#Basic Quest System v1.3f

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

#Features: Quests! What more can you say.

#

#Usage: Set up your quests and away you go!

# Script calls:

# accept\_quest(:questid) - force quest accept

# ask\_accept(:questid) - open quest acceptance window

# abandon\_quest(:questid) - force quest abandon

# turnin\_quest(:questid) - force quest turnin

# fail\_quest(:questid) - force abandon with ME

# ask\_turnin(:questid) - open quest complete window

#

# adv\_obj(:questid, :objectiveid, value) - changes obj by value

# set\_obj(:questid, :objectiveid, value) - sets obj to value

# obj(:questid, :objectiveid) - gets obj value

# hide\_obj(:questid, :objectiveid) - hides objective

# show\_obj(:questid, :objectiveid) - shows objective

#

# $game\_quests[:questid].accepted? - true if quest is accepted

# $game\_quests[:questid].completed? - true if quest is completed

# $game\_quests[:questid].turned\_in? - true if quest is turned in

#

# Examples:

# The obj function can be used in conditional branches to check progress

# of certain objectives. Example.

# #Checking if :obj3 of :quest89 is greater than 3:

# obj(:quest89, :obj3) > 3

#

#~ #----------#

#-- Script by: V.M of D.T

#

#- Questions or comments can be:

# given by email: sumptuaryspade@live.ca

# provided on facebook: http://www.facebook.com/DaimoniousTailsGames

# All my other scripts and projects can be found here: http://daimonioustails.weebly.com/

#

#--- Free to use in any project, commercial or non-commercial, with credit given

# - - Though a donation's always a nice way to say thank you~ (I also accept actual thank you's)

#Visibility of quest log on map

$questlogvisibility = true

#Maximum # of quests displayed on the quest log overlay

$questlogmaxdisplay = 5

#Quest log position, 1 - top-left, 2 - top-right

QUEST\_LOG\_POSITION = 2

#Quest log offsets

QUEST\_LOG\_OFFSET\_X = 0

QUEST\_LOG\_OFFSET\_Y = 0

# Quest Format and set up!

# DETAILS[:quest\_id] = {

# :name => "quest name" #Quest name

# :level => value #Arbitrary value (Optional)

# :difficulty => "string" #Arbitrary string (Optional)

# :auto\_complete => true #Recieve rewards on the spot (Optional)

# :abandonable => false #Set's whether quest can be abandoned (Optional)

# :force\_accept => true #ask\_accept only allows accepting (Optional)

# :force\_turnin => true #ask\_turnin only allows completing (Optional)

# }

# DESCRIPTIONS[:quest\_id] = {

# :qgiver\_name => "string" #Quest giver name (shows in log) (Optional)

# :location => "string" #Quest giver location (shows in log) (Optional)

# :desc => "string" #Description of quest displayed in log (Optional)

# }

# OBJECTIVES[:quest\_id] = { #Quest objectives, "string" is name, id is max value

# # boolean is hidden objective (true for hidden)

# :obj\_id1 => ["string", id]

# :obj\_id2 => ["string", id, boolean],

# etc...

# }

# REWARDS[:quest\_id] = {

# :gold => value #Gold recieved from quest (Optional)

# :exp => value #Exp recieved from quest (Optional)

# #Items recieved from quest, :type is :item, :weapon, or :armor

# :scale\_exp => value #Percent value to scale exp based on level vs party

# :items => [[:type,id,value], ...]], (Optional)

# }

module QUEST

DETAILS= {}

DESCRIPTIONS = {}

OBJECTIVES = {}

REWARDS = {}

#Main Quest 1

DETAILS[:questid001] = {

:name => "Get rid of slimes"}

DESCRIPTIONS[:questid001] = {

:desc => " Beowulf noticed slimes near his house

which is rather unlikely sight." }

OBJECTIVES[:questid001] = {

:obj1 => ["Kill 2 groups of Slimes",2], }

REWARDS[:questid001] = {

:exp => 15}

#Main Quest 2

DETAILS[:questid002] = {

:name => "Help the citizen"}

DESCRIPTIONS[:questid002] = {

:location => "Town",

:desc => " Beowulf heard cry for help and saw

woman attacked by... chocolate bats.

