Basic concepts for Using Creator and Runescript

Version 1.1

After reading the documentation, seeing the tutorial videos and Darwin's primer on Runescript, I thought another bit of contribution was needed, especially for those not orientated to programming, and help clarify a few things that, although written down already, I think could benefit from a different order, if targeted at new Creators. This document was initially developed for my own gathering of ideas, but then changed into this form.

I apologize if, due to my ignorance, I induce a reader into mistakes, but I believe you'll find this document helpful. Thanks to Darwin for his comments and suggestions.

VampiricDread

15 March 2001

The concepts

I've seen some confusion over triggers and events, so let me start with them. Events are occurrences (Pre-, On-, and Post-) that relate to the timing of RuneSword flow of things, while triggers are objects containing scripts – little computer high-level language programs – in Runescript (the Runesword system language for users – you and me, not Dan Schnake or Adam West), that you must associate to the events. No trigger makes sense if not associated to an event.

This leads us to another concept – objects. While the documentation refers 5 RuneSword objects (creatures, items, encounters, tomes, and dialog topics), triggers are also objects (as maps and tiles are), because they all can have properties defined, that makes them unique in their own class of objects, or using a different wording, they can have a number of characteristics that defines them in their class and another bunch of said properties that makes them different from the other objects in their class.

Example: an impassable mountain tile and an impassable forest tile are both in the same class (they are tiles), and share a property (both are impassable, or cannot be crossed by the party), but have an important difference at least: one is a mountain, the other a forest, therefore, they are depicted graphically different for the enjoyment of the player.

So, in all, you in fact have 8 classes of **objects**, that can be defined by their **properties**, and that flow through the **events** of Runesword, basically along with the story and the options the player takes, which can be customized to an extent by using Runescript, the programming language for Creator, through the **trigger** class of objects.

And that's it. These are the concepts.

Building your library of stuff

Adam and Dan are absolutely right when they say that you may create an adventure without ever worrying about coding/programming/scripting (take your pick). But – there's always a but – when what you want to do cannot be accomplished by the engine alone: spells, anyone? Of course you can copy a thing you saw, modify it, and accommodate it to your needs, but if you are an ambitious Creator (and there seems to be a lot out there), you need to go further,

sometimes, and build something new from something old or something different, if not from scratch (brrr).

To do this, I suggest you build your library of stuff. How do you do that?

Every single one of us has one major interest in RuneSword, besides playing the tomes out there: we want to CREATE stuff, be it a story, a map, nice graphics, whatever, we all have a bit of the Dungeon Master in ourselves (if not, why are you reading this at all?), and want to share our creativity with the world.

The problem here is a bit of focus:

Option 1. Are you good at computers graphics, like Sarchimus? Give the community graphics of Player Characters and Non-Players Characters, Tiles and Tilesets, Combat Wallpapers, and Items. Animations are nice, as well.

Option 2. Are you good at computer sounds? Show us your Wav, Mid, and special effects files.

Option 3. Are you good at coding Runescript, like Darwin? We need your triggers.

Option 4. Do you have a fertile imagination and/or are good with words? Toss them up into the open to get someone to string the adventure/creature/encounter into a Tome.

Option 5. You can do it all? Ask Dan and Adam for an employment opportunity (hehehe), after dazzling all of us with your masterpiece (you do have to show you're good, don't you?).

Whatever option you pick, I suggest that you start building your own library of stuff. Make sure it is separate from your playing RuneSword environment, so that you know what's standard, and what you are required to provide for your creation to work with any of us out here. Or keep a log. Or keep a list of what comes with RuneSword (like a list of contents, with dates, release, etc. to avoid surprises). Whatever you do, don't trust your memory, or people will complain.

Things you must know - Definitions

Nothing comes cheap in life (so they say), therefore let me give you some basics on requirements on your deliverables to the RuneSword community:

All graphics in RuneSword **must** be BMP.
All creatures must be 8bit (256 colours). I've never seen in the game (yet) a creature graphic bigger than 308x227 pixels.
All tiles must be EXACTLY 96x72 pixels and 8bit (256 colours).
A map maximum dimension is 255x255 tiles.
A tileset should have at least one tile for each of the 11 styles of tiles, if the engine is supposed to generate maps.
Items should be 32x32 pixels and 8bit (256 colours).
Combat wallpapers should be 629x353 pixels and 8bit (256 colours)

Looking into classes of objects – another view of things

I won't go into properties, because they are nicely and alphabetically covered in the "Encyclopedia Arcana" (12 full pages of them, expertly described). I also refer the Creator to check and understand the topic Commonly Used Events in the same document, which is copied and pasted to the following text box for your convenience:

Commonly Used Events

Probably 95% of all triggers use one of the events with the specific timing shown on this brief list. So don't worry about esoteric events like Post-RollArmorChit (great for a battle axe with a magical shield splitting effect) because you can look that kinda stuff up when needed. This Top 5 list shows the events/timing you should know by heart.

