ПРО СОЗДАНИЕ СОБСТВЕННОГО СЦЕНАРИЯ НУ И ВООБЩЕ ПРО ВСЯКОЕ.

Тема в разделе 'Серп и Молот - Обсуждение Модификаций Игры', создана пользователем Novik, 12 май 2006.

Novik
Guest

Один раз (правда, применительно к S^3) что-то такое уже писал, т.к. форум на silest.ru сдох вместе с сайтом желающие могут ознакомиться с предыдущим бессмертным творением на страничке mp5k, раздел "Редактор" - http://mp5k.nightmail.ru/
Решил продолжить данную традицию.

Про сценарий.
В отличие от предшественников конструирование сценария в СиМ практически целиком построено на скриптах и не требует привлечения к/л дополнительного софта за исключением редактора.
Я предполагаю, что читающие данное описание вполне освоили создание и скриптование отдельных шаблонов – здесь речь пойдет в основном только о том, как связать эти шаблоны в единый сценарий.
Для простоты дела я полагаю, что работы по созданию собственного сценария ведутся поверх сценария существующего. Это вовсе не значит, что в Ваш сценарий каким-то образом будет связан со сценарием оригинальной игры, данный образ действий просто избавляет от выполнения массы рутинной работы, которая уже произведена в оригинале.

Сценарные зоны.
Единицей построения сценария является сценарная зона. Это некая надстройка над готовым шаблоном. Редактирование и создание новых сценарных зон осуществляется на вкладке редактора ScenarioZones.
(Соответствующая таблица в варианте прямого доступа к БД называется DBScenarioZones).
Поля к заполнению:

Scenario – ссылка на сценарий, к которому относится сценарная зона. Смело ставим сценарий оригинальной игры – Main.
TemplateID1 – ссылка на шаблон, над которым «надстраивается» сценарная зона.
PWLImageID – ссылка на текстуру PWL (Please wait, loading), которая будет показываться при загрузке сценарной зоны. Обычно это стилизованный скриншот шаблона. Если не заполнять данное поле, то будет показан стандартный синий экран с лого игры.
SmallDescription – имя сценарной зоны для ссылки на нее из скриптовых функций. Должно содержать только латинские буквы и цифры.
Name – имя сценарной зоны для использования на интерфейсе (например, в журнале)
Остальные поля не существенны и могут быть оставлены с умолчательными значениями.

Забавный факт – из одного и того же шаблона можно соорудить несколько сценарных зон. Текущее состояние (наличие разрушений, произведенных игроком, разнообразный лут и т.п.) сохраняется в сейве применительно к сценарной зоне, а не к шаблону. Возможно, с этой особенностью можно как-то пооригинальничать.

Перемещение игрока между сценарными зонами
Осуществляется скриптовыми функциями

BeginZone(zoneName)

forceZone(zoneName)
forseSubZone(zoneName)