Time to save her"

}

OBJECTIVES[:questid002] = {

:obj1 => ["Kill attackers",2] }

REWARDS[:questid002] = {}

#Main Quest 3

DETAILS[:questid003] = {

:name => "Old friend nearby"}

DESCRIPTIONS[:questid003] = {

:location => "Town",

:desc => " Seeing familiar face, Beowulf

decides to talk with her" }

OBJECTIVES[:questid003] = {

:obj1 => ["Talk to Helen",1] }

REWARDS[:questid003] = {}

#Main Quest 4

DETAILS[:questid004] = {

:name => "Back to guild"}

DESCRIPTIONS[:questid004] = {

:location => "Town",

:desc => " Beowulf needs to take his

beloved sword from guild storage" }

OBJECTIVES[:questid004] = {

:obj1 => ["Talk with guildmaster",1] }

REWARDS[:questid004] = {}

#Main Quest 5

DETAILS[:questid005] = {

:name => "Destroy chocolate hazard"}

DESCRIPTIONS[:questid005] = {

:qgiver\_name => "Guildmaster",

:location => "Town",

:desc => " Our heroes need to destroy

monsters around town"

}

OBJECTIVES[:questid005] = {

:obj1 => ["Kill chocolate monsters",8],

:obj2 => ["Report to guilmaster",1]}

REWARDS[:questid005] = {

:gold => 200,

:exp => 120

}

#Main Quest 6

DETAILS[:questid006] = {

:name => "Help merchants"}

DESCRIPTIONS[:questid006] = {

:qgiver\_name => "Guildmaster",

:location => "Town",

:desc => " Our heroes came to help

merchant's get rid of well... chocolate"

}

OBJECTIVES[:questid006] = {

:obj1 => ["Kill chocolate monsters",5] }

REWARDS[:questid006] = {

:gold => 250,

:exp => 165

}

#Main Quest 7

DETAILS[:questid007] = {

:name => "Apprehend the mage"}

DESCRIPTIONS[:questid007] = {

:desc => " It was high time to

to find that one who caused

everything"

}

OBJECTIVES[:questid007] = {

:obj1 => ["Find villainous mage", 1] }

REWARDS[:questid007] = {

:gold => 500,

:exp => 400

}

#Main Quest 8

DETAILS[:questid008] = {

:name => "Find the Dentist"}

DESCRIPTIONS[:questid008] = {

:desc => " Helen proposed to

made new ally for her

trickery plan"

}

OBJECTIVES[:questid008] = {

:obj1 => ["Go to inn and find person

that Helen was talking about", 1] }

REWARDS[:questid008] = {

:exp => 100

}

#Main Quest 9

DETAILS[:questid009] = {

:name => "Helen's plan"}

DESCRIPTIONS[:questid009] = {

:desc => " Fergus agreed to join

forces"

}

OBJECTIVES[:questid009] = {

:obj1 => ["Scare Alice", 1] }

REWARDS[:questid009] = {

:gold => 400,

:exp => 200

}

#Main Quest 10

DETAILS[:questid010] = {

:name => "Alice"}

DESCRIPTIONS[:questid010] = {

:desc => "Our heroes decided to

meet up in Helen's house"

}

OBJECTIVES[:questid010] = {

:obj1 => ["Talk to Alice near Helen's House", 1] }

REWARDS[:questid010] = {

:exp => 100

}

#Main Quest 11

DETAILS[:questid011] = {

:name => "Meeting up"}

DESCRIPTIONS[:questid011] = {

:desc => "Our heroes met up to

discuss next actions"

}

OBJECTIVES[:questid011] = {

:obj1 => ["Talk to Helen", 1] }

REWARDS[:questid011] = {

:exp => 100

}

#Main Quest 12

DETAILS[:questid012] = {

:name => "Advancing"}

DESCRIPTIONS[:questid012] = {

:desc => "Our heroes made their

way to cave entrance"

}

OBJECTIVES[:questid012] = {

:obj1 => ["Move to north-eastern cave", 1] }

REWARDS[:questid012] = {

:exp => 100

}

#Main Quest 13

DETAILS[:questid013] = {

:name => "Chocolate for demon"}

DESCRIPTIONS[:questid013] = {

:desc => " Heroes decided to avoid

senseless confrontation and

bring Ibaraki chocolate"

}

OBJECTIVES[:questid013] = {

:obj1 => ["Kill 3 chocomonsters", 3] }

REWARDS[:questid013] = {

:gold => 400,

:exp => 200

}

#Main Quest 14

DETAILS[:questid014] = {

:name => "Final battle"}

DESCRIPTIONS[:questid014] = {

:desc => " Shake was defeated.

