

Ajoutez un script au dessus de "main", appelez le "Mini Map"

```
#=====
=====
# ■Game_Event
#=====
=====

class Game_Event < Game_Character
#-----
# ●name
#-----
def name
return @event.name
end
end

#=====
=====
# ■Map_Base
#-----
# Base class for mini maps
#
# made by squall // squall@loeber.zzn.com
#=====
=====

class Map_Base
MP_VERSION = 5
#-----
# ●Initialize
#-----
def initialize(corner, use_window_skin)
@tileset_name = $game_map.tileset_name
@bg = Window_Base.new(0, 0, 144, 112)
@bg.z = 9000
if no_display?
@bg.window_skin = nil
else
unless use_window_skin
@bg.dispose
@bg = Window_Base.new(0, 0, 160, 128)
@bg.contents = Bitmap.new(128, 96)
@bg.window_skin = nil
bmp = RPG::Cache.picture("mapback")
@bg.contents.blit(0, 0, bmp, Rect.new(0, 0, 128, 96))
@bg.z = 9015
end
end
end
```

```

end
@map = Sprite.new
@map.bitmap = Bitmap.new(map_width, map_height)
@map.z = 9005
@event = Sprite.new
@event.bitmap = Bitmap.new(map_width, map_height)
@event.z = 9010
self.back_opacity = 180
self.opacity = 180
case corner
when 1
self.x = 16
self.y = 16
when 2
self.x = 640 - width - 16
self.y = 16
when 3
self.x = 16
self.y = 480 - height - 16
when 4
self.x = 640 - width - 16
self.y = 480 - height - 16
else
self.x = 16
self.y = 16
end
end
end
#-----
# ●display_map?
#-----
def no_map_display?
for event in $game_map.events.values
if event.name.include?("[no map]")
return true
end
end
return false
end
#-----
# ●display_map?
#-----
def no_event_display?
for event in $game_map.events.values
if event.name.include?("[no event]")
return true
end
end
return false
end
#-----
# ●display_map?
#-----

```

```

def no_display?
  for event in $game_map.events.values
  if event.name.include?("[no map]") and event.name.include?("[no event]")
  return true
  end
  end
  return false
  end
#-----
# ●dispose
#-----
def dispose
  @bg.dispose
  @map.bitmap.dispose
  @map.dispose
  @event.bitmap.dispose
  @event.dispose
end
#-----
# ●map
#-----
def map
  return @map
end
#-----
# ●event
#-----
def event
  return @event
end
#-----
# ●width
#-----
def width
  return 128
end
#-----
# ●opacity=
#-----
def height
  return 96
end
#-----
# ●opacity=
#-----
def visible=(bool)
  @bg.visible = bool
  @event.visible = bool
  @map.visible = bool
end
#-----
# ●opacity

```

```

#-----
def visible
return @bg.visible
end
#-----
# ●opacity=
#-----
def opacity=(opacity)
@event.opacity = opacity
@map.opacity = opacity
end
#-----
# ●opacity
#-----
def opacity
return @event.opacity
end
#-----
# ●back_opacity=
#-----
def back_opacity=(opacity)
@bg.opacity = opacity
@bg.contents_opacity = opacity if @bg.contents != nil
end
#-----
# ●back_opacity
#-----
def back_opacity
return @bg.opacity
end
#-----
# ●x=
#-----
def x=(x)
@bg.x = x - (@bg.width - 128) / 2
@event.x = x + 8
@map.x = x + 8
end
#-----
# ●x
#-----
def x
return @bg.x
end
#-----
# ●y=
#-----
def y=(y)
@bg.y = y - (@bg.height - 96) / 2
@event.y = y + 8
@map.y = y + 8
end

```

```

#-----
# ●y
#-----
def y
return @bg.y
end
#-----
# ●map_width
#-----
def map_width
return $game_map.width * 112/20
end
#-----
# ●map_height
#-----
def map_height
return $game_map.height * 80/15
end
#-----
# ●display_x
#-----
def display_x
return $game_map.display_x / 128 * 112/20
end
#-----
# ●map_height
#-----
def display_y
return $game_map.display_y / 128 * 80/15
end
end
#=====
=====
# ■Map_Mini
#-----
#   Base class for mini maps
#
# made by squall // squall@loeber.zzn.com
#=====
=====

class Map_Mini < Map_Base
#-----
# ●initialize
#-----
def initialize(corner, use_window_skin)
super(corner, use_window_skin)
unless no_map_display?
draw_map
end
end
end
#-----

