Experiment No. 4

Aim: To Implement Wumpus World.

wumpus_world.py

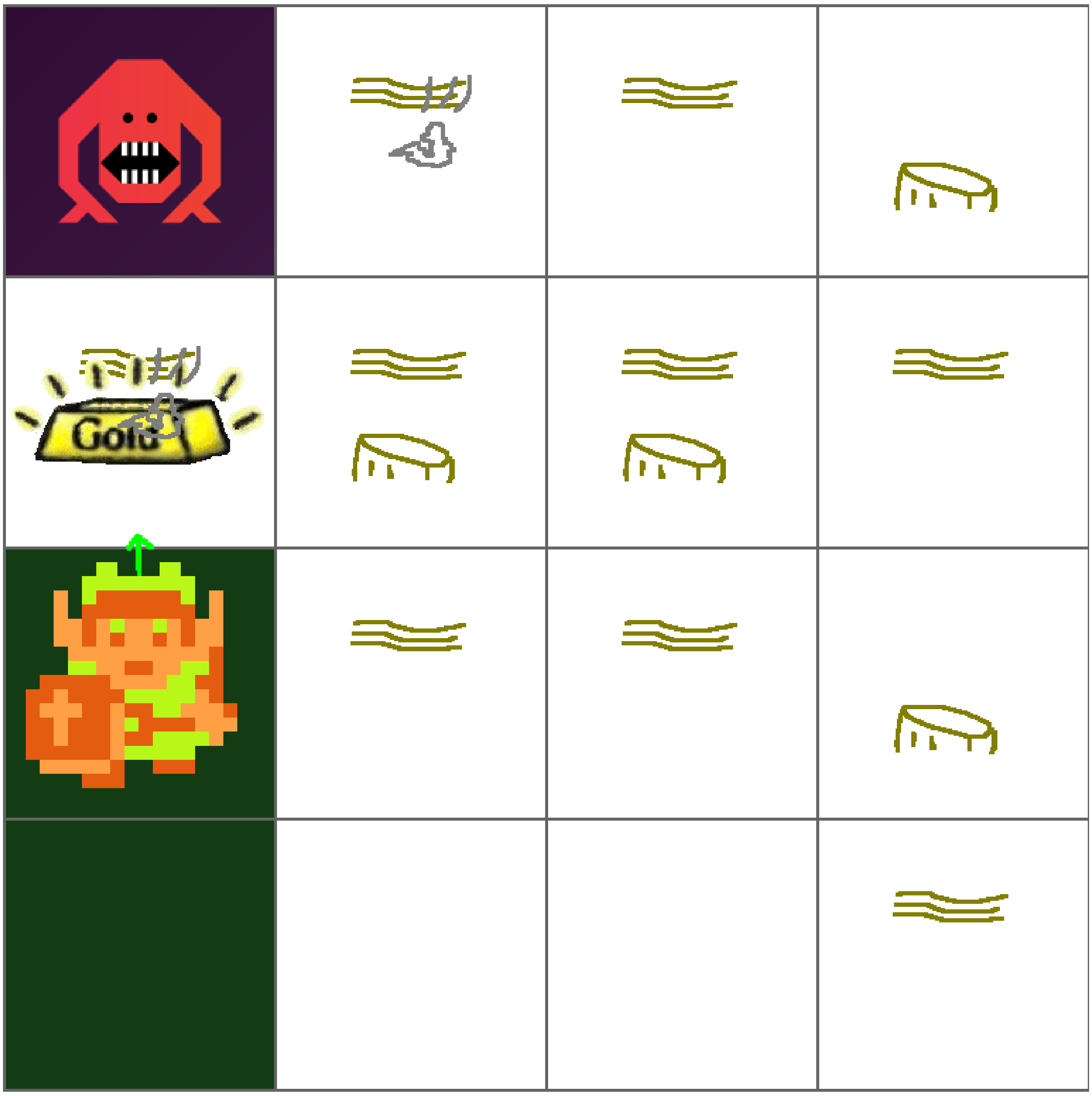
```
from event import EventManager
from view import MainFrame
from controller import *
from app import App
from ai import Agent


def main():
    ev_manager = EventManager()

    main_frame = MainFrame(ev_manager)
    spinner = CPUSpinnerController(ev_manager)
    keybd = KeyboardController(ev_manager)
    ai = Agent(ev_manager)
    app = App(ev_manager)

    spinner.run()


if __name__ == "__main__":
    main()
```