Is this truly

the end?"

}

OBJECTIVES[:questid014] = {

:obj1 => ["Defeat Shake Williamspeare", 1] }

REWARDS[:questid014] = {

:gold => 1000,

:exp => 1000

}

#Side Quest 1

DETAILS[:sidequest001] = {

:name => "Old scientist"

}

DESCRIPTIONS[:sidequest001] = {

:qgiver\_name => "Old man",

:location => "Town",

:desc => " Old man asks our heroes

to collect for him 2 dead

slimes so he can produce

anti-chocolate potion to

give citizens means of

defense" }

OBJECTIVES[:sidequest001] = {

:obj1 => ["Kill slimes",2] }

REWARDS[:sidequest001] = {

:gold => 50,

:exp => 50

}

#Side Quest 2

DETAILS[:sidequest002] = {

:name => "Inn in need"

}

DESCRIPTIONS[:sidequest002] = {

:qgiver\_name => "Inn owner",

:location => "Merchant district",

:desc => " Old woman aks our heroes

to get rid of chocomonster

on her garden." }

OBJECTIVES[:sidequest002] = {

:obj1 => ["Kill chocobeaver",1] }

REWARDS[:sidequest002] = {

:gold => 75,

:exp => 75

}

end

class Game\_Quests

attr\_accessor :reset\_hash

def initialize

@quests = {}

QUEST::DETAILS.each do |id, quest|

@quests[id] = Quest.new(id,quest)

end

@reset\_hash = {}

@quests.each\_value do |quest|

@reset\_hash[quest.id] = {}

@reset\_hash[quest.id][:accepted] = false

@reset\_hash[quest.id][:turnedin] = false

quest.objectives.each do |id, obj|

@reset\_hash[quest.id][id] = obj

end

end

end

def check\_quests

@quests.each do |id, quest|

if !$game\_party.quests[id]

$game\_party.quests[id] = {}

quest.reset

end

end

end

def [](quest\_id)

return msgbox("No Quest with id " + quest\_id.to\_s) if @quests[quest\_id].nil?

@quests[quest\_id]

end

def []=(quest\_id, val)

@quests[quest\_id] = val

end

def quests

@quests

end

def no\_quests?

@quests.each do |id, quest|

return false if quest.accepted? && !quest.turned\_in

end

return true

end

def tracking?

$game\_party.tracking

end

def track\_quest(id)

return if $game\_party.tracking.include?(id)

$game\_party.tracking.push(id)

if $game\_party.tracking.size > $questlogmaxdisplay = 5

$game\_party.tracking.reverse!.pop

$game\_party.tracking.reverse!

end

end

def untrack\_quest(id)

return unless $game\_party.tracking.include?(id)

$game\_party.tracking.delete(id)

$game\_party.tracking.compact!

end

end

class Quest

attr\_accessor :name

attr\_accessor :level

attr\_accessor :id

attr\_accessor :desc

attr\_accessor :objectives

attr\_accessor :turned\_in

attr\_accessor :difficulty

attr\_accessor :qgiver\_name

attr\_accessor :location

attr\_accessor :auto\_complete

attr\_accessor :abandonable

attr\_accessor :force\_accept

attr\_accessor :force\_turnin

def initialize(id,quest\_hash)

@id = id

@level = 0

@difficulty = 0

@name = "No Quest Name"

@desc = ""

@qgiver\_name = 0

@location = 0

@auto\_complete = false

@abandonable = true

@need\_popup = false

@force\_turnin = false

@force\_accept = false

@name = quest\_hash[:name] if quest\_hash[:name]

@level = quest\_hash[:level] if quest\_hash[:level]

@force\_accept = quest\_hash[:force\_accept] if quest\_hash[:force\_accept]

@force\_turnin = quest\_hash[:force\_turnin] if quest\_hash[:force\_turnin]

@difficulty = quest\_hash[:difficulty] if quest\_hash[:difficulty]

@auto\_complete = quest\_hash[:auto\_complete] if quest\_hash[:auto\_complete]

@abandonable = quest\_hash[:abandonable] if !quest\_hash[:abandonable].nil?

