```
"""
#~ /***************************************************************************
 #~ *            numberhunt.py
 #~ *
 #~ *  Sat Jul 10 15:42:33 2004
 #~ *  Copyright  2004  Stas
 #~ *  stas@linux.isbeter.nl
 #~ ****************************************************************************/
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Library General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

## Numberhunt is based on the classic Atari 2600 game, Pitfall.
"""
# Colors used
### trees at the top of the screen
# tree leafs =  39,176,8
# tree branch = 188,150,18
### overall background
# green background = 135,227,113
# brown floor = 255,197,29
# yellow = 244,236,16

RCFILE = 0
_NHDEBUG = 1

import os,sys,random
from math import sin,cos,pi
import pygame
from pygame.constants import *
from utils import trace_error,MyError,load_image,load_music,char2surf,load_sound
from SpriteUtils import CPSprite,CPGroup,CPinit
import Timer

class Img:
    """ Container to store image objects"""
    pass
class Snd:
    """ Container to store sound objects"""
    pass
class Misc:
    """ Container to store all kind of stuff"""
    pass

class Croc(CPSprite):
    def __init__(self,pos,activate=0):
        CPSprite.__init__(self)
        self.image0 = Img.croc_close
```



```
        self.image1 = Img.croc_open
        self.image = self.image1
        self.switch = 8
        self.rect = self.image.get_rect()
        self.rect.move_ip(pos)
        self.display_sprite()
        if activate:
            self.activate
    def activate(self):
        self.add(Misc.actives)
    def on_update(self,*args):
        if self.image is self.image0 and self.rect.colliderect(Misc.harry.rect):
            return# if harry is on us, do nothing
        if self.switch < 1:
            if self.image is self.image0:
                self.image = self.image1
                self.add(Misc.hazards)
            else:
                self.image = self.image0
                self.remove(Misc.hazards)

            self.switch = 100
        else:
            self.switch -= 1

class Wrapper(CPSprite):
    """This is just a CPSprite wrapper for a sprite so that we can add it to the
       hazard group."""
    def __init__(self,img,pos,activate=0):
        CPSprite.__init__(self)
        self.image = img
        self.rect = self.image.get_rect()
        self.rect.move_ip(pos)
        if activate:
            self.activate()
    def activate(self):
        self.display_sprite()
        self.add(Misc.hazards)

class Rope:
    """This is NOT a CPSprite object so you have to call the update method
    manually from within the event loop."""
    def __init__(self,pos,screen,backgr,delay=4):
        x,y = pos
        self.pos = pos
        self.screen = screen
        self.background = backgr
        self.delay_org = delay
        self.delay = delay
        self.with = 4 # dia of the rope
        self.color = (255,197,29)
        swing = 70
        lng = 300
        self.swingleft,self.swingright =[],[]
        for t in range(-swing,swing,3):
            alpha = sin(t * pi / 200) * pi / 4
            a,b = (x + lng * sin(alpha) ,y + lng * cos(alpha))
            self.swingleft.append((int(a),int(b)))
            self.swingright.insert(0,(int(a),int(b)))
```

```
        self.endpoints = self.swingleft[:]
        self.kind_used = 'left'
        self.drawline = pygame.draw.line
        self.display = pygame.display.update
        # This is just to be able to call update for the first time
        self.dirtyrect = (0,0,1,1)
    def __del__(self):
        try:
            self.display(self.screen.blit(self.background,self.dirtyrect,self.dirtyrect))
        except:
            pass
    def update(self,*args):
        self.delay -= 1
        if self.delay > 0: return
        else:
            self.delay = self.delay_org
        try:
            endpos = self.endpoints.pop()
        except IndexError,info:
            if _NHDEBUG: print self.__class__,info
            if self.kind_used == 'left':
                self.kind_used = 'right'
                self.endpoints = self.swingright[:]
            else:
                self.kind_used = 'left'
                self.endpoints = self.swingleft[:]
        else:
            self.oldrect = self.screen.blit(self.background,self.dirtyrect,self.dirtyrect)
            #self.display(self.oldrect)
            self.dirtyrect = self.drawline(self.screen,self.color,self.pos,endpos,self.with)
            self.display((self.oldrect,self.dirtyrect))

    def get_swing(self):
        """used to get the end points of the rope currently in use.