Здесь zoneName – это значение поля SmallDescription, которое Вы указали при заведении сценарной зоны. По вызову функции BeginZone немедленно будет загружена требуемая сценарная зона и стартует скрипт соответствующего ей шаблона. Замечу – при отладке в редакторе эта, да и прочие функции, так или иначе затрагивающие сценарий, работать не будут, т.к. сценария там нет.
Для того, чтобы пояснить, зачем необходимы оставшиеся две функции, требуется небольшое отступление. Итак, по вызову BeginZone требуемая зона немедленно загрузится, но существует небольшая неприятность – состояние текущей зоны, из скрипта которой был произведен данный вызов, сохранено не будет. Т.е. все изменения, произведенные игроком в интерактивном окружении (разрушение тех или иных зданий, объектов и т.п.) будет потеряны, и в следующий раз эта зона стартует такой же, как была «до первого визита». В случае если зоны одноразовые (как в S^2 и S^3) в этом нет ничего страшного. Но в СиМ зоны – повторновходимы, и сохранять их состояние таки хотелось бы. К сожалению, к моменту, когда я получил доступ к исходникам движка, данная проблема мной была уже решена. Поэтому никакой скриптовой функции типа SaveCurrentZone в API нет. Есть более хм… другой способ, реализованный, я бы сказал, «через задницу». Сохранение состояния текущей зоны производится при выходе игрока с текущей зоны на карту чаптера (или на глобальную карту). Соответственно, при вызовах скриптовых функций ExitToChapter и ExitToGlobal. Но в данном случае нам не нужно на глобальную карту (и тем паче на карту чаптера, визуальное представление которой в СиМ вообще отсутствует). Решение проблемы очевидно – в функции, вызываемой из скрипта для перехода мы не используем немедленно вызов BeginZone, а просто устанавливаем значение некоторой глобальной переменной в zoneName. После чего даем команду выхода на чаптер или на глобальную карту. А уже в скриптах глобальной карты (forceZone) или чаптера (forseSubZone) производим анализ этой переменной, если она не пуста – обнуляем ее и вызываем BeginZone с необходимым параметром.
Разница между forceZone и forseSubZone чисто косметическая – после покидания текущей зоны но до момента срабатывания BeginZone игроку будет на короткое время демонстрироваться глобальная карта, либо карта чаптера. Я предпочитаю пользоваться вторым вариантом, оформляя карту чаптера в виде PWL текущей зоны. Таким образом и волки сыты, и овцы целы. При должном оформлении скриптов чаптеров Вам, в принципе, не нужно заботиться обо всем вышеперечисленном – весь этот алгоритм уже «зашит» внутри стандартных функций.
На самом деле полный прототип функций forceZone и forseSubZone выглядит немного иначе, так:

forceZone(zoneName,deployParam = nil,direction = 8)
forceSubZone(zoneName,deployParam = nil,direction = 8)

deployParam – это «добавка» к имени вейпоинтов, по которым будет расставлена партия игрока после входа на зону. Например, deployParam = ”_1” приведет к тому, что партия буде расставлена по вейпоинтам с именами UnitHero_1, Unit1_1, Unit2_1… По умолчанию пусто, т.е. партия расставляется по вейпоинтам UnitHero, Unit1, Unit2… или, в случае их отсутствия – по месту шаблона Deploy.
Direction – направление, куда будут повернуты юниты, от 1 до 7.
Данные параметры актуальны при повторных заходах на локацию, когда место размещения партии определяется скриптом вызывающей зоны.

Оформление карт чаптеров.
Карта чаптера, как и сам чаптер в СиМ – вещь «неинтерфейсная» и используется только для обеспечения механики перехода между зонами. Особо не вдаваясь в подробности, сформулирую ряд э-э-э, правил «как сделать так, чтобы все работало».
1) Каждой сценарной зоне, для которой имеется возможность посещения игроком путем тырканья в область на глобальной карте, должен соответствовать чаптер.
2) Этот чаптер должен иметь в качестве карты (поле Background) PWL соответствующей зоны.
3) Этот чаптер должен иметь в качестве PWL (поле PWLImageID) PWL соответствующей зоны.
4) Зона должна быть помещена на чаптер (в произвольное место). Саму процедуру помещения описывать не буду – в принципе, она похожа на описанное ниже создание сектора на глобальной карте. Тип сектора должен быть Zone.
5) Этот чаптер должен иметь скрипт следующего содержания:
checkGlobalMap()
BeginZone(zoneName)
где zoneName – имя соответствующей чаптеру зоны.
Вся необходимая работа производится на вкладке редактора Chapters. В принципе, одному чаптеру может соответствовать несколько сценарных зон, но грузиться при переходе с глобальной карты будет только одна - та, zoneName которой Вы указали в скрипте чаптера. Переход на прочие зоны Вы должны обеспечивать самостоятельно в скриптах зон (путем использования областей перехода, реакции на открытие всяческих люков и т.п.)

Оформление глобальной карты.
Редактирование и создание глобальной карты осуществляется на вкладке редактора GlobalMaps.
Поля к заполнению:

Scenario – ссылка на сценарий, к которому относится сценарная зона. Смело ставим сценарий оригинальной игры – Main.
StartZoneID – ссылка на самую первую в Вашем сценарии сценарную зону.
Background – ссылка на текстуру глобальной карты.
ScriptID – скрипт глобальной карты. Рекомендую выставить в Scripts\Sickle\Global.

