#==============================================================================

#

# Бе Yanfly Engine Ace - Visual Battlers v1.01

# -- Last Updated: 2012.07.24

# -- Level: Easy

# -- Requires: n/a

#

# Бе Modified by:

# -- Yami

# -- Kread-Ex

# -- Archeia\_Nessiah

#==============================================================================

$imported = {} if $imported.nil?

$imported["YEA-VisualBattlers"] = true

#==============================================================================

# Бе Updates

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# 2012.12.18 - Added preset views and able to change direction in-game.

# 2012.07.24 - Finished Script.

# 2012.01.05 - Started Script.

#

#==============================================================================

# Бе Introduction

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# This script provides a visual for all actors by default charsets. The actions

# and movements are alike Final Fantasy 1, only move forward and backward when

# start and finish actions.

#

#==============================================================================

# Бе Instructions

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# To change the player direction in-game, use the snippet below in a script

# call:

#

# $game\_system.party\_direction = n

#

# To install this script, open up your script editor and copy/paste this script

# to an open slot below Бе Materials but above Бе Main. Remember to save.

#

#==============================================================================

# Бе Compatibility

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# This script is made strictly for RPG Maker VX Ace. It is highly unlikely that

# it will run with RPG Maker VX without adjusting.

#

#==============================================================================

module YEA

module VISUAL\_BATTLERS

#=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# - Party Location Setting -

#=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# These settings are adjusted for Party Location. Each Actor will have

# coordinates calculated by below formula. There are two samples coordinates

# below, change PARTY\_DIRECTION to the base index you want to use.

#=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

PARTY\_DIRECTION = 2 # This direction is opposite from actual direction.

PARTY\_LOCATION\_BASE\_COORDINATES ={

# Index => [base\_x, base\_y, mod\_x, mod\_y],

2 => [ 250, 340, 40, 0], #UP

4 => [ 150, 280, 20, -20], #LEFT

3 => [ 460, 280, 30, -10], #RIGHT

6 => [ 460, 230, 20, 20], #DEFAULT RIGHT

8 => [ 260, 230, 40, 0], #DOWN

} # Do not remove this.

PARTY\_LOCATION\_FORMULA\_X = "base\_x + index \* mod\_x"

PARTY\_LOCATION\_FORMULA\_Y = "base\_y + index \* mod\_y"

end # VISUAL\_BATTLERS

end # YEA

#==============================================================================

# Бе Editting anything past this point may potentially result in causing

# computer damage, incontinence, explosion of user's head, coma, death, and/or

# halitosis so edit at your own risk.

#==============================================================================

#==============================================================================

# ? Бе Direction

#==============================================================================

module Direction

#--------------------------------------------------------------------------

# self.correct

#--------------------------------------------------------------------------

def self.correct(direction)

case direction

when 1; return 4

when 3; return 6

when 7; return 4

when 9; return 6

else; return direction

end

end

#--------------------------------------------------------------------------

# self.opposite

#--------------------------------------------------------------------------

def self.opposite(direction)

case direction

when 1; return 6

when 2; return 8

when 3; return 4

when 4; return 6

when 6; return 4

when 7; return 6

when 8; return 2

when 9; return 4

else; return direction

end

end

end # Direction

#==============================================================================

# ? Бе Game\_System

#==============================================================================

class Game\_System; attr\_accessor :party\_direction; end

#==============================================================================

# ? Бе Game\_BattleCharacter

#==============================================================================

class Game\_BattleCharacter < Game\_Character

#--------------------------------------------------------------------------

# initialize

#--------------------------------------------------------------------------

def initialize(actor)

super()

setup\_actor(actor)

@move\_x\_rate = 0

@move\_y\_rate = 0

end

#--------------------------------------------------------------------------

# setup\_actor

#--------------------------------------------------------------------------

def setup\_actor(actor)