@desc = QUEST::DESCRIPTIONS[id][:desc] if QUEST::DESCRIPTIONS[id][:desc]

@qgiver\_name = QUEST::DESCRIPTIONS[id][:qgiver\_name] if QUEST::DESCRIPTIONS[id][:qgiver\_name]

@location = QUEST::DESCRIPTIONS[id][:location] if QUEST::DESCRIPTIONS[id][:location]

@objectives = {}

if QUEST::OBJECTIVES[id]

QUEST::OBJECTIVES[id].each do |id, obj|

@objectives[id] = Objective.new(id, obj)

end

else

msgbox("Quest " + id.to\_s + " has no objectives.")

end

@reward\_gold = 0

@reward\_exp = 0

@scale\_exp = 0

@reward\_items = []

begin

if QUEST::REWARDS[id][:gold]

@reward\_gold = QUEST::REWARDS[id][:gold]

end

if QUEST::REWARDS[id][:exp]

@reward\_exp = QUEST::REWARDS[id][:exp]

@scale\_exp = QUEST::REWARDS[id][:scale\_exp] if QUEST::REWARDS[id][:scale\_exp]

end

if QUEST::REWARDS[id][:items]

@reward\_items = QUEST::REWARDS[id][:items]

end

rescue

msgbox(id.to\_s + " has no defined REWARDS. This is not optional.")

end

end

def accept

reset

$game\_party.quests[id][:accepted] = true

track\_quest

$game\_map.need\_refresh = true

Audio.se\_play("Audio/SE/Book2")

end

def abandon

reset

$game\_party.quests[id][:accepted] = false

end

def fail

Audio.me\_play("Audio/ME/Gag")

abandon

end

def accepted?

$game\_party.quests[id][:accepted]

end

def accepted

accepted?

end

def completed?

@objectives.each do |id, obj|

return false if !$game\_party.quests[@id][id].completed?

end

return true

end

def force\_done

$game\_party.quests[id][:accepted] = true

@objectives.each do |id, obj|

$game\_party.quests[@id][id].current = obj.max

end

turnin

end

def reset

$game\_party.quests[id][:accepted] = false

@objectives.each do |id, obj|

$game\_party.quests[@id][id] = obj

$game\_party.quests[@id][id].current = 0

end

$game\_party.quests[id][:turnedin] = false

end

def objective(id)

return Objective.new(id, ["No Objective Found",0]) if @objectives[id].nil?

$game\_party.quests[@id][id]

end

def set\_obj(id, value)

objective(id).current = value

@need\_popup = false if !completed?

popup if completed? && !@need\_popup

turnin if completed? && @auto\_complete

$game\_map.need\_refresh = true

end

def adv\_obj(id, value)

objective(id).current += value

@need\_popup = false if !completed?

popup if completed? && !@need\_popup

turnin if completed? && @auto\_complete

$game\_map.need\_refresh = true

end

def reward\_gold

@reward\_gold

end

def reward\_exp

get\_mod\_exp.to\_i

end

def reward\_items

@reward\_items

end

def turnin

$game\_party.quests[id][:turnedin] = true

untrack\_quest

$game\_map.need\_refresh = true

$game\_party.gain\_gold(@reward\_gold)

$game\_party.members.each do |actor|

actor.gain\_exp(@reward\_exp)

end

@reward\_items.each do |array|

item = $data\_items[array[1]] if array[0] == :item

item = $data\_weapons[array[1]] if array[0] == :weapon

item = $data\_armors[array[1]] if array[0] == :armor

$game\_party.gain\_item(item, array[2])

end

end

def track\_quest

$game\_quests.track\_quest(@id)

end

def untrack\_quest

$game\_quests.untrack\_quest(@id)

end

def can\_abandon?

@abandonable

end

def popup

@need\_popup = true

Audio.me\_play("Audio/ME/Item")

if Module.const\_defined?(:Popup)

Popup.add([@name + ' complete!'])

end

end

def turned\_in?