Чтобы привязать чаптер (ну или сценарную зону, т.к. как уже рассказывал ранее каждой «показабельной» сценарной зоне должен соответствовать чаптер) к месту на глобальной карте, нужно
1) Нажать ПКМ в желаемом месте карты, и выбрать пункт меню New Sector.
2) Ответ на вопрос о количестве сегментов, которое оно задаст, зависит от того, какой картинкой у Вас будет отображаться чаптер на глобальной карте. В случае СиМ это квадратик, посему 4.
3) Таская мышью за «узлы» обровнять форму и поправить позицию созданного сектора.
4) Нажать ПКМ при положении курсора на границе сектора. Если попали – появится меню, выбрав в котором пункт Sector Properties можно будет указать свойства чаптера. Это
Chapter – тыркаем в кнопку, выбираем оформленный чаптер
Description – строка, содержащая zoneName. Нужна для связи со скриптом, который будет показывать время на перемещение в данную зону. За подробностями смотреть функцию skipTravelTime в файле scripts\s3Sickle.l. Чтобы время таки рассчитывалось, данную функцию нужно скорректировать, внеся в нее zoneName.
Image – фактически, это совсем даже не image, а ссылка на интерфейс, который определяет, что за значок будет рисоваться на этом месте, где будет поясняющая надпись, что там будет написано и т.п. Предварительно нужно все это хозяйство создать на загладке редактора Interface.
Image Pos – координаты картинки из предыдущего пункта относительно верхнего левого угла карты. Определяет место, где будет показана «подсвечивающая» картинка при помещении мыши в область сектора. Крайне неудобная вещь для редактирования – фактически, координаты нужно подбирать, время от времени смотря – а как оно в игре.

Передача данных между скриптами зон.
Без этого, понятно, никуда. Т.к. всегда полезно знать, чего там предпринял игрок на предыдущей зоне. Осуществляется это с помощью механизма глобальных переменных, базовые примитивы выглядят так:

void SetGlobalGameVar(sVarName,sVarValue)
string GetGlobalGameVar(sVarName,sDefVarValue)

соответственно, установить переменную в то или иное значение, и получить ее значение (defVarValue – значение по умолчанию, будет возвращаться в случае, если значение переменной не определено, т.е. SetGlobalGameVar для нее ни разу не вызвано).
Можно, конечно, пользовать данные примитивы, но на мой взгляд удобнее использовать «врапперы» над ними, которые несколько облегчают жизнь. На их реализацию можно посмотреть в файле scripts\s3Sickle.l.
Это

void putTimeMarker(sMarker, nHours)
bool isTimeExpired(sMarker, bDefault)
void putMarker(sMarker, nFlag)
number getMarker(sMarker)
void incMarker(sMarker, nDelta)

putTimeMarker заводит «будильник» с именем sMarker, который сработает через nHours часов.
isTimeExpired позволяет получить состояние «будильника» - сработал ли. Если не был заведен, то вернет bDefault.
putMarker выставляет числовой флаг.
getMarker, соответственно, позволяет его получить.
incMarker – увеличивает значение флага на nDelta. Или уменьшает – от знака nDelta зависит.

Так же имеется общий блок глобальных переменных, объединенных функциями initVariables, getVariables и putVariables.
Фактически это отображение глобальных переменных на локальные переменные скрипта, т.к. писать
GVCarma = GVCarma+1
гораздо удобнее, чем последовательность
local GVCarma = GetGlobalGameVar(”carma”,”0”)

GVCarma = GVCarma+1
SetGlobalGameVar(”carma”, GVCarma)
initVariables вызывается один раз, на старте сценария, и присваивает начальные значения всем переменным блока.
getVariables вызывается на старте скрипта зоны (если Вы используете стандартный «скелет» скрипта зоны, описанный ниже, то это происходит автоматически) и «перебрасывает» значения глобальных переменных в локальные.
putVariables вызывается при выходе с зоны и производит операцию, обратную getVariables.
Чтобы добавить в блок собственную переменную, нужно откорректировать все три функции по имеющемуся в их теле образцу.