@actor = actor

@step\_anime = true

set\_graphic(@actor.character\_name, @actor.character\_index)

setup\_coordinates

dr = $game\_system.party\_direction || YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION

direction = Direction.opposite(dr)

set\_direction(Direction.correct(direction))

end

#--------------------------------------------------------------------------

# sprite=

#--------------------------------------------------------------------------

def sprite=(sprite)

@sprite = sprite

end

#--------------------------------------------------------------------------

# setup\_coordinates

#--------------------------------------------------------------------------

def setup\_coordinates

location = ($game\_system.party\_direction ||

YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION)

base\_x = YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_BASE\_COORDINATES[location][0]

base\_y = YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_BASE\_COORDINATES[location][1]

mod\_x = YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_BASE\_COORDINATES[location][2]

mod\_y = YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_BASE\_COORDINATES[location][3]

@actor.screen\_x = eval(YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_FORMULA\_X)

@actor.screen\_y = eval(YEA::VISUAL\_BATTLERS::PARTY\_LOCATION\_FORMULA\_Y)

@actor.origin\_x = @actor.screen\_x

@actor.origin\_y = @actor.screen\_y

@actor.create\_move\_to(screen\_x, screen\_y, 1)

end

#--------------------------------------------------------------------------

# index

#--------------------------------------------------------------------------

def index

return @actor.index

end

#--------------------------------------------------------------------------

# screen\_x

#--------------------------------------------------------------------------

def screen\_x

return @actor.screen\_x

end

#--------------------------------------------------------------------------

# screen\_y

#--------------------------------------------------------------------------

def screen\_y

return @actor.screen\_y

end

#--------------------------------------------------------------------------

# screen\_z

#--------------------------------------------------------------------------

def screen\_z

return @actor.screen\_z

end

end # Game\_BattleCharacter

#==============================================================================

# ? Бе Game\_Battler

#==============================================================================

class Game\_Battler < Game\_BattlerBase

#--------------------------------------------------------------------------

# public instance variables

#--------------------------------------------------------------------------

attr\_accessor :moved\_back

attr\_accessor :origin\_x

attr\_accessor :origin\_y

attr\_accessor :screen\_x

attr\_accessor :screen\_y

attr\_accessor :started\_turn

#--------------------------------------------------------------------------

# alias method: execute\_damage

#--------------------------------------------------------------------------

alias game\_battler\_execute\_damage\_vb execute\_damage

def execute\_damage(user)

game\_battler\_execute\_damage\_vb(user)

if @result.hp\_damage > 0

move\_backward(24, 6) unless @moved\_back

@moved\_back = true

end

end

#--------------------------------------------------------------------------

# face\_opposing\_party

#--------------------------------------------------------------------------

def face\_opposing\_party

direction = ($game\_system.party\_direction ||

YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION)

character.set\_direction(Direction.correct(direction)) rescue 0

end

#--------------------------------------------------------------------------

# new method: face\_coordinate

#--------------------------------------------------------------------------

def face\_coordinate(destination\_x, destination\_y)

x1 = Integer(@screen\_x)

x2 = Integer(destination\_x)

y1 = Graphics.height - Integer(@screen\_y)

y2 = Graphics.height - Integer(destination\_y)

return if x1 == x2 and y1 == y2

#---

angle = Integer(Math.atan2((y2-y1),(x2-x1)) \* 1800 / Math::PI)

if (0..225) === angle or (-225..0) === angle

direction = 6

elsif (226..675) === angle

direction = 9

elsif (676..1125) === angle

direction = 8

elsif (1126..1575) === angle

direction = 7

elsif (1576..1800) === angle or (-1800..-1576) === angle

direction = 4

elsif (-1575..-1126) === angle

direction = 1

elsif (-1125..-676) === angle

direction = 2

elsif (-675..-226) === angle

direction = 3

end

#---

character.set\_direction(Direction.correct(direction)) rescue 0

end

#--------------------------------------------------------------------------

# create\_move\_to

#--------------------------------------------------------------------------

def create\_move\_to(destination\_x, destination\_y, frames = 12)