view.py

```
#!/usr/bin/env python

import pdb
import pygame
import event
import ai
from config import *
from util import load_image


class MainFrame:

    """Main Frame -- the whole window"""
    def __init__(self, ev_manager):
        self.ev_manager = ev_manager
        self.ev_manager.register_listener(self)

        pygame.init()
        self.screen = pygame.display.set_mode((769, 820))
        pygame.display.set_caption('wumpus world')

        self.background = pygame.Surface(self.screen.get_size())
        self.background.convert()
        self.background.fill(color['gray'])

        self.screen.blit(self.background, (0, 0))
        pygame.display.flip()

        self.back_sprites = pygame.sprite.RenderUpdates()
        self.front_sprites = pygame.sprite.RenderUpdates()

        self._sectors = {}
        self.player_sector = None
        self.view_all = False

        self._help_display = HelpDisplay()
        self._help_display.rect.center = self.background.get_rect().center
```

```
def _handle_app_start(self):
    dx, dy = (192, 192)
    x, y = (3, 4)
    rect = pygame.Rect(2+192*3, -192+2, 188, 188)
    for count in xrange(16):
        if count % 4 == 0:
            x -= 3
            y -= 1
            rect = rect.move(-3 * dx, dy)
        else:
            x += 1
            rect = rect.move(dx, 0)
        new_sector = Sector(self.back_sprites)
        new_sector.index = (x, y)
        new_sector.rect = rect
        self._sectors[(x, y)] = new_sector

    self._status_display = StatusDisplay(self.back_sprites)
    self._status_display.rect = pygame.Rect(0, 770, 769, 50)

    self._player = Player()

    ev = event.GenerateRequestEvent()
    self.ev_manager.post(ev)

    self._status_display.display(instruction)

def _player_moveto(self, pos):
    self.player_sector = self._sectors[pos]
    self._player.moveto = self.player_sector.rect.center

def _handle_help(self, pos):
    if not self.front_sprites.has(self._help_display):
        self._help_display.add(self.front_sprites)
    else:
        self._help_display.remove(self.front_sprites)

def _handle_ready(self, ev):
    self._status_display.set_ready()

def _handle_busy(self, ev):
    self._status_display.set_busy()

def _handle_player_forward(self, ev):
    self._status_display.display(ev.name)
    if not self.front_sprites.has(self._player):
        self._player.add(self.front_sprites)
        self._player.update_facing(ai.facing_list['right'])
    self._player_moveto(ev.pos)
    self._sectors[ev.pos].visit()
    self.front_sprites.update()
    self._redraw()

def _handle_player_turn(self, ev):
    self._status_display.display(ev.name)
    self._player.update_facing(ev.facing)

def _handle_player_pick(self, ev):
    self._status_display.display(ev.name)

def _handle_player_die(self, ev):
    self._status_display.display(ev.name, color['urgent'])

def _handle_wumpus_die(self, ev):
    self._status_display.display(ev.name, color['urgent'])

def _handle_found_danger(self, ev):
    self._sectors[ev.pos].set_danger()

def _handle_toggle_view(self, ev):
    if self.view_all:
        self.view_all = False
    else:
        self.view_all = True

    for s in self._sectors.values():
        if not s.visited:
            s.toggle_view(self.view_all)

def _handle_world_built(self, ev):
    self._status_display.display(ev.name)
    for key, sector in self._sectors.items():
        item = ev.world[key]
        for x in xrange(5):
            if item[x] == 2:
                thing = ai.map_list[x].lower()
                sector.things.append(thing)

def _handle_reset_world(self, ev):
    for sector in self._sectors.values():
        sector.visited = False
        sector.danger = False
        sector.image.fill(color['background'])
        sector.things = []
    self._player.remove(self.front_sprites)

    ev = event.GenerateRequestEvent()
    self.ev_manager.post(ev)

def _redraw(self):
    # Draw everything
    self.back_sprites.clear(self.screen, self.background)
    self.front_sprites.clear(self.screen, self.background)

    self.back_sprites.update()
    self.front_sprites.update()

    dirty_rects1 = self.back_sprites.draw(self.screen)
    dirty_rects2 = self.front_sprites.draw(self.screen)

    dirty_rects = dirty_rects1 + dirty_rects2
    pygame.display.update(dirty_rects)

def notify(self, ev):
    if isinstance(ev, event.TickEvent):
        self._redraw()
    elif isinstance(ev, event.AppStartEvent):
        self._handle_app_start()
    elif isinstance(ev, event.ResetEvent):
        self._handle_reset_world(ev)
    elif isinstance(ev, event.WorldBuiltEvent):
        self._handle_world_built(ev)
    elif isinstance(ev, event.PlayerForwardEvent):
        self._handle_player_forward(ev)
    elif isinstance(ev, event.PlayerTurnEvent):
        self._handle_player_turn(ev)
    elif isinstance(ev, event.PlayerPickEvent):
        self._handle_player_pick(ev)
    elif isinstance(ev, event.PlayerDieEvent):
        self._handle_player_die(ev)
    elif isinstance(ev, event.WumpusDieEvent):
        self._handle_wumpus_die(ev)
    elif isinstance(ev, event.ReadyEvent):
        self._handle_ready(ev)
    elif isinstance(ev, event.BusyEvent):
        self._handle_busy(ev)
    elif isinstance(ev, event.ToggleViewEvent):
        self._handle_toggle_view(ev)
    elif isinstance(ev, event.HelpEvent):