Оформление скриптов зон.
Стандартный «скелет» скрипта зоны выглядит следующим образом:

Код:

```
    OnEnterZonePrim( "StartCamp", ZT_FOREST, AT_DUST )
    if(zoneEntered>1) then
    -- здесь какая-то инициализация для второго и последующих заходов на зону
    end
end

-- Покидаем зону. (Игрок нажал кнопку «Выход»)
-- Умолчательная реализация уже есть в файле scripts\s3Sickle.l,
-- если на финише ничего делать не надо, то данный перехватчик можно не
-- писать
function OnRealExit()
    -- какая-то финализация перед затемнением экрана
    OnRealExitPrim()
    -- какая-то финализация после затемнения экрана
    ExitToGlobal()
end

-- первый раз зашли на зону
DelayGameStart()
--
-- параметры:
-- zonename – имя данной зоны для рассчета времени перехода на нее
-- zone type - (ZT_FOREST,ZT_CITY,ZT_CAVE) тип зоны (лесная, городская,
-- пещера) для выставления эффектов освещения и обработки хайда
-- zone subtype (по умолчанию AT_NORMAL, может быть AT_DUST, AT_SPECIAL,
-- AT_SPECIAL1 or AT_SPECIAL2 для всяческих эффектов освещения и тумана)
startStandardZone( "StartCamp", ZT_FOREST, AT_DUST )
-- инициализация, не видимая игроку (корректируем уровень врагов и т.п)
initEnemyPlayer(1)
StartGame()
AutoSave()
-- инициализация, видимая игроку (ролик)
```

1 человеку нравится это.

Novik, 12 май 2006 #1

Novik
Guest

Продолжая про скриптовые функции работы со сценарием.

void BeginScenario(nGlobalID, sStartZoneName = nil)

позволяет в произвольном месте "зарядить" новый сценарий, поменяв глобальную карту. Параметры, соответственно, ID глобальной карты и имя стартовой сценарной зоны (если пусто, будет использована стартовая зона из свойств новой глобальной карты).
Состав партии будет сохранен, но все значения глобальных переменных будут потеряны.

void BeginTemplate(nTemplateID)

стартует указанный шаблон. Очень похоже на BeginZone, но "повторновходимость" в этом случае не реализовать.

void ExitToGlobal()

выход на глобальную карту с сохранением состояния текущей зоны.

void ExitToChapter()

выход на карту текущего чаптера с сохранением состояния текущей зоны. Если текущий чаптер отсутствует, т.е. игрок попал на зону минуя чаптер, будет произведен выход на глобальную карту.

void AddJournalRecord(nJournalRecordID)

Заносит в журнал запись с соотв. ID если таковой там еще нет.

bool AddJournalRecordEx(nJournalRecordID, sZoneName, param1, param2, ...)

Заносит в журнал запись с соотв. ID если таковой там еще нет. Кроме прочего позволяет передать ряд параметров для формирования текста записи (если шаблон записи оформлен должным образом). Второй параметр - строка, которая будет вынесена в качестве заголовка записи на "левую половину" журнала. Обычно это имя текущей зоны, получаемое с помощью функции ScenarioGetCurrentZoneName.
Если такая запись уже есть - вернет false.
См. так же враппер для данной функции в файле внешнего скрипта - toJournal.

void ScenarioAddVisibleZone(sZoneName)
void ScenarioDeleteVisibleZone(sZoneName)

Добавляет/убирает зону с карты. См. так же файл внешнего скрипта, функции addZone и isZoneOpened.

string ScenarioGetCurrentZoneName()

возвращает имя текущей зоны. Если таковой нет, то nil. Имя берется из поля Name, заданного при оформлении сценарной зоны (см. предыдущий постинг).

void ScenarioHighlightZone(sZoneName,bEnable = true)

Включает/отключает эффект мигания сектора зоны на глобальной карте.

