞋的卷轴价格是0而已。
也就是说，对于任何一个合成物品，在DOTA2中，都会需要定义他的合成卷轴，在合成卷轴中，定义他的所需材料和最终成品。

4、自定义英雄和自定义单位。
把自定义英雄和自定义单位放到一起说，是因为除了几个关键的字段以外，他们两个可以算得上是毫无分别：
```
HeroCustom
// HERO: Zuus
"npc_dota_hero_zuus_holdout"
{
	"override_hero"			"npc_dota_hero_zuus"
	"Ability1"				"holdout_arc_lightning"
	"Ability2"				"holdout_lightning_bolt"
	"Ability3"				"holdout_static_field"
	"Ability4"				"zuus_thundergods_vengeance"
	//覆盖四个技能
	"VisionNighttimeRange"		"1800"
	//覆盖夜间视野范围
}
```
而其他的方面，英雄模型什么的，则完全继承他所覆盖的npc_dota_zuus，无需再做定义。

而即使是比较复杂的，定义那个可以分裂的石头人，其实也一样简单
```
	//=================================================================================
		// Splitter A
		//=================================================================================
		"npc_dota_splitter_a"
		{
			// General
			//
			"BaseClass"					"npc_dota_creature"	// Class of entity
			of link to.
			"Model"
				"models/creeps/neutral_creeps/n_creep_golem_b/n_creep_golem_b.mdl"	//
			Model.
			"ModelScale"				"1.2"
			"Level"							"1"
			"CanBeDominated"			"0"

			// Abilities
			//----------------------------------------------------------------
			"Ability1"					"creature_split_a"				//
			Ability 1.
```
之后是他的分裂技能：
```
	//
	=============================================================================================
	=====================
		"creature_split_a"
		{
			// General
			//-------------------------------------------------------------------------------------------
			------------------------------
			"BaseClass"					"ability_datadriven"
			"AbilityBehavior"				"DOTA_ABILITY_BEHAVIOR_PASSIVE"
			"AbilityTextureName"			"dark_seer_wall_of_replica"

			"precache"
			{
				"particlefile"					"particles/creature_splitter.pcf"
```

```
                "soundfile"
                        "scripts/game_sounds/ability_split.txt"
        }
```

这个技能中的核心是：

```
"OnOwnerDied"//当技能所有者死亡之后触发
     {
          "FireEffect"
          {
               "Target"                          "CASTER"
               "EffectName"              "splitter_a"
               "EffectAttachType"        "follow_origin"
          }

          "FireSound"
          {
               "EffectName"              "Ability.SplitA"
               "Target"                          "CASTER"
          }

          "SpawnUnit"//产生三个新的单位
          {
               "UnitName"                        "npc_dota_splitter_b"
               "UnitCount"                       "3"
               "SpawnRadius"             "50"
               "Target"                          "CASTER"
               "GrantsGold"              "1"
               "GrantsXP"                        "1"
               "OnSpawn"
               {
                    "Knockback"//在新单位诞生的时候，Knockback
                    {
                         "Target"             "TARGET"
                         "Center"             "CASTER"
                         "Duration"           "0.75"
                         "Distance"           "275"
                         "Height"             "200"
                    }
               }
          }
     }
}
```

继续回到unit定义他的分裂之后的单位：

```
"npc_dota_splitter_b"
     {
          // General
          //
          "BaseClass"                              "npc_dota_creature"  // Class of entity
          of link to.
          "Model"
                 "models/creeps/neutral_creeps/n_creep_golem_b/n_creep_golem_b.mdl"   //
          Model.
          "ModelScale"                     "0.7"
          "Level"                                        "1"
```

改变也很简单，无非是将ModelScale从1.2改为0.7，之后再在下面修正了下他们的攻击力等等数值而已。

此外，关于自定义音效：
依然举例：
在abitlities_custom中，关于summon_undead有这么一个定义：

```
"soundfile"                          "scripts/game_sounds/ability_summon_undead.txt"
```

那么，这个ability_summon_undead.txt，就存放在scripts/game_sounds文件夹中，全文如下：

```
"Ability.SummonUndead"
{
     "channel"            "CHAN_WEAPON"
     "volume"             "1"
     "soundlevel"         "SNDLVL_90dB"
     "pitch"                      "95, 105"
     "wave"               ")weapons/hero/skeleton_king/mortal_strike_cast.wav"
```

```
}

"Ability.SummonUndeadSuccess"
{
      "channel"              "CHAN_STATIC"
      "volume"               "1.0"
      "soundlevel"                  "SNDLVL_81dB"
      "pitch"                       "90, 100"
      "rndwave"
      {
            "wave"        "weapons/hero/death_prophet/exorcism_impact01.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact02.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact03.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact04.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact05.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact06.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact07.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact08.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact09.wav"
            "wave"        "weapons/hero/death_prophet/exorcism_impact10.wav"
      }
```
注意rndwave的用法。

```
}
```
这是技能音效的用法。

而对于环境音效，则在scripts/soundscape_frostivus.txt中有定义，例如整个夜晚时不时响起一声狼叫：
```
"random.wolf.call"
{
      "playrandom"
            {
            "volume"        ".6,.9"
            "pitch"                "90,100"
            "time"          "17,39"
            "position"      "random"

            "rndwave"
            {
                  "wave" "ambient/soundscapes/diretide/diretide_wolf_01.wav"
                  "wave" "ambient/soundscapes/diretide/diretide_wolf_02.wav"
                  "wave" "ambient/soundscapes/diretide/diretide_wolf_03.wav"
                  "wave" "ambient/soundscapes/diretide/diretide_wolf_04.wav"

            }
      }
}
```
注意，这里wave调用的，是官方文件ambient中的声音文件，而不是自定义的文件。

而如果是自定义的声音文件呢？
```
"random.disquiet"
{



      "playrandom"
      {
            "volume"        ".3,.6"
            "pitch"                "90,110"
            "time"          "16,29"
            "position"      "random"

            "rndwave"
            {
                  "wave" "ambient/disquiet01.wav"
                  "wave" "ambient/disquiet02.wav"
                  "wave" "ambient/disquiet03.wav"
            }
      }
```
也是一样直接调用。

OK，时间不早，就写这么多吧，粗浅之言，希望能够对大家有所帮助。