$game\_party.quests[id][:turnedin]

end

def turned\_in

turned\_in?

end

def active?

accepted? && !completed?

end

def get\_mod\_exp

pval = @scale\_exp \* (@level - $game\_party.highest\_level).to\_f / 100 + 1

@reward\_exp \* pval

end

end

class Objective

attr\_accessor :id

attr\_accessor :name

attr\_accessor :current

attr\_accessor :max

attr\_accessor :hidden

def initialize(id, obj)

@name = obj[0]

@current = 0

@max = obj[1]

@hidden = obj[2] ? obj[2] : false

end

def completed?

@current >= @max

end

end

module DataManager

class << self

alias quest\_cgo load\_database

alias quest\_sng setup\_new\_game

end

def self.load\_database

quest\_cgo

$game\_quests = Game\_Quests.new

end

def self.setup\_new\_game

$game\_quests = Game\_Quests.new

quest\_sng

end

end

class Scene\_Quest < Scene\_MenuBase

def start

super

@help\_window = Window\_Help.new(1)

@help\_window.set\_text("Quest Log")

@list\_window = Window\_SceneList.new

@list\_window.set\_handler(:cancel, method(:list\_cancel))

@list\_window.set\_handler(:ok, method(:list\_ok))

@list\_window.refresh

@list\_window.activate

@list\_window.select(0)

@detail\_window = Window\_SceneDetail.new

@command\_window = Window\_QuestTrack.new

@command\_window.x = Graphics.width / 2 - @command\_window.width / 2

@command\_window.y = Graphics.height / 2 - @command\_window.height / 2

@command\_window.set\_handler(:track, method(:track))

@command\_window.set\_handler(:untrack, method(:untrack))

@command\_window.set\_handler(:abandon, method(:abandon))

@command\_window.set\_handler(:cancel, method(:command\_cancel))

end

def update

super

@detail\_window.quest = @list\_window.current\_item

end

def list\_cancel

SceneManager.return

end

def list\_ok

@command\_window.quest(@list\_window.current\_item)

@command\_window.refresh

@command\_window.select(0)

@command\_window.activate

@command\_window.open

end

def track

$game\_quests.track\_quest(@list\_window.current\_item.id)

command\_cancel

end

def untrack

$game\_quests.untrack\_quest(@list\_window.current\_item.id)

command\_cancel

end

def abandon

@list\_window.current\_item.abandon

command\_cancel

end

def command\_cancel

@command\_window.close

@list\_window.refresh

@list\_window.activate

list\_cancel if $game\_quests.no\_quests?

end

end

class Window\_SceneList < Window\_Selectable

def initialize

super(0,48,Graphics.width/5\*2,Graphics.height-48)

refresh

end

def make\_item\_list

@data = []

$game\_quests.quests.each do |id, quest|

@data.push(quest) if quest.accepted? && !quest.turned\_in?

end

@data.push(nil) if @data.empty?

end

def draw\_item(index)

contents.font.size = 18

item = @data[index]

if item

rect = item\_rect(index)

rect.width -= 4

if $game\_quests.tracking?.include?(item.id)

text = "\*" + item.name

else

text = item.name

end

draw\_text(rect, text)

draw\_text(rect, "Lv" + item.level.to\_s,2) if item.level > 0

end

end

def col\_max; 1; end

def current\_item

@data[@index]

end

def current\_item\_enabled?

true

end

def refresh

make\_item\_list

create\_contents

draw\_all\_items

end

def item\_max

@data ? @data.size : 0

end

end

class Window\_SceneDetail < Window\_Base

def initialize

super(Graphics.width/5\*2,48,Graphics.width-Graphics.width/5\*2,Graphics.height-48)

end

def quest=(quest)

return if @quest == quest

@quest = quest

refresh

end

def refresh

contents.clear

return unless @quest

contents.font.size = 18

change\_color(system\_color)

draw\_text(0,0,contents.width,line\_height,@quest.qgiver\_name) if @quest.qgiver\_name != 0

draw\_text(0,0,contents.width,line\_height,@quest.location,2) if @quest.location != 0

change\_color(normal\_color)