@destination\_x = destination\_x

@destination\_y = destination\_y

frames = [frames, 1].max

@move\_x\_rate = [(@screen\_x - @destination\_x).abs / frames, 2].max

@move\_y\_rate = [(@screen\_y - @destination\_y).abs / frames, 2].max

end

#--------------------------------------------------------------------------

# update\_move\_to

#--------------------------------------------------------------------------

def update\_move\_to

@move\_x\_rate = 0 if @screen\_x == @destination\_x || @move\_x\_rate.nil?

@move\_y\_rate = 0 if @screen\_y == @destination\_y || @move\_y\_rate.nil?

value = [(@screen\_x - @destination\_x).abs, @move\_x\_rate].min

@screen\_x += (@destination\_x > @screen\_x) ? value : -value

value = [(@screen\_y - @destination\_y).abs, @move\_y\_rate].min

@screen\_y += (@destination\_y > @screen\_y) ? value : -value

end

#--------------------------------------------------------------------------

# move\_forward

#--------------------------------------------------------------------------

def move\_forward(distance = 24, frames = 12)

direction = forward\_direction

move\_direction(direction, distance, frames)

end

#--------------------------------------------------------------------------

# move\_backward

#--------------------------------------------------------------------------

def move\_backward(distance = 24, frames = 12)

direction = Direction.opposite(forward\_direction)

move\_direction(direction, distance, frames)

end

#--------------------------------------------------------------------------

# move\_direction

#--------------------------------------------------------------------------

def move\_direction(direction, distance = 24, frames = 12)

case direction

when 1; move\_x = distance / -2; move\_y = distance / 2

when 2; move\_x = distance \* 0; move\_y = distance \* 1

when 3; move\_x = distance / -2; move\_y = distance / 2

when 4; move\_x = distance \* -1; move\_y = distance \* 0

when 6; move\_x = distance \* 1; move\_y = distance \* 0

when 7; move\_x = distance / -2; move\_y = distance / -2

when 8; move\_x = distance \* 0; move\_y = distance \* -1

when 9; move\_x = distance / 2; move\_y = distance / -2

else; return

end

destination\_x = @screen\_x + move\_x

destination\_y = @screen\_y + move\_y

create\_move\_to(destination\_x, destination\_y, frames)

end

#--------------------------------------------------------------------------

# forward\_direction

#--------------------------------------------------------------------------

def forward\_direction

return ($game\_system.party\_direction ||

YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION)

end

#--------------------------------------------------------------------------

# move\_origin

#--------------------------------------------------------------------------

def move\_origin

create\_move\_to(@origin\_x, @origin\_y)

face\_coordinate(@origin\_x, @origin\_y)

@moved\_back = false

end

#--------------------------------------------------------------------------

# moving?

#--------------------------------------------------------------------------

def moving?

return false if dead? || !exist?

return @move\_x\_rate != 0 || @move\_y\_rate != 0

end

end # Game\_Battler

#==============================================================================

# ? Бе Game\_Actor

#==============================================================================

class Game\_Actor < Game\_Battler

#--------------------------------------------------------------------------

# overwrite method: use\_sprite?

#--------------------------------------------------------------------------

def use\_sprite?