1. **Pre-Turn attached to a Creature** Number one for a reason. When you want a certain character to get a +2 to hit on his next turn, or –3 to defense, or whatever. You can also attach pre-turn triggers to items, but it's not necessary for a bonus to hit or damage because that can be built in to the item.
2. **On-Attack attached to a Creature** This event governs how computer-controlled creatures act in combat. It's unnecessary for simple fighters – the RS engine enables critters to fight competently with short and long-range weapons, no trigger needed. But that's it; no built in spellcasting brains or special attack/defense logic. Don't worry, though, some great On-Attack triggers are available in our libraries for cut and paste. You don't have to code such tricky stuff. Good On-Attack triggers containing attack logic are named "brains."
3. **On-UseOnCreature attached to an Item** A classic. When you point that wand at someone, by jiminy, you want it to do something.
4. **Post-Topic attached to a Topic** When you say something to someone, sometimes you want something to happen as a result. Could be an old man giving you a key - or maybe a curse.
5. **Pre-Turncycle attached to an Encounter** In combat, this fires just before characters and monsters get their turns. After they go, it fires again. Outside of combat, this fires every 10 steps the party travels. This is good for stuff that affects the whole party, like damaging them whilst standing in a room of boiling ooze.

I would like to point out, first, that some events are more "common" than others, requiring more attention from you as a coder or tweaker of someone else's code. The following is an exclusive list, in the sense that, as you go down, triggers above are not repeated:

Note: All other triggers are exclusive of the class of objects

- Triggers common to Creatures, Items, Encounters and Tomes

Turncycle – Frequent usage

- Triggers common to Creatures, and Items

Death – Frequent usage
Examine – Frequent usage
PickLock – Frequent usage
RollArmorChit – Rare usage
RollAttack – Rare usage
RollDamage – Rare usage
TargetOfApplyDamage - Frequent usage
TargetOfAttack – Rare usage
Turn – Frequent usage

- Triggers common to Items, and Encounters

SearchTraps – Frequent usage
Take – Frequent usage
Unlock – Frequent usage

As you can see, this effectively creates a sub-set of common triggers of 9, out of 13. While the total may look daunting, you must not forget that all trigger statements are in a drop-down box at your disposal, for the appropriate object, and the explanation of each is at hand.

Also, there are some hidden rules you should be aware of:

1) for all things involving spell casting, you need a trigger;
2) for any dialog where checks have to be made or items exchange hands, you need a trigger;
3) for any effect you want to happen delayed in time, you need a trigger (and a sub-trigger, we'll get into that now).

Sub-Triggers

This is a trigger within a trigger, and you have 2 ways to use sub-triggers:

1) You use the CopyTrigger statement to copy the sub-trigger into the relevant target object (encounter, creature, item, or even Party). You normally use it when targeting someone or something;
2) You use the ExecTrigger to fire the sub-trigger like you would do a sub-routine in programming (like a program within a program). Note that you must create the sub-trigger first, so that it is available when you complete the ExecTrigger statement, and that if you set Local.Fail=True in the sub-trigger, the calling trigger will also fail. This statement is particularly useful in complex situation requiring triggers, for cleanliness of your code.

Let me expand a bit on their usage.

You have seen that triggers are always related to events (the timings), and though a sub-trigger must respect this rule, it fires when the parent triggers tells it to, applying the proper defined timing it has for the sequence of events. Also note the duration box and its options: a sub-trigger will go away, if its duration is limited to turns, charges, or One time (single execution), but not if it's Infinite. This opens up a myriad of possibilities for the usage of sub-triggers, particularly when you CopyTrigger it.

A few examples could be:

1) Applying a bonus for a number of turns to a character;
2) Assigning cumulative penalties for a number of turns to a character;
3) Discharging a single fireball on top of a monster;
4) Installing a trap in a room that fires after a number of turns have elapsed;
5) Making an entrance collapse, if a door is opened somewhere else, thereby preventing the party from using the way they came to get out;
6) Replacing tiles in a map in a controlled way, so that you may create a sliding maze;
7) Allowing a NPC to teach or improve a skill to a member of your party.

The possibilities are endless, and allow you to create indirect effects in the world, without the player becoming aware of them immediately.

Nothing prevents you from creating sub-sub-triggers, for real complex actions, for example, you want to have a NPC that acts like a Shaolin monk, and your character goes around as his apprentice, but later, when he gets to become a master, he can take on an apprentice by himself, but what he can teach is limited by what he learned from the first master, or even several masters, with different teaching skills and areas for study.

Triggers and sub-triggers are the most difficult part to master, but they are also the most rewarding, enabling you to do pretty complex things.

Variables and handles

All variables are pre-defined for you; you cannot create your own, and they come in 4 broad flavours (or types):

1) Local variables are specific to a trigger, and reset once the trigger ends;
2) Global variables can be checked just about anywhere, and reset when you save or quit a tome;
3) TriggerNow variables allow you to store values you want to check later, and they remain as properties of that trigger. You can inspect the contents of these trigger properties through the use of handles (CreatureA, CreatureB, CreatureC, ItemA, ItemB, ItemC, TriggerA, TriggerB, TriggerC, EncounterA, EncounterB, EncounterC, etc.) from other triggers;
4) Factoid text variables permit you to store information that you want to remain globally available, even from save to save (where Local and Global variables are reset).

Some of the most commonly used variables have a range of permitted values, like:

ByteX – an integer number between 0 and 255 (both inclusive)
IntegerX – an integer number between –32000 and 32000

Note that RuneSword requires you to precede a value with POS for positive and NEG for negative numbers, so

Let Local.IntegerA = NEG 14000 is correct and assigns the value –14000 to Local.IntegerA

Last remarks

You can find a lot of examples all over the community. Start by exploring The Land Beyond tome, where you can find ideas tested and implemented in the adventure.

Don’t be shy to ask for help. This is an open community around the RuneSword project, and you are bound to find someone to help you.