         Used to place the swinging harry."""
        return self.endpoints

class Boulder(CPSprite):
    def __init__(self,x=0,activate=0):
        CPSprite.__init__(self)
        self.image0 = Img.boulder
        self.image1 = Img.boulder1
        self.image = self.image0
        self.rect = self.image.get_rect()
        pos = (x,340)
        # setup the movement of this object
        # The dokill arg will remove this sprite from every group it belongs
        self.set_movement(start=pos,end=(800,pos[1]),step=2,dokill=1)
        self.sound = load_sound(Snd.bouldersoundfile)
        self.switch = 4
        self.soundplay = 0
        if activate:
            self.activate
    def __del__(self):
        try:
            self.sound.stop()
        except:
            pass
```



```
    def activate(self):
        self.add((Misc.actives,Misc.hazards))
    def on_update(self,*args):
        if not self.soundplay:
            self.sound.play(-1)
            self.soundplay = 1
        if self.switch < 1:
            if self.image is self.image0:
                self.image = self.image1
            else:
                self.image = self.image0
            self.switch = 4
        else:
            self.switch -= 1
        # we move accoording to the movement set in the constructor.
        # As we have set the 'dokill' argument we don't care about removing
        # The sound is stop by the destructor
        v = self.moveit()
        if v != -1: self.sound.stop()

class Number(CPSprite):
    def __init__(self,number,place,x=400,y=340,activate=0):
        """number = 'int', place = which position in the display (0,1,2,3),
        x,y = position in the level, activate = show it or not"""
        CPSprite.__init__(self)
        self.place = place
        self.display_positions = {0:(200,430),1:(250,430),2:(300,430),3:(350,430)}
        pos = (x,y)
        color = (29,254,255)
        self.image = char2surf(number,48,color)
        self.rect = self.image.get_rect()
        self.rect.move_ip(pos)
        self.number = number
        if activate:
            self.activate()
    def activate(self):
        self.display_sprite()
        self.add(Misc.hazards)
    def on_update(self,*args):
        pass
    def __str__(self):
        return self.number
    def __eq__(self,other):
        if other == 'number': return True
        else: return False
    def __ne__(self,other):
        if other != 'number': return True
        else: return False
    def place_it(self):
        self.erase_sprite()
        self.rect = pygame.Rect\
                    (self.display_positions[self.place]+self.image.get_size())
        self.display_sprite()

class MusicButton(CPSprite):
    def __init__(self,img_on,img_off,music,pos):
        CPSprite.__init__(self)
        self.image_on = img_on
```

```
        self.image_off = img_off
        self.musicfile = music
        self.image = self.image_on
        self.rect = self.image.get_rect()
        self.rect.move_ip(pos)
        self.connect_callback(self.callback, event_type=MOUSEBUTTONDOWN)
    def callback(self,*args):
        if self.image is self.image_on:
            self.off()
        else:
            self.on(-1)
    def on(self,loop=0):
        self.music = load_music(self.musicfile)
        self.music.play(loop)
        self.image = self.image_on
    def off(self):
        self.image = self.image_off
        try:
            self.music.stop()
        except:
            pass

class Harry(CPSprite):
    """Meet Harry, the hero of the game. This object runs, jumps, reakt to key presses
     and detect any collision with the objects from the Misc.hazards group."""
    def __init__(self,img1,img2,img3,img4,pos=(760,270)):
        """img1 to img4 are the different poses harry can asume.
        pos is the start position of our hero."""