return true

end

#--------------------------------------------------------------------------

# new method: screen\_x

#--------------------------------------------------------------------------

def screen\_x

return @screen\_x rescue 0

end

#--------------------------------------------------------------------------

# new method: screen\_y

#--------------------------------------------------------------------------

def screen\_y

return @screen\_y rescue 0

end

#--------------------------------------------------------------------------

# new method: screen\_z

#--------------------------------------------------------------------------

def screen\_z

return 100

end

#--------------------------------------------------------------------------

# new method: sprite

#--------------------------------------------------------------------------

def sprite

index = $game\_party.battle\_members.index(self)

return SceneManager.scene.spriteset.actor\_sprites[index]

end

#--------------------------------------------------------------------------

# new method: character

#--------------------------------------------------------------------------

def character

return sprite.character\_base

end

#--------------------------------------------------------------------------

# face\_opposing\_party

#--------------------------------------------------------------------------

def face\_opposing\_party

dr = $game\_system.party\_direction || YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION

direction = Direction.opposite(dr)

character.set\_direction(Direction.correct(direction)) rescue 0

end

#--------------------------------------------------------------------------

# forward\_direction

#--------------------------------------------------------------------------

def forward\_direction

return Direction.opposite(($game\_system.party\_direction ||

YEA::VISUAL\_BATTLERS::PARTY\_DIRECTION))

end

end # Game\_Actor

#==============================================================================

# ? Бе Game\_Enemy

#==============================================================================

class Game\_Enemy < Game\_Battler

#--------------------------------------------------------------------------

# new method: sprite

#--------------------------------------------------------------------------

def sprite

return SceneManager.scene.spriteset.enemy\_sprites.reverse[self.index]

end

#--------------------------------------------------------------------------

# new method: character

#--------------------------------------------------------------------------

def character

return sprite

end

end # Game\_Enemy

#==============================================================================

# ? Бе Game\_Troop

#==============================================================================

class Game\_Troop < Game\_Unit

#--------------------------------------------------------------------------

# alias method: setup

#--------------------------------------------------------------------------

alias game\_troop\_setup\_vb setup

def setup(troop\_id)

game\_troop\_setup\_vb(troop\_id)

set\_coordinates

end

#--------------------------------------------------------------------------

# new method: set\_coordinates

#--------------------------------------------------------------------------

def set\_coordinates

for member in members

member.origin\_x = member.screen\_x

member.origin\_y = member.screen\_y

member.create\_move\_to(member.screen\_x, member.screen\_y, 1)

end

end

end # Game\_Troop

#==============================================================================

# ? Бе Sprite\_Battler

#==============================================================================

class Sprite\_Battler < Sprite\_Base

#--------------------------------------------------------------------------

# public instance\_variable

#--------------------------------------------------------------------------

attr\_accessor :character\_base

attr\_accessor :character\_sprite

#--------------------------------------------------------------------------

# alias method: dispose

#--------------------------------------------------------------------------

alias sprite\_battler\_dispose\_vb dispose

def dispose

dispose\_character\_sprite

sprite\_battler\_dispose\_vb

end

#--------------------------------------------------------------------------

# new method: dispose\_character\_sprite

#--------------------------------------------------------------------------

def dispose\_character\_sprite

@character\_sprite.dispose unless @character\_sprite.nil?

end

#--------------------------------------------------------------------------

# alias method: update

#--------------------------------------------------------------------------

alias sprite\_battler\_update\_vb update

def update

sprite\_battler\_update\_vb

return if @battler.nil?

update\_move\_to

update\_character\_base

update\_character\_sprite

end

#--------------------------------------------------------------------------

# new method: update\_character\_base

#--------------------------------------------------------------------------

def update\_character\_base

return if @character\_base.nil?

@character\_base.update

end

#--------------------------------------------------------------------------

# new method: update\_character\_sprite

#--------------------------------------------------------------------------

def update\_character\_sprite

return if @character\_sprite.nil?