```

```

# ●update
#-----
def update
  map.src_rect.set(display_x, display_y, width - 16, height - 16)
  if @tileset_name != $game_map.tileset_name
    @tileset_name = $game_map.tileset_name
  unless no_map_display?
    map.bitmap.clear
    draw_map
  end
end
#-----
# ●draw_map
#-----
def draw_map
  bitmap = Bitmap.new($game_map.width * 32, $game_map.height * 32)
  for i in 0...($game_map.width * $game_map.height)
    x = i % $game_map.width
    y = i / $game_map.width
    for level in 0...3
      tile_id = $game_map.data[x, y, level]
      if tile_id >= 384
        tileset_bitmap = RPG::Cache.tile($game_map.tileset_name, tile_id, 0)
        src_rect = Rect.new(0, 0, 32, 32)
        bitmap.blit(x * 32, y * 32, tileset_bitmap, src_rect)
      end
      if tile_id >= 48 and tile_id < 384
        id = tile_id / 48 - 1
        tileset_bitmap = RPG::Cache.autotile($game_map.autotile_names[id])
        src_rect = Rect.new(32, 64, 32, 32)
        bitmap.blit(x * 32, y * 32, tileset_bitmap, src_rect)
      end
    end
  end
  d_rect = Rect.new(0, 0, map_width, map_height)
  s_rect = Rect.new(0, 0, bitmap.width, bitmap.height)
  map.bitmap.stretch_blit(d_rect, bitmap, s_rect)
  bitmap.clear
  bitmap.dispose
end
#=====
=====
# ■Map_Event
#-----
#   draw the events and hero position
#=====
=====

class Map_Event < Map_Mini
#-----

```

```

# ●initialize
#-----
def initialize(corner = 4, window_skin = true)
  super(corner, window_skin)
  @dots = []
end
#-----
# ●refresh_dots
#-----
def refresh_event_dots
  for event in $game_map.events.values
    bitmap = nil
    x = event.x * map_width / $game_map.width
    y = event.y * map_height / $game_map.height
    next if event.list == nil
    for i in 0...event.list.size
      if event.list[i].parameters[0].is_a?(String)
        if event.list[i].parameters[0] == "event"
          bitmap = RPG::Cache.picture(event.list[i].parameters[0])
          break
        elsif event.list[i].parameters[0] == "enemy"
          bitmap = RPG::Cache.picture(event.list[i].parameters[0])
          break
        elsif event.list[i].parameters[0].include?("teleport")
          bitmap = RPG::Cache.picture("teleport")
          break
        elsif event.list[i].parameters[0] == "chest"
          bitmap = RPG::Cache.picture(event.list[i].parameters[0])
          break
        elsif event.list[i].parameters[0] == "npc"
          bitmap = RPG::Cache.picture(event.list[i].parameters[0])
          break
        elsif event.list[i].parameters[0] == "savepoint"
          bitmap = RPG::Cache.picture(event.list[i].parameters[0])
          break
        else
          bitmap = nil
        end
      end
    end
    @dots.push([x, y, bitmap])
  end
end
#-----
# ●refresh_dots
#-----
def refresh_player_dot
  x = $game_player.x * map_width / $game_map.width
  y = $game_player.y * map_height / $game_map.height
  bitmap = RPG::Cache.picture("hero")
  @dots.push([x, y, bitmap])
end

```

```

#-----
# ●update
#-----
def update
  super
  @dots.clear
  event.bitmap.clear
  refresh_event_dots unless no_event_display?
  refresh_player_dot unless no_display?
  for dot in @dots
    unless dot[2] == nil
      event.bitmap.blit(dot[0], dot[1], dot[2], Rect.new(0, 0, 4, 4))
    end
  end
  event.src_rect.set(display_x, display_y, width - 16, height - 16)
end
end

```

```

#=====
=====
# ■Map_Full
#-----
# made by squall // squall@loehher.zzn.com
#=====
=====