@quest.qgiver\_name != 0 || @quest.location != 0 ? yy = line\_height : yy = 0

draw\_text\_ex(0,yy,@quest.desc)

change\_color(system\_color)

draw\_text(0,line\_height\*7,contents.width,24,"Objectives:")

change\_color(normal\_color)

yy = line\_height \* 8

@quest.objectives.each do |id, obj|

next if obj.hidden

draw\_objective(yy, obj)

yy += 24

end

change\_color(system\_color)

draw\_text(0,yy,contents.width,line\_height,"Rewards:")

yy += line\_height

if @quest.reward\_exp > 0

draw\_text(6,yy,contents.width/2,line\_height,"XP: ")

change\_color(normal\_color)

draw\_text(36,yy,contents.width/2,line\_height,@quest.reward\_exp)

yy += line\_height

end

if @quest.reward\_gold > 0

change\_color(normal\_color)

draw\_text(6,yy,contents.width/2,line\_height,@quest.reward\_gold.to\_s)

cx = text\_size(@quest.reward\_gold).width

change\_color(system\_color)

draw\_text(6+cx,yy,contents.width/2,line\_height,Vocab::currency\_unit)

end

yy += line\_height

change\_color(normal\_color)

@quest.reward\_items.each do |array|

item = $data\_items[array[1]] if array[0] == :item

item = $data\_weapons[array[1]] if array[0] == :weapon

item = $data\_armors[array[1]] if array[0] == :armor

draw\_item\_name(item, 6, yy, true, contents.width)

if array[2] > 1

draw\_text(6+text\_size(item.name).width+36,yy,48,24,"x"+array[2].to\_s)

end

yy += line\_height

end

if @quest.difficulty != 0

text = "Difficulty: " + @quest.difficulty

draw\_text(0,contents.height-line\_height,contents.width,line\_height,text,2)

end

end

def draw\_objective(yy, obj)

draw\_text(6,yy,contents.width,24,obj.name)

draw\_text(0,yy,contents.width,24,obj.current.to\_s+"/"+obj.max.to\_s,2)

end

def reset\_font\_settings

change\_color(normal\_color)

contents.font.bold = Font.default\_bold

contents.font.italic = Font.default\_italic

end

end

class Window\_QuestTrack < Window\_Command

def initialize

super(0,0)

self.openness = 0

end

def quest(quest)

@quest = quest

end

def make\_command\_list

return unless @quest

if !$game\_quests.tracking?.include?(@quest.id)

add\_command("Track Quest", :track)

else

add\_command("Untrack Quest", :untrack)

end

add\_command("Abandon Quest", :abandon, @quest.can\_abandon?)

end

def window\_height

fitting\_height(2)

end

end

class Window\_MenuCommand

alias quest\_aoc add\_original\_commands

def add\_original\_commands

quest\_aoc

add\_command("Quest Log", :quest, !$game\_quests.no\_quests?)

end

end

class Scene\_Menu

alias quest\_ccw create\_command\_window

def create\_command\_window

quest\_ccw

@command\_window.set\_handler(:quest, method(:scene\_quest))

end

def scene\_quest

SceneManager.call(Scene\_Quest)

end

end

class Scene\_Map

alias quest\_start start

alias quest\_update update

def start

quest\_start

@quest\_log = Window\_QuestLog.new

@quest\_confirm = Window\_QuestConfirm.new

@quest\_confirm.set\_handler(:accept, method(:confirm\_accept))

@quest\_confirm.set\_handler(:decline, method(:confirm\_cancel))

@quest\_confirm.set\_handler(:cancel, method(:confirm\_cancel))

@quest\_turnin = Window\_QuestTurnin.new

@quest\_turnin.set\_handler(:accept, method(:turnin\_accept))

@quest\_turnin.set\_handler(:decline, method(:confirm\_cancel))

@quest\_turnin.set\_handler(:cancel, method(:confirm\_cancel))

@quest\_apply = Window\_QuestApply.new(@quest\_confirm,@quest\_turnin)

end

def update(\*args)

@quest\_log = Window\_QuestLog.new if @quest\_log.disposed?

quest\_update(\*args)

end

def show\_quest(id, turnin = false)

@quest\_apply.show($game\_quests[id],turnin)

end

def accepting?