        CPSprite.__init__(self)
        self.image1 = img1
        self.image2 = img2
        self.image3 = img3
        self.image4 = img4
        self.startpos = pos
        #self.image = self.image1#start image
        #self.rect = self.image.get_rect().move(self.startpos)
        self.reset()
        self.startrect = self.rect.move(0,0)#just a copy
        self.connect_callback(self.callback, event_type=KEYDOWN)
        self.delay = 2
        self.x = -12
        self.up = 32
        #self.run,self.jump = None,None
        self.collide = pygame.sprite.spritecollideany#this will speed up the collision detection
    def reset(self):
        """reset harry to his starting position"""
        self.run,self.jump = None,None
        self.image = self.image1#start image
        self.rect = self.image.get_rect().move(self.startpos)
        if _NHDEBUG: print 'harry rect reset to',self.rect
    def on_update(self,*args):
        """Called by the group refresh methods."""
        obj = self.collide(self,Misc.hazards)
        if obj:
            if obj != 'number':
                print "AAAAAARGH"
                self.kill()
                return (-2,0)#Harry is dead :-(
            else:
```



```
                obj.place_it()#this will place the number in the bottom bar
        if self.rect.left < 0:
            self.kill()
            return (-1,0)#we left the screen
        if self.delay:
            self.delay -= 1
            return
        else: self.delay = 4
        if self.run:
            self.image = self.run.pop()
            self.rect.move_ip(self.step,0)
        elif self.jump:
            self.rect.move_ip(0,self.up)
            self.jump = None

    def callback(self,*args):
        """ This is called when the sprite class update method is called.
        Remember that you must 'connect' a callback first with a call to
        self.connect_callback, see the CPSprite reference for info on possibilities
        of connecting callbacks .
        """
        if self.run: return# are we still running, we should first come to a stop
        c = args[1].key
        #print "key =",c
        if c == K_LEFT:
            self.run = [self.image1,self.image3,self.image2]
            self.step = -12
        elif c == K_UP and not self.jump:
            self.rect.move_ip(0,-self.up)
            self.jump = 1
            self.step = -16 - (Misc.crocs*8)
            self.run = [self.image1,self.image4,self.image4,self.image4,self.image4]
            print "jump",self.step

class Game(Img,Snd):
    """  Findsound - part of childsplay.py, a suite of educational games for
  young children. """
    def __init__(self,screen,backgr,rc_dic,basepath,libdir):
        self.screen = screen
        self.backgr = backgr
        Img.screen = self.screen
        Img.backgr = self.backgr
        self.rc_dic = rc_dic
        self.basedir  = basepath
        self.libdir = libdir
        self.gamelevels =(1,2,3,4)
        self.gameitems = (range(20))
        Misc.actives = CPinit(self.screen,self.backgr)
        # used as a container to hold all the stuff harry should avoid
        Misc.hazards = pygame.sprite.Group()
        self._setup()

    def __del__(self):
        """If you use the timer object you MUST stop it."""
        try:
            self.timer.stop()
```

```
        except:
            pass

    def _setup(self):
        """ Set all the stuff we need"""
        trees = load_image(os.path.join(self.libdir,'NumberhuntData','tr
ees1.png'))
        s = pygame.Surface((800,500)).convert()
        s.fill((39,176,8),(0,0,800,30))#leafs
        s.fill((135,227,113),(0,30,800,300))#between trees
        s.fill((255,197,29),(0,332,800,80))#floor
        s.blit(trees,(0,30))
        self.startscreen = s

        self.numbercol = (135,227,113)
        harry_still = load_image(os.path.join(self.libdir,'NumberhuntDat
a','harry_still.png'),1)
        harry_run1 = load_image(os.path.join(self.libdir,'NumberhuntData
','harry_run1.png'),1)
        harry_run2 = load_image(os.path.join(self.libdir,'NumberhuntData
','harry_run2.png'),1)
        harry_jump = load_image(os.path.join(self.libdir,'NumberhuntData
','harry_jump_up.png'),1)
        Misc.harry = Harry(harry_still,harry_run1,harry_run2,harry_jump)
        Img.boulder = load_image(os.path.join(self.libdir,'NumberhuntDat
a','boulder.png'),1)
        Img.boulder1 = load_image(os.path.join(self.libdir,'NumberhuntDa
ta','boulder1.png'),1)
        Img.pool = load_image(os.path.join(self.libdir,'NumberhuntData',
'pool.png'),0)
        Img.pool_left = load_image(os.path.join(self.libdir,'NumberhuntD
ata','pool_left.png'),0)
        Img.pool_right = load_image(os.path.join(self.libdir,'Numberhunt
Data','pool_right.png'),0)
        Img.croc_open = load_image(os.path.join(self.libdir,'NumberhuntD
ata','croc_open.png'),1)
        Img.croc_close = load_image(os.path.join(self.libdir,'Numberhunt
Data','croc_close.png'),1)
        # Snd.bouldersoundfile is just the filename not a sound object b
ecause we can have multiple
        # boulders. Each boulder controls the start and stop of it's sou
nd.