Обращаю внимание, что прочие сценарные функции из S^2\S^3 как то

GetCurrentZoneAILevel
ScenarioAddGoal
ScenarioSetGoalComplete
SetGoalBlocking
ScenarioSetTaskComplete
SetMaxCriticalSeverity
IsScenarioBlockComplete
CheckScenarioBlockJustComplete
ScenarioSetGoalVisible
GetNextBlockOrder
IsReadyForNextBlock
IsMissionTaken
CompleteAllBlocksBefore
NumMissionInBlock
HighlightClue
NumAvailableMissions
TakeMission
ScenarioGetTaskComplete
Win
WasLastZoneComplete
WasLastZoneFailed
WasLastZoneEveryObjComplete

в СиМ либо не работают, либо бесполезны. И в целом не нужны.

Novik, 15 май 2006 #2

SilentSmart
Guest

➪ Novik

Вот! Вот! Вот то, что мне ТАК нужно... Спасибо огромнейшее 

SilentSmart, 15 май 2006 #3

Novik
Guest

Функции управления течением времени.

Для начала несколько упрощений, за ради облегчить мне жизнь.
Все функции, упомянутые в тексте, имена которых начинаются с большой буквы, реализованы в коде игры. Имена которых начинаются с маленькой буквы - в файлах внешних скриптов. Есть несколько исключений, только их я буду оговаривать отдельно.
Если кому не понятна нотация описания примитивов, поясняю. Параметры и возвращаемые значения, начинающиеся с буквы
n - число.
u - юнит.
g - группа юнитов.
s - строка.
Слово "void" в качестве результата функции означает, что функция ничего не возвращает. Несколько параметров через запятую там же означает, что функция возвращает более одного параметра. Троеточие в списке параметрв функции означает то, что функция принимает переменное количество параметров.

Итак, про время в игре. Оно полностью реализовано на скриптах (что, в частности, означает, что с небольшими упрощениями и оговорками данная схема может быть перенесена в моды для S^3).
Текущее время на локации характеризуется следующими параметрами.
1) Время суток в понятии движка. Влияет на дальность видимости, рассчет ToHit, применение определенных перков и т.п.
2) Время суток в понятии скрипта. Просто показывает, как соотносится текущее время с промежутком 9.00-21.00. (Соотв. константы см. в файлах внешних скриптов). Влияет на доступность/недоступность хайда.
3) Эффекты освещения (горят/не горят лампы, общее освещение на локации и т.п.)
Следует понимать, что всеми этими параметрами можно управлять *независимо*, хотя, конечно же, примитивы более высокого уровня написаны так, чтобы значения параметров находились в согласованном состоянии.

Время суток в понятии движка.

nimber GetTimeOfDay()

Возвращает DAY или NIGHT.

void SetTimeOfDay(nTimeOfDay, bSetAmbient = true)

Устанавливает время суток, если второй параметр = true, то устанавливает и соотв. освещение тоже.

Время суток в понятии скрипта.

nimber getTimeOfDay()

Возвращает DAY или NIGHT.

void applyTimeOfDay()

Согласовывает "время движка" со "временем скрипта", выставляя первое по последнему, выставляет должное освещение.
Автоматически вызывается из startStandardZone.
Учитывает тип текущей локации - в подземных локациях всегда ночь (как в понятии освещения, так и в понятии движка)

void skipHour(nHour, bNoApplyTime = false)

Быстрая прокрутка указанного количества часов (может быть дробным). Если второй параметр = false, то корректирует освещение и время суток.

void skipToHour(nHour, bNoApplyTime = false)

Аналогично, но время прокручивается *до* указанного часа.

void showSkip()

Показ игроку меню для быстрой прокрутки времени.

void pauseTime(bPause = false)

позволяет временно включить/отключить течение времени.

void startTime()

стартует поток, отсчитывающий время в realtime. Автоматически вызывается из startStandardZone. Обращаю внимание, что за временем в походовом режиме следит функция OnStartTurnPrim, вызываемая из обработчика OnStartTurn. Если Вы перекрываете этот обработчик в скрипте зоны - не забудьте вызвать в конце OnStartTurnPrim, иначе время в походовке идти не будет.