@quest\_confirm.active || @quest\_turnin.active

end

def confirm\_accept

@quest\_apply.accept

@quest\_apply.hide

end

def confirm\_cancel

@quest\_apply.hide

end

def turnin\_accept

@quest\_apply.turnin

@quest\_apply.hide

end

def update\_call\_menu

if $game\_system.menu\_disabled || $game\_map.interpreter.running? || accepting?

@menu\_calling = false

else

@menu\_calling ||= Input.trigger?(:B)

call\_menu if @menu\_calling && !$game\_player.moving?

end

end

end

class Scene\_Base

def accepting?

false

end

end

class Window\_QuestLog < Window\_Base

def initialize

super(Graphics.width/5\*3,0,Graphics.width/5\*2,Graphics.height)

self.x = 0 if QUEST\_LOG\_POSITION == 1

self.x += QUEST\_LOG\_OFFSET\_X

self.y += QUEST\_LOG\_OFFSET\_Y

self.opacity = 0

self.contents.font.size = 18

end

def update

super

return unless Graphics.frame\_count % 20 == 0

self.visible = $questlogvisibility

return unless self.visible

self.visible = !$game\_quests.no\_quests?

self.visible = $game\_quests.tracking?.size > 0

return unless self.visible

contents.clear

change\_color(crisis\_color)

draw\_text(0,0,contents.width,18,"Quest Log:",1)

yy = 18;iter = 0

$game\_quests.tracking?.each do |id|

quest = $game\_quests[id]

next unless quest.accepted? && !quest.turned\_in

change\_color(system\_color)

draw\_text(6,yy,contents.width-6,18,quest.name)

change\_color(normal\_color)

yy += 18

quest.objectives.each do |obj\_id, obj|

next if obj.hidden

draw\_objective(yy, $game\_party.quests[id][obj\_id])

yy += 18

end

iter += 1

end

end

def draw\_objective(yy, obj)

draw\_text(0,yy,contents.width-24,18,obj.name)

draw\_text(0,yy,contents.width,18,obj.current.to\_s+"/"+obj.max.to\_s,2)

end

end

class Window\_QuestApply < Window\_Base

def initialize(confirm\_window, turnin\_window)

super(Graphics.width/8,Graphics.width/8,Graphics.width/5\*3,Graphics.height-Graphics.width/8\*2)

self.openness = 0

@confirm\_window = confirm\_window

@turnin\_window = turnin\_window

self.contents.font.size = 18

end

def refresh

return unless @quest

contents.clear

change\_color(system\_color)

yy = 0

if @quest.qgiver\_name != 0

draw\_text(0,0,contents.width/2,line\_height,@quest.qgiver\_name)

yy = line\_height

end

if @quest.location != 0

draw\_text(contents.width/2,0,contents.width/2,line\_height,@quest.location,2)

yy = line\_height

end

change\_color(crisis\_color)

draw\_text(0,yy,contents.width,line\_height,"Lvl: " + @quest.level.to\_s) if @quest.level > 0

draw\_text(0,yy,contents.width,line\_height,@quest.name,1)

draw\_text(0,yy,contents.width,line\_height,@quest.difficulty,2) if @quest.difficulty != 0

change\_color(normal\_color)

draw\_text\_ex(0,line\_height+yy,@quest.desc)

change\_color(system\_color)

draw\_text(0,line\_height\*8,contents.width,line\_height,"Objectives:")

change\_color(normal\_color)

yy = line\_height \* 9

@quest.objectives.each do |obj\_id, obj|

next if obj.hidden

draw\_objective(yy, $game\_party.quests[@quest.id][obj\_id])

yy += line\_height

end

change\_color(system\_color)

draw\_text(0,yy,contents.width,line\_height,"Rewards:")

yy += line\_height

if @quest.reward\_exp > 0

draw\_text(6,yy,contents.width/2,line\_height,"XP: ")

change\_color(normal\_color)

draw\_text(36,yy,contents.width/2,line\_height,@quest.reward\_exp)

yy += line\_height

end

if @quest.reward\_gold > 0

change\_color(normal\_color)

draw\_text(6,yy,contents.width/2,line\_height,@quest.reward\_gold.to\_s)

cx = text\_size(@quest.reward\_gold).width

change\_color(system\_color)

draw\_text(6+cx,yy,contents.width/2,line\_height,Vocab::currency\_unit)

end

yy += line\_height

change\_color(normal\_color)