        Snd.bouldersoundfile = os.path.join(self.libdir,'NumberhuntData'
,'boulder.wav')
        Snd.startwinkle = load_sound(os.path.join(self.libdir,'Numberhun
tData','chimes.wav'))
        music = os.path.join(self.libdir,'NumberhuntData','Numberhunt.og
g')
        but_on = load_image(os.path.join(self.libdir,'NumberhuntData','m
usic_on.png'),1)
        but_off = load_image(os.path.join(self.libdir,'NumberhuntData','
music_off.png'),1)
        pos = (640,430)
        self.music_button = MusicButton(but_on,but_off,music,pos)
        self.music_button.display_sprite()
        self.music_button.on(-1)
        Misc.actives.add(self.music_button)

    def __str__(self):
        """Must return the original, not translated, title of this game.
        It's needed by the high score class of childsplay."""
        return "Numberhunt"
```



```

    def reset(self):
        """Reset everything after a level is finished"""
        for obj in Misc.actives.sprites():
            obj.kill()
        for obj in Misc.hazards.sprites():
            obj.kill()
        pygame.event.pump()
        Misc.harry.reset()

    def start(self,levels,fields):
        """Help string displayed when this game is started."""
        #~ self.screen.blit(self.startscreen,(0,0))
        #~ self.backgr.blit(self.startscreen,(0,0))
        self.stop = 0
        self.level = levels#used for debugging messages

        if _NHDEBUG: print "Entring level",self.level
        #~ self.reset()#this will reset the scene
        #~ Misc.harry.display_sprite()
        #~ Misc.crocs = 0#to be set so we can test it for jump distance.
        #~ self.rope = None# used by loop to test if we have a rope
        if levels == 1:
            exerc,self.solution = apply(random.choice(\
                                    (self._add,self._min)),(5,)\
                                    ).split('=')
            self.level1_setup(exerc)
            #~ self.rope = Rope((400,0),self.screen,self.backgr)
            #~ self.number = Number('20')
            #~ self.number.display_sprite()
        elif levels == 2:
            self.rope = None
            self.exerc,self.solution = apply(random.choice(\
                                    (self._add,self._min)),(10,)\
                                    ).split('=')
            self.level2_setup()
            #~ Boulder()#adds itself to actives and hazard
            #~ Boulder(x=-300)

        elif levels == 3:
            self.exerc,self.solution = apply(random.choice(\
                                    (self._add,self._min,self._mul,self.
_div)),\
                                    (10,)\
                                    ).split('=')
            self.level3_setup()
            #~ self._make_croc_pool()
            #~ Misc.crocs = 1
        if levels == 4: pass #XXX perhaps some stuff as end level??
        #~ self.exerc = self.exerc + " = ?"
        #~ s = char2surf(self.exerc,64,(135,227,113))
        #~ self.screen.blit(s,(100,420))
        self.restart()
        pygame.display.update()
        if _NHDEBUG: print "starting field, active sprites:", Misc.activ
es.sprites()

    def level1_setup(self,exerc):
        """Misc.fields should become a list with tuples containing the o
bjects which
        can be started by calling their activate method."""
        Misc.fields = [(Boulder(),)]
```

```
        i = 0
        for c in exerc:
            Misc.fields.append((Number(c,i),))
            i += 1
        random.shuffle(Misc.fields)

    def level2_setup(self):
        pass
    def level3_setup(self):
        pass

    def restart(self):
        """We use this to restart the game after a 'field' is cleared.
