

Sigmodr v0.0.2

Ben Boeckel <MathStuf@gmail.com>

September 23, 2008

Contents

1	What is Sigmodr?	3
2	Sigmod Classes	4
2.1	Ability	4
2.1.1	Variables	4
2.1.2	Examples	5
2.2	Author	8
2.2.1	Variables	8
2.3	Badge	8
2.3.1	Variables	8
2.4	Coin List	9
2.4.1	Variables	9
2.5	Coin List Object	10
2.5.1	Variables	10
2.6	Egg Group	10
2.6.1	Variables	10
2.7	Global Script	10
2.7.1	Variables	11
2.8	Item	11
2.8.1	Variables	11
2.9	Item Type	11
2.9.1	Variables	11
2.10	Map	12
2.11	Map Effect	14
2.12	Map Trainer	14
2.13	Map Trainer Team Member	14
2.14	Map Warp	14
2.15	Map Wild List	14

2.16	Map Wild List Encounter	14
2.17	Move	14
2.18	Nature	14
2.19	Sound	14
2.20	Species	14
2.21	Species Ability	14
2.22	Species Item	14
2.23	Species Move	14
2.24	Store	14
2.25	Tile	14
2.26	Time	14
2.27	Trainer	14
2.28	Type	14
3	Scripting	15
3.1	Collections	15
4	Battle System	16
5	Breeding	17

Chapter 1

What is Sigmodr?

Chapter 2

Sigmoid Classes

2.1 Ability

Abilities add an extra dimension to the battle system. Abilities can help turn the tide in a battle and balance species that are typically under-powered. Abilities were introduced to the original games in the third generation of the games.

Abilities are responsible for keeping track of any information they change during their execution and restoring it if necessary. Changing anything that is not local to the battle will persist after the battle is over (this allows status afflictions and moves that are permanently overwritten to persist). Take note that anything that is species-wide will not persist no matter what happens during the battle (i.e. types, base stats, effort values) since these are copied for the battle so that they can be modified for strategic purposes.

The APIs for the battle system will be under development in the near future with the beta release of Pokémodr.

2.1.1 Variables

Name The name of the ability. This is how the player will see the ability in the game.

Description A succinct description of what the ability does.

Script The script contains functions that will be triggered by the game when appropriate. Only functions that are written will be called, so dummy functions are not needed for those the ability does not need. The following events can be used in an ability (these are in no way final, the battle system design will provide all of these):

- | | |
|----------------------|-------------------------|
| • userSentOut | • enemyStatChanged |
| • userBeginTurn | • enemyFlinch |
| • userAttack | • enemyFaint |
| • userHit | • roundBegin |
| • userDamaged | • roundEnd |
| • userStatusChanged | • battleEnd |
| • userStatChanged | • heldItemUsed |
| • userFlinch | • itemUsed |
| • userFaint | • userFlee |
| • enemySentOut | • userSwitch |
| • enemyBeginTurn | • weatherChanged |
| • enemyAttack | • enemyAbilityTriggered |
| • enemyHit | • mapEntered |
| • enemyStatusChanged | • wildBattleStarted |

See chapter 3 on page 15 for more information on scripting.

2.1.2 Examples

Note These examples are not guaranteed to work until an API has been made for the game and battle system.

Changing types

This script will change the user's first type depending on the type of the last move used. Since types do not persist outside of the battle, the type only needs to be reset when the owner of the ability switches out or faints.

Code Listing 2.1: Type changer

```
import sigmod # Data from the Sigmod
import owner  # The owner of the ability

# Get the first type of the user
originalType = owner.type(0)
# Set the type requested
ghost = sigmod.type("Ghost")

def setType():
    # Set the type
    owner.setType(0, ghost)

def retreat():
    # Reset the type
    owner.setType(0, originalType)

owner.connect("sentOut()", setType)
owner.connect("switch()", retreat)
owner.connect("faint()", retreat)
```

Changing the weather

This script will force the weather to be rainy all the time. Other weather effects can be in effect from the time it works until the beginning of the next turn where this ability will kick in and make it rain again.