```

```

class Map_Full
#-----
# ●Initialize
#-----
def initialize
  @dots = []
  @teleport_sprites = []
  if $game_map.width > $game_map.height
    @map_width = 640
    @map_height = $game_map.height * @map_width / $game_map.width
    if @map_height > 480
      @map_height = 480
    end
  else
    @map_height = 480
    @map_width = $game_map.width * @map_height / $game_map.height
    if @map_width > 640
      @map_width = 640
    end
  end
  @map = Sprite.new
  @event = Sprite.new
  @map.bitmap = Bitmap.new(width, height)
  @event.bitmap = Bitmap.new(width, height)
  @map.x = @event.x = 320 - width / 2
  @map.y = @event.y = 240 - height / 2
  draw_map unless no_map_display?
end
end

```

```

draw_event_dots unless no_event_display?
draw_player_dot unless no_display?
if no_display?
  @message = Window_Base.new(0, 208, 640, 64)
  message = "the map is not available"
  @message.contents = Bitmap.new(608, 32)
  @message.contents.font.name = "Arial"
  @message.contents.font.size = 32
  @message.contents.font.color.set(255, 255, 255, 100)
  @message.contents.draw_text(-1, -1, 608, 32, message, 1)
  @message.contents.draw_text(-1, 1, 608, 32, message, 1)
  @message.contents.draw_text(1, -1, 608, 32, message, 1)
  @message.contents.draw_text(1, 1, 608, 32, message, 1)
  @message.contents.font.color.set(255, 255, 255, 50)
  @message.contents.draw_text(-2, -2, 608, 32, message, 1)
  @message.contents.draw_text(-2, 2, 608, 32, message, 1)
  @message.contents.draw_text(2, -2, 608, 32, message, 1)
  @message.contents.draw_text(2, 2, 608, 32, message, 1)
  @message.contents.font.color.set(255, 255, 255)
  @message.contents.draw_text(0, 0, 608, 32, message, 1)
  @message.window_skin = nil
end
end
#-----
# ●display_map?
#-----
def no_map_display?
  for event in $game_map.events.values
    if event.name.include?("[full]")
      return false
    end
    if event.name.include?("[no map]")
      return true
    end
  end
  return false
end
#-----
# ●display_map?
#-----
def no_event_display?
  for event in $game_map.events.values
    if event.name.include?("[full]")
      return false
    end
    if event.name.include?("[no event]")
      return true
    end
  end
  return false
end
#-----

```

```

# ●display_map?
#-----
def no_display?
for event in $game_map.events.values
if event.name.include?("[full]")
return false
end
if event.name.include?("[no map]") and event.name.include?("[no event]")
return true
end
end
return false
end
#-----
# ●dispose
#-----
def dispose
for sprite in @teleport_sprites
sprite.dispose
end
@message.dispose if no_display?
@map.bitmap.dispose
@map.dispose
@event.bitmap.dispose
@event.dispose
end
#-----
# ●width
#-----
def width
return @map_width
end
#-----
# ●opacity=
#-----
def height
return @map_height
end
#-----
# ●draw_map
#-----
def draw_map
bitmap = Bitmap.new($game_map.width * 32, $game_map.height * 32)
for i in 0..($game_map.width * $game_map.height)
x = i % $game_map.width
y = i / $game_map.width
for level in 0..3
tile_id = $game_map.data[x, y, level]
if tile_id >= 384
tileset_bitmap = RPG::Cache.tile($game_map.tileset_name, tile_id, 0)
src_rect = Rect.new(0, 0, 32, 32)
bitmap.blit(x * 32, y * 32, tileset_bitmap, src_rect)