        self._handle_help(ev)
    elif isinstance(ev, event.FoundDangerEvent):
        self._handle_found_danger(ev)


class HelpDisplay(pygame.sprite.Sprite):

"""Help information"""

def __init__(self):
    pygame.sprite.Sprite.__init__(self)
    self.image = pygame.Surface((500, 580))
    self.image.set_alpha(255 * 0.6)
    self.image.fill(color['gray'])
    self.rect = self.image.get_rect()
    self.text = help

    self._draw_text()

def _draw_text(self):
    try:
        fo = pygame.font.Font(*help_font)
    except IOError:
        fo = pygame.font.Font(None, help_font[1])

    prevpos = None
    for line in self.text.split('\n'):
        textr = fo.render(line, 1, color['help'])
        textrpos = textr.get_rect()
        textrpos.left = self.image.get_rect().left
        if prevpos:
            textrpos.top = prevpos.bottom
        else:
            textrpos.top = self.image.get_rect().top
        prevpos = textrpos
        self.image.blit(textr, textrpos)

def update(self):
    pass


class Sector(pygame.sprite.Sprite):

"""Sector of the map"""

def __init__(self, group=None):
    pygame.sprite.Sprite.__init__(self, group)
    self.image = pygame.Surface((190, 190))
    self.image.fill(color['white'])

    self.index = None
    self.visited = False
    self.danger = False
    self.view = False
    self.things = []

def _draw_things(self):
    if self.danger:
        self.image.fill(color['danger'])
    for t in self.things:
        self.draw_img(t)

def _clear_things(self):
    if self.danger:
        self.image.fill(color['danger'])
    else:
        self.image.fill(color['white'])


def toggle_view(self, view_flag):
    self.view = view_flag
    if view_flag:
        self._draw_things()
    else:
        self._clear_things()

def set_danger(self):
    if not self.danger:
        self.danger = True
        self.image.fill(color['danger'])
        if self.view:
            self._draw_things()

def draw_img(self, s):
    image, rect = load_image('%s.png' % s, -1)
    rect.center = self.image.get_rect().center
    self.image.blit(image, rect)


def visit(self):
    if not self.visited:
        self.visited = True
        self.image.fill(color['light_green'])
        self._draw_things()

def update(self):
    pass


class StatusDisplay(pygame.sprite.Sprite):

"""Game information display area"""

def __init__(self, group=None):
    pygame.sprite.Sprite.__init__(self, group)
    self.image = pygame.Surface((769, 50))
    self.image.fill(color['background'])
    self.text = ''
    self.ready = True
    self.light = False
    self.timer = 0

    self.red_light, self.red_pos = \
                    load_image('red_light.png', -1)
    self.green_light, self.green_pos = \
                     load_image('green_light.png', -1)
    self.red_pos.midleft = \
                       self.image.get_rect().move(10, 0).midleft
    self.green_pos.midleft = \
                        self.image.get_rect().move(10, 0).midleft

def display(self, text, col=color['info']):
    self.image.fill(color['background'])
    self.text = text
    try:
        fo = pygame.font.Font(*status_font)
    except IOError:
        fo = pygame.font.Font(None, status_font[1])
    textr = fo.render(text, 1, col)
    textrpos = textr.get_rect()
    textrpos.center = self.image.get_rect().center
    self.image.blit(textr, textrpos)

def set_busy(self):
    self.ready = False
    self.draw_ready_busy()

def set_ready(self):
    self.ready = True
    self.draw_ready_busy()
def draw_ready_busy(self):
    if not self.ready:
        light = self.red_light
        rect = self.red_pos
        self.light = True
        self.timer = 0
    else:
        light = self.green_light
        rect = self.green_pos
        if self.timer == 0:
            if self.light:
                self.light = False
            else:
                self.light = True
            self.timer = light_flick_ticks
        else:
            self.timer -= 1

    if self.light:
        self.image.blit(light, rect)
    else:
        pygame.draw.rect(self.image,
                       color['background'], rect, 0)

def update(self):
    self.draw_ready_busy()

class Player(pygame.sprite.Sprite):

"""Player in the cave"""

def __init__(self):
    pygame.sprite.Sprite.__init__(self)
    self.image, self.rect = load_image('u1.png', -1)

    self.moveto = None
    self.facing = None

def update_facing(self, facing=None):

    def draw_facing(image, rect):
        if self.facing == 0:
            rect.midtop = self.image.get_rect().midtop
        elif self.facing == 1:
            rect.midright = self.image.get_rect().midright
        elif self.facing == 2:
            rect.midbottom = self.image.get_rect().midbottom
        elif self.facing == 3:
            rect.midleft = self.image.get_rect().midleft
        self.image.blit(image, rect)

    # clear the old facing line
    if self.facing is not None:
        image = pygame.Surface((30, 30))
        image.fill(color['black'])
        rect = image.get_rect()
        draw_facing(image, rect)
    self.facing = facing
    image, rect = load_image('facing_%s.png' % \
                             self.facing, -1)
    draw_facing(image, rect)

def update(self):
    if self.moveto:
        self.rect.center = self.moveto
        self.moveto = None
```

Output:-



● Player moves forward