Ruby trunk - Feature #709

Enumerator#+

11/03/2008 07:02 PM - candlerb (Brian Candler)

Status:	Rejected	
Priority:	Normal	
Assignee:	knu (Akinori MUSHA)	
Target version:	2.0.0	

Description

=begin
Enumerators could be directly composable:

```
class Enumerator
def +(other)
Enumerator.new do |y|
each { |e| y << e }
other.each { |e| y << e }
end
end
end

if __FILE__ == $0
a = (1..3).to_enum + (10..12).to_enum + (17..20).to_enum
a.each { |i| puts i }
end
```

The only problem I can see here is that this might open the floodgates to requests for more methods such as & and |, and it's not clear how those would behave. (Personally I'd go for a merge which compares the head items from each of the operands, which assumes that the operands are already in sorted order, but others may disagree)
=end

Related issues:

Related to Ruby trunk - Feature #15144: Enumerator#chain	**Closed**

History

#1 - 11/04/2008 05:27 AM - candlerb (Brian Candler)

=begin
For clarity, here's what that example outputs:

```
$ ruby19 -v
ruby 1.9.1 (2008-10-28 revision 19983) [i686-linux]
$ ruby19 enum_plus.rb
1
2
3
10
11
12
17
18
19
20
$
```

=end

#2 - 11/29/2008 04:31 PM - ko1 (Koichi Sasada)

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#3 - 12/12/2008 07:37 PM - knu (Akinori MUSHA)

- Status changed from Open to Rejected

- Assignee changed from matz (Yukihiro Matsumoto) to knu (Akinori MUSHA)

=begin
To tell the truth, I was also tempted to add something like that and
you are not alone.

You might as well add:

```
class Enumerator
alias * cycle

def -@
Enumerator.new { |y| to_a.reverse_each { |i| y << i } }
end
end
```

and have fun with them as follows:

```
e = -[1,4,7].each * 3 + ("a".."c").each
p e.to_a #=> [3, 2, 1, 3, 2, 1, 3, 2, 1, "a", "b", "c"]
```

However, I am yet to think that even the addition of enumerators is so
commonly in need.

Besides, as you said, operators must be carefully chosen and defined so
that they work consistently and systematically and they look intuitive
and practical, all at the same time. We should also think how other
types of iterators should be introduced, such as bidirectional iterators
and indexable iterators, before defining a variety of operators for
the current Enumerator.

So, my answer here, for now, is the block based enumerator. Just as you
have aptly shown, you can define anything you like with it without the
need for predefined operators.
=end

#4 - 10/10/2018 06:02 AM - knu (Akinori MUSHA)

- Related to Feature #15144: Enumerator#chain added

#5 - 10/10/2018 06:12 AM - knu (Akinori MUSHA)

- Description updated

I've become positive about this proposal after experiences with potential use cases.

In today's developer meeting, Matz approved this proposal, so I'm going to start working on this!