@character\_sprite.update

end

#--------------------------------------------------------------------------

# new method: update\_move\_to

#--------------------------------------------------------------------------

def update\_move\_to

@battler.update\_move\_to

end

#--------------------------------------------------------------------------

# new method: moving?

#--------------------------------------------------------------------------

def moving?

return false if @battler.nil?

return @battler.moving?

end

end # Sprite\_Battler

#==============================================================================

# ? Бе Sprite\_BattleCharacter

#==============================================================================

class Sprite\_BattleCharacter < Sprite\_Character

#--------------------------------------------------------------------------

# initialize

#--------------------------------------------------------------------------

def initialize(viewport, character = nil)

super(viewport, character)

character.sprite = self

end

end # Sprite\_BattleCharacter

#==============================================================================

# ? Бе Spriteset\_Battle

#==============================================================================

class Spriteset\_Battle

#--------------------------------------------------------------------------

# public instance\_variable

#--------------------------------------------------------------------------

attr\_accessor :actor\_sprites

attr\_accessor :enemy\_sprites

#--------------------------------------------------------------------------

# overwrite method: create\_actors

#--------------------------------------------------------------------------

def create\_actors

total = $game\_party.max\_battle\_members

@current\_party = $game\_party.battle\_members.clone

@actor\_sprites = Array.new(total) { Sprite\_Battler.new(@viewport1) }

for actor in $game\_party.battle\_members

@actor\_sprites[actor.index].battler = actor

create\_actor\_sprite(actor)

end

end

#--------------------------------------------------------------------------

# new method: create\_actor\_sprite

#--------------------------------------------------------------------------

def create\_actor\_sprite(actor)

character = Game\_BattleCharacter.new(actor)

character\_sprite = Sprite\_BattleCharacter.new(@viewport1, character)

@actor\_sprites[actor.index].character\_base = character

@actor\_sprites[actor.index].character\_sprite = character\_sprite

actor.face\_opposing\_party

end

#--------------------------------------------------------------------------

# alias method: update\_actors

#--------------------------------------------------------------------------

alias spriteset\_battle\_update\_actors\_vb update\_actors

def update\_actors

if @current\_party != $game\_party.battle\_members

dispose\_actors

create\_actors

end

spriteset\_battle\_update\_actors\_vb

end

#--------------------------------------------------------------------------

# new method: moving?

#--------------------------------------------------------------------------

def moving?

return battler\_sprites.any? {|sprite| sprite.moving? }

end

end # Spriteset\_Battle

#==============================================================================

# ? Бе Scene\_Battle

#==============================================================================

class Scene\_Battle < Scene\_Base

#--------------------------------------------------------------------------

# public instance variables

#--------------------------------------------------------------------------

attr\_accessor :spriteset

#--------------------------------------------------------------------------

# alias method: process\_action\_end

#--------------------------------------------------------------------------

alias scene\_battle\_process\_action\_end\_vb process\_action\_end

def process\_action\_end

start\_battler\_move\_origin

scene\_battle\_process\_action\_end\_vb

end

#--------------------------------------------------------------------------

# alias method: execute\_action

#--------------------------------------------------------------------------

alias scene\_battle\_execute\_action\_vb execute\_action

def execute\_action

start\_battler\_move\_forward

scene\_battle\_execute\_action\_vb

end

#--------------------------------------------------------------------------

# new method: start\_battler\_move\_forward

#--------------------------------------------------------------------------

def start\_battler\_move\_forward

return if @subject.started\_turn

@subject.started\_turn = true

@subject.move\_forward

wait\_for\_moving

end

#--------------------------------------------------------------------------

# new method: start\_battler\_move\_origin

#--------------------------------------------------------------------------

def start\_battler\_move\_origin

@subject.started\_turn = nil

move\_battlers\_origin

wait\_for\_moving

@subject.face\_opposing\_party rescue 0

end

#--------------------------------------------------------------------------

# new method: move\_battlers\_origin

#--------------------------------------------------------------------------

def move\_battlers\_origin

for member in all\_battle\_members

next if member.dead?

next unless member.exist?

member.move\_origin

end

end

#--------------------------------------------------------------------------

# new method: wait\_for\_moving

#--------------------------------------------------------------------------

def wait\_for\_moving

update\_for\_wait

update\_for\_wait while @spriteset.moving?

end

end # Scene\_Battle

#==============================================================================

#

# Бе End of File

#

#==============================================================================