OnTimeOfDayChanged(nTimeOfDay)

Это обработчик, вызываемый при смене времени суток. Если Вы перекрываете этот обработчик в скрипте зоны - не забудьте вызвать в конце OnTimeOfDayChangedPrim, иначе запрет/разрешение хайда не будет обрабатываться корректно.

Эффекты освещения.
Эффекты освещения как таковые задаются на вкладке редактора AmbientLights. Каждая запись характеризует одну "световую сцену" с набором кучи свойств - цвет травы, общий оттенок, направление тени, наличие и густота тумана и т.п. Для СиМ подробно этим занимался Марков, если захочет, напишет значение полей сам.
Все зоны разбиты на 5 типов - AT_NORMAL, AT_DUST, AT_SPECIAL, AT_SPECIAL1, AT_SPECIAL2. Для каждого типа заданы 24 записи (т.е. переключение освещения раз в час). Желающие, кстати, могут сделать переключение освещения менее дискретным, увеличив количество записей.
Тип зоны передается в startStandardZone и в OnEnterZonePrim в качестве третьего параметра. Наличие нескольких типов обусловлено попыткой создать максимально разнообразные решения в освещении разных зон. Например, на зонах с большими оврагами хорошо смотрится схема AT_DUST, а на равнинных зонах наличие тумана выглядит плохо (особенно при наличии строений на зоне, т.к. туман будет скапливаться и внутри зданий тоже). На подземных зонах схема освещения вообще не меняется с течением времени.
Конкретный набор схем смотреть на упомянутой закладке редактора и в файле внешних скриптов, массив AmbientLights.

void SetupAmbientLight(nAmbientID)

задействовать эффект освещения с nAmbientID

void StopWeatherEffect()

остановить эффект дождя/снега. Для старта необходимо соотв. образом отредактировать погоду на поле свойств шаблона.

void SetAmbientEffect(nEffectID)

задействовать "общий" эффект. Он может иметь отношение как к освещению (например 941, красная мигалка) так и к погоде (817 - дождь). Работает "через пятое на десятое". Теоретически проигрывание эффекта должно прекращаться при вызове функции с параметром минус единица, практически это не всегда прокатывает. В отличие от SetupAmbientLight параметр берется не из AmbientLights, а из Effects (применительно к БД - из EffectTemplates)

Рассчет времени на перемещение между локациями.

number skipTravelTime(sZoneName,bNoApplyTime = false,bNoSkipTime = false)

Данная функция рассчитывает время на перемещение между двумя локациями (с учетом типа перемещения - на машине, пешком, на паравозе). Возвращает время в часах. В случае, если bNoSkipTime = true, производится немедленная коррекция времени. Последнее связано с тем, что данная функция используется в двух местах - при реальном перемещении между локациями (автоматически вызывается из OnEnterZonePrim и startStandardZone) и для показа времени на перемещение на глобальной карте.
Внутри, фактически, с помощью if/elseif реализована матрица перемещений. К слову, можно было решить и поэлегантнее, с помощью двумерного массива.

OnMouseUnderZone(sZoneName)
OnMouseMove()

пара перехватчиков, вызывается при работе мышкой на глобальной карте. Используется для показа времени на перемещение (OnMouseUnderZone, указатель мыши на к/л секторе) и чтобы убрать это время с экрана (OnMouseMove, указатель мыши вне сектора).
Обращаю внимание, что параметр sZoneName, засылаемый в OnMouseUnderZone, берется движком из поля Description, которое Вы задали при редактировании свойств сектора на глобальной карте. Чтобы все работало как надо, имена в Description должны совпадать с именами (SmallDescription) соответствующих сценарных зон.

1 человеку нравится это.

Novik, 15 май 2006 #4

(Вы должны войти или зарегистрироваться, чтобы ответить.)