@quest.reward\_items.each do |array|

item = $data\_items[array[1]] if array[0] == :item

item = $data\_weapons[array[1]] if array[0] == :weapon

item = $data\_armors[array[1]] if array[0] == :armor

draw\_item\_name(item, 6, yy, true, contents.width)

if array[2] > 1

draw\_text(6+text\_size(item.name).width+36,yy,48,24,"x"+array[2].to\_s)

end

yy += line\_height

end

end

def reset\_font\_settings

change\_color(normal\_color)

contents.font.bold = Font.default\_bold

contents.font.italic = Font.default\_italic

end

def line\_height

18

end

def draw\_objective(yy, obj)

draw\_text(6,yy,contents.width,24,obj.name)

draw\_text(0,yy,contents.width,24,obj.current.to\_s+"/"+obj.max.to\_s,2)

end

def show(quest,turnin)

@quest = quest

return if @quest.turned\_in

refresh

open

@confirm\_window.quest(@quest)

@confirm\_window.open if !turnin

if turnin

@turnin\_window.quest(@quest)

@turnin\_window.open

end

end

def hide

close

@confirm\_window.close

@turnin\_window.close

end

def accept

@quest.accept

end

def turnin

@quest.turnin

end

end

class Window\_QuestConfirm < Window\_HorzCommand

def initialize

super(Graphics.width/8,Graphics.width/8+Graphics.height-Graphics.width/8\*2)

self.openness = 0

self.active = false

@enabled = true

refresh

end

def window\_width

Graphics.width/5\*2

end

def window\_height

48

end

def make\_command\_list

add\_command("Accept",:accept)

add\_command("Decline",:decline, @enabled)

end

def item\_width

width / 2 - padding \* 2

end

def open

super

activate

select(0)

end

def quest(quest)

@quest = quest

@enabled = !@quest.force\_accept

refresh

end

def cancel\_enabled?

super && @enabled

end

end

class Window\_QuestTurnin < Window\_QuestConfirm

def quest(quest)

@quest = quest

@enabled = true

@enabled = !@quest.completed? if @quest.force\_turnin

refresh

end

def make\_command\_list

return unless @quest

add\_command("Complete",:accept,@quest.completed? && !@quest.turned\_in)

add\_command("Cancel",:decline, @enabled)

end

end

class Game\_Party

attr\_accessor :quests

attr\_accessor :tracking

alias quests\_init initialize

def initialize(\*args)

quests\_init(\*args)

@quests = $game\_quests.reset\_hash unless $game\_quests.nil?

@tracking = []

end

end

class Game\_Player

alias quest\_update update

def update

return if SceneManager.scene.accepting?

quest\_update

end

end

class Game\_Event

def obj(quest, objective)

$game\_quests[quest].objective(objective).current

end

end

class Game\_Interpreter

def accept\_quest(quest)

$game\_quests[quest].accept

end

def ask\_accept(quest)

return unless SceneManager.scene.is\_a?(Scene\_Map)

SceneManager.scene.show\_quest(quest)

Fiber.yield while SceneManager.scene.accepting?

end

def abandon\_quest(quest)

$game\_quests[quest].abandon

end

def fail\_quest(quest)

$game\_quests[quest].fail

end

def turnin\_quest(quest)

$game\_quests[quest].turnin

end

def ask\_turnin(quest)

return unless SceneManager.scene.is\_a?(Scene\_Map)

SceneManager.scene.show\_quest(quest,true)

Fiber.yield while SceneManager.scene.accepting?

end

def adv\_obj(quest, objective, value)

$game\_quests[quest].adv\_obj(objective, value)

end

def set\_obj(quest, objective, value)

$game\_quests[quest].set\_obj(objective, value)

end

def obj(quest, objective)

$game\_quests[quest].objective(objective).current

end

def hide\_obj(quest, objective)

$game\_quests[quest].objective(objective).hidden = true

end

def show\_obj(quest, objective)

$game\_quests[quest].objective(objective).hidden = false

end

end

module DataManager

class << self

alias quest\_load\_game load\_game

end

def self.load\_game(index)

quest\_load\_game(index)

$game\_quests.check\_quests

end

end