        Because childsplay uses a number of levels we use 'restart' to execute
        extra 'levels' inside a level."""
        if _NHDEBUG: print "Restarting level",self.level,"fields",Misc.fields
        if not Misc.fields:
            self.stop = -1#this will stop the eventloop and childsplay will start the next (real) level
            return
        self.screen.blit(self.startscreen,(0,0))
        self.backgr.blit(self.startscreen,(0,0))
        self.reset()#this will reset the scene
        Misc.harry.display_sprite()
        Misc.actives.add(Misc.harry)
        Misc.crocs = 0#to be set so we can test it for jump distance.
        self.rope = None# used by loop to test if we have a rope
        for obj in Misc.fields.pop():
            if _NHDEBUG: print obj.__class__
            obj.activate()

    def harry_dead(self):
        pass

    def loop(self,events):
        item,self.score = None,0
        for event in events:
            if event.type is KEYDOWN or MOUSEBUTTONDOWN:
                item=event
                break
        try:
            if _NHDEBUG == 2 and item: print "loop event",item
            v = Misc.actives.refresh(item)# This can return None or [(object name,(flag,score))]
        except:
            print trace_error()
            raise MyError
        if v:
            if _NHDEBUG: print "v=",v
            stop = v[0][1][0]
            if stop == -1: self.restart()
            if stop == -2: self.harry_dead()
            self.score = v[0][1][1]
            if _NHDEBUG: print "stop/score",self.stop,self.score
        if self.rope:# call this manually, because rope is NOT a CPSprite
            self.rope.update()
        return self.stop,self.score

 #### methods for internal use #########
```



```
    def _add(self,i):
        rand = random.choice
        a = rand(range(1,i+1))
        b = rand(range(1,i+1))
        c = a+b
        answer = "%s+%s=%s" % (a,b,c)
        return answer
    def _min(self,i):
        rand = random.choice
        b = rand(range(1,i+1))
        a = rand(range(b,i+1))
        c = a-b
        answer = "%s-%s=%s" % (a,b,c)
        return answer
    def _mul(self,i):
        rand = random.choice
        a = rand(range(1,i+1))
        b = rand(range(1,i+1))
        c = a*b
        answer = "%sx%s=%s" % (a,b,c)
        return answer
    def _div(self,i):
        s = self._mul(i)
        a = int(s[0])
        b = int(s[-1])
        c = b/a
        answer = "%s/%s=%s" % (a,b,c)
        return answer
    def _make_croc_pool(self):
        x,y = 124,348
        s = pygame.Surface((401,45))
        s.fill((62,148,33))
        # Wrapper and Croc objects add themselfs to hazard group and/or display themself
        Wrapper(Img.pool_left,(x,y))
        x += 80
        self.screen.blit(s,(x,y))
        self.backgr.blit(s,(x,y))
        for i in range(0,2):
            Croc((x,y-8))
            x += 80
            Wrapper(Img.pool,(x,y))
            x += 80
        Croc((x,y-8))
        x += 80
        Wrapper(Img.pool_right,(x,y))
 ############
    def helptitle(self):
        return _("Numberhunt")

    def help(self):
        text = [_("The aim of the game:"),
        _("The game is a based on the old Atari game called 'Pitfall'."),\
        _("You must walk to the left of the screen by using the arrow keys."),\
        _("First you come accross two numbers and a operator like '+,-,: or x'."),\
        _("These numbers will form a mathematic exercise which you have to solve."),\
        _("The number(s) which form the solution are hidden somewhere in the fields."),\
```

```
        _("Try to find these numbers, but be aware of wrong numbers and other dangers along the way."),\
        _("Controls:"),\
        _("Arrow key left - Walk to the left."),\
        _("Arrow key up - Jump over things or jump to grep a rope."),\
        _("Arrow key down - Let go of a rope."),\
        " ",
        _("Difficulty : 5-7 years"),
        " ",
        _("Number of levels : 3")]
        return text
```