Code Listing 2.2: Weather changer

```
import arena # The battle system
import client # Used for messaging
import sigmod # Data from the Sigmod
import owne  # The owner of the ability
```

```

# Get the handle for the "Rain" weather
rain = sigmod.weather("Rain")

# Local variable to keep track of when to force the weather
needChanging = False

def setWeather():
    # Set the weather to rain for unlimited turns
    arena.setWeather(rain, -1)
    # Let the player(s) know that it has started raining
    client.message("The skies have opened up!")

def sentOut():
    setWeather()

def weatherChanged():
    # Force the weather back next turn
    # if it the weather change wasn't to rain
    if not arena.weather() == rain:
        needChanging = True

def roundStart():
    # Reset the weather if needed
    if needChanging:
        setWeather()
        needChanging = False

def retreat():
    # If the owner of the ability is
    # no longer in the battle, set the rain to stop
    if arena.weather() == rain:
        arena.setWeatherDuration() = 2
        client.message("The skies are clearing!")

# Connect signals from the arena
arena.connect("roundStart()", roundStart)
# Connect signals from the owner
owner.connect("sentOut()", sentOut)
owner.connect("switch()", retreat)
owner.connect("faint()", retreat)

```


2.2 Author

This is where the authors of the Sigmod go. This class could be expanded to allow for something more detailed to allow the creation of credits, but it is simple for now.

2.2.1 Variables

Name The name of the author.

Email The email address of the author.

Role What the person did.

2.3 Badge

Badges are given to the player after accomplishing a task. In the original games, they were used to allow the player to use moves in the main world to gain access to new parts of the world. They also boosted the stats for in-game battles and allowed the player to train to higher levels without side effects.

2.3.1 Variables

Name The name of the badge. This is the name that the player will see in-game.

Face The image that will be used in the game for the badge before it is obtained.

Badge The image of the badge and will show up in the player's information window.

Obey The level that the player can train to without any side effects.

Stat The multiplier for each stat will be applied to in-game battles to give an extra edge to the player. It will not have any effect on battles between two games.

2.4 Coin List

Coin lists are used as alternate places to acquire items and team members in exchange for coins that are won at slot machines and card flip games. These are typically much harder to afford than items in an item shop, but they are also usually more valuable.

2.4.1 Variables

Name This is the internal name of the coin list. The player will not be told what this is.

Script This script should interface with the user's items to determine the following:

- Check to see if the player has an item that can buy the objects on the list
- If so:
 - Set the `coinListMoney` object in `Game` to the amount of currency that the player has for display
 - Provide a `buy` function that takes the object to buy in the list
 - It should handle giving the object to the player and erroring if anything happens
- else:
 - Set the `coinListMoney` object in `Game` to 0
 - The game will take this as a signal that the player cannot buy things from this coin list and will not display it
 - Any messages necessary must be handled in the script

2.5 Coin List Object

Coin list objects are either items or team members that can be bought from coin lists.

2.5.1 Variables

Type Whether the value of the object points to a species for a team member or to an item.

Value The ID number of the item or species.

Amount The number of the object that the player gets at a time.

Cost How much the object costs.

2.6 Egg Group

An egg group is a group of species that can breed to create eggs. More details can be found in chapter 5 on page 17.

2.6.1 Variables

Name The name of the egg group. Used internally only.

2.7 Global Script

Global scripts are functions that are available on demand in any other script. They can do anything that the scripting API allows with the game or any common function that is used a lot in order to save space. They are loaded when the game is loaded and in the `scripts` action collection. See more on scripting in chapter 3 on page 15.

2.7.1 Variables

Name

The name of the action in the collection.

Script

The code of the script.

2.8 Item

Items are used in the game to heal, boost stats, and other similar things. They can be used during in-game battles and on the overworld maps.

2.8.1 Variables

Name

The name of the item

2.9 Item Type

Item types are used to group items into sections within the player's item storage.

2.9.1 Variables

Name

The name of the item type. Used internally only.

Computer

How many of the item the computer can store.

Player

How many of the item the player can carry at once.

Count

Determines how the items are counted and grouped in its storage. If they can be grouped, multiples of an item will count as one towards the limit of the storage, otherwise duplicates are counted individually.

2.10 Map

Maps are view of the overworld for the game. When not battling, the player travels around the maps in search of other battles and quests.

- 2.11 Map Effect
- 2.12 Map Trainer
- 2.13 Map Trainer Team Member
- 2.14 Map Warp
- 2.15 Map Wild List
- 2.16 Map Wild List Encounter
- 2.17 Move
- 2.18 Nature
- 2.19 Sound
- 2.20 Species
- 2.21 Species Ability
- 2.22 Species Item
- 2.23 Species Move
- 2.24 Store
- 2.25 Tile
- 2.26 Time
- 2.27 Trainer
- 2.28 Type

Chapter 3

Scripting

3.1 Collections

Chapter 4

Battle System

Chapter 5

Breeding