```

```

end
if tile_id >= 48 and tile_id < 384
id = tile_id / 48 - 1
tileset_bitmap = RPG::Cache.autotile($game_map.autotile_names[id])
src_rect = Rect.new(32, 64, 32, 32)
bitmap.blit(x * 32, y * 32, tileset_bitmap, src_rect)
end
end
end
d_rect = Rect.new(0, 0, width, height)
s_rect = Rect.new(0, 0, bitmap.width, bitmap.height)
@map.bitmap.stretch_blit(d_rect, bitmap, s_rect)
bitmap.clear
bitmap.dispose
end
#-----
# ●refresh_dots
#-----
def draw_event_dots
for event in $game_map.events.values
bitmap = nil
x = event.x * width / $game_map.width
y = event.y * height / $game_map.height
next if event.list == nil
for i in 0...event.list.size
if event.list[i].parameters[0].is_a?(String)
if event.list[i].parameters[0] == "event"
bitmap = RPG::Cache.picture(event.list[i].parameters[0])
break
elsif event.list[i].parameters[0] == "enemy"
bitmap = RPG::Cache.picture(event.list[i].parameters[0])
break
elsif event.list[i].parameters[0].include?("teleport")
bitmap = RPG::Cache.picture("teleport")
name = event.list[i].parameters[0].dup
name.slice!("teleport, ")
@teleport_sprites.push(new_name_sprite(name, x, y))
break
elsif event.list[i].parameters[0] == "chest"
bitmap = RPG::Cache.picture(event.list[i].parameters[0])
break
elsif event.list[i].parameters[0] == "npc"
bitmap = RPG::Cache.picture(event.list[i].parameters[0])
break
elsif event.list[i].parameters[0] == "savepoint"
bitmap = RPG::Cache.picture(event.list[i].parameters[0])
break
else
bitmap = nil
end
end
end
end
end

```

```

@dots.push([x, y, bitmap])
end
for dot in @dots
unless dot[2] == nil
@event.bitmap.blit(dot[0], dot[1], dot[2], Rect.new(0, 0, 4, 4))
end
end
end
#-----
# ●new_name_sprite
#-----
def new_name_sprite(name, x, y)
sprite = Sprite.new
sprite.y = y + 240 - height / 2
x + 128 > 640 ? sprite.x = 512 : sprite.x = x + 320 - width / 2
bitmap = Bitmap.new(128, 32)
bitmap.font.name, bitmap.font.size = "Arial", 20
bitmap.font.color.set(255, 255, 255)
bitmap.draw_text(0, 0, 128, 32, name)
sprite.bitmap = bitmap
return sprite
end
#-----
# ●refresh_dots
#-----
def draw_player_dot
x = $game_player.x * width / $game_map.width
y = $game_player.y * height / $game_map.height
bitmap = RPG::Cache.picture("hero")
@event.bitmap.blit(x, y, bitmap, Rect.new(0, 0, 4, 4))
end
end

#=====
=====
# ■Scene_MiniMap
#-----
# draw the map full screen
#-----
=====

class Scene_MiniMap
#-----
# ●main
#-----
def main
@map = Map_Full.new
Graphics.transition
loop do
Graphics.update
Input.update
update

```

```

if $scene != self
break
end
end
Graphics.freeze
@map.dispose
end
#-----
# ●Update the contents of all five windows on the main menu
#-----
def update
if Input.trigger?(Input::B) | Input.trigger?(Input::CTRL)
$game_system.se_play($data_system.cancel_se)
$scene = Scene_Map.new
return
end
end
end

#=====
=====
# ■Scene_Map
#-----
# draw the mini map
# @corner is the corner you want the mini map to be displayed in.
# 1 is upper left, 2 is upper right, 3 is bottom left and 4 is bottom right
#
# @use_windowskin is whether true or false. true if you want to use the
# the current window skin for the minimap background.
# or false if you want to use the picture named mapback in your picture folder.
#=====
=====

class Scene_Map
alias main_minimap main
alias update_minimap update
alias transfer_minimap transfer_player
#-----
# ●initialize
#-----
def initialize
@corner = 4 # 1 or 2 or 3 or 4
@use_windowskin = false # true or false
end
#-----
# ●main
#-----
def main
@event_map = Map_Event.new(@corner, @use_windowskin)
main_minimap
@event_map.dispose
end

```

```
#-----  
# ●update  
#-----  
def update  
  @event_map.update  
  if $game_system.map_interpreter.running?  
    @event_map.visible = false  
  else  
    @event_map.visible = true  
  end  
  if Input.trigger?(Input::CTRL)  
    $game_system.se_play($data_system.decision_se)  
    $scene = Scene_MiniMap.new  
  return  
  end  
  update_minimap  
end  
#-----  
# ●transfer_player  
#-----  
def transfer_player  
  transfer_minimap  
  @event_map.dispose  
  @event_map = Map_Event.new(@corner, @use_window_skin)  
end  
end
```

QUESTIONS: [pierreon@laposte.net](mailto:pierreon@laposte.net)

<http://epidemais.idoo.com/>