CARRERA: Computación
NRO. PRÁCTICA: 4
ESTUDIANTE: Doménica Merchán García

PRÁCTICA DE LABORATORIO

ASIGNATURA: Simulación
TÍTULO: Simulación de la propagación del COVID-19 en Ecuador

ACTIVIDADES DESARROLLADAS

In [1]:
```
import pandas as pd
import numpy as np
from random import randrange
import pygame as pg
```

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

El factor R0 del COVID-19 oscila entre 1,5 y 4,08. Con propósito de este trabajo se tomará el valor de 4,08 como el peor caso (Eisenberg, 2020). La probabilidad de muerte por COVID es del 6.3% (Sousa et. al, 2020)

Primero se establecen los parámetros de entrada:

- PM: Probabilidad de muerte por COVID
- R0: Factor de intensidad del COVID
- PI: Probabilidad de infección
- PV: Probabilidad de vacunación
- SPEED: Tiempo del día en milisegundos

In [2]:
```
PM = 6.3
R0 = 4,08
PI = R0*10
PV = 5
SPEED = 50
ROWS = 50
COLS = 50
```

Se crean métodos para obtener los vecinos inmediatos a una posición dada, simular el proceso de vacunación, y simular la muerte por COVID.

In [3]:
```
def get_vecinos(x, y):
    incx = randrange(3)
    incy = randrange(3)
    incx = (incx * 1) - 1
    incy = (incy * 1) - 1
    x2 = x + incx
    y2 = y + incy
    #Validar limites
    if x2 < 0:
        x2 = 0
    if x2 >= COLS:
        x2 = COLS - 1
    if y2 < 0:
        y2 = 0
    if y2 >= ROWS:
        y2 = ROWS - 1
    return [x2, y2]
```

In [4]:
```
def vacunar():
    for x in range(COLS):
        for y in range(ROWS):
            if randrange(99) < PV:
                states[x][y] = 1
```

In [5]:
```
def contar_muertes():
    count = 0
    for x in range(COLS):
        for y in range(ROWS):
            if states[x][y] == -1:
                count += 1
    return count
```

Se establecen los valores iniciales para la simulación, como la posición inicial del primer infectado. Luego se establecen variables para contabilizar la cantidad de muertes y el número de iteraciones para la simulación.

In [6]:
```
states = [[0] * COLS for i1 in range(ROWS)]
states_temp = states.copy()
states[randrange(50)][randrange(50)] = 10
it = 0
total_muerte = 0
vacunar()
```

Se definen los colores a usarse para los diferentes casos:

- WHITE: color del fondo
- BLUE: No infectado
- GREEN: Recuperado
- BLACK: Muerto

In [7]:
```
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
GREEN = (0, 247, 0)
BLACK = (0, 0, 0)
```

Se inicia pygame y se genera una ventana donde se mostrará el avance de la simulación de acuerdo a las iteraciones con los parametros definidos anteriormente.

In [8]:
```
pg.init() #Inicializo el motor de juegos pygame
pg.font.init() #Inicializo el tipo de letra
display=pg.display.set_mode((800,750),0,32) #Tamanio de la ventana
pg.display.set_caption("Simulación de la propagación del COVID-19 en Ecuador") # Titulo
font=pg.font.SysFont('Times New Roman', 40) # Tipo de letra
display.fill(WHITE)
```

Out[8]:
```
<rect(0, 0, 800, 750)>
```

In [ ]:
```
while True:
    pg.time.delay(SPEED) # Sleep o pausa
    it = it + 1
    if it <= 10000 and it >= 2:
        states_temp = states.copy() #Copia de la matriz
        #Recorrera la matriz
        for x in range(COLS):
            for y in range(ROWS):
                state = states[x][y]
                if state == -1:
                    pass
                if state >= 10: # Numero de dias de contagio
                    states_temp[x][y] = state + 1
                if state >= 20:
                    if randrange(99) < PM: # Genero un randomico para verificar si fa
                        states_temp[x][y] = -1 # Muere
                    else:
                        states_temp[x][y] = 1 # Cura o recupera
                if state >= 10 and state <= 20: # Rango de infectado
                    if randrange(99) < PI: # Infecto a las personas cercanas entre  1
                        neighbour = get_vecinos(x, y) #Obtenemos los vecinos a contag
                        x2 = neighbour[0]
                        y2 = neighbour[1]
                        neigh_state = states[x2][y2]
                        if neigh_state == 0: #Verifico que este sano
                            states_temp[x2][y2] = 10 # Contagia
        states = states_temp.copy()
        total_muerte = contar_muertes() # contar el numero de muertos

    pg.draw.rect(display, WHITE, (300, 30, 260, 50)) # Grafico el fondo
    textsurface = font.render("Total muertes: "+ str(total_muerte), False, (255,160,12
    display.blit(textsurface, (250, 30)) # Graficar el texto de muertes
    #Graficar el estado del paciente matriz
    for x in range(COLS):
        for y in range(ROWS):
            if states[x][y] == 0:
                color = BLUE # No infectado
            if states[x][y] == 1:
                color = GREEN # Recupero
            if states[x][y] >= 10:
                color = (states[x][y] * 12, 50, 50) # Injectado - Rojo
            if states[x][y] == -1:
                color = BLACK # Muerto
            pg.draw.circle(display, color, (100 + x * 12 + 5, 100 + y * 12 + 5), 5)
            pg.draw.rect(display, WHITE, (100 + x * 12 + 3, 100 + y * 12 + 4, 1, 1))
    #Escuachar los eventos del teclado
    for event in pg.event.get():
        if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE: #Presiona y Escape
            pg.image.save(display, "img.png")
            pg.quit() #Termino simulacion
        if event.type == pg.KEYDOWN and event.key == pg.K_SPACE: #Presiona y espacio
            #Reiniciamos valores
            states = [[0] * nb_cols for i1 in range(nb_rows)]
            states_temp = states.copy()
            states[5][5] = 10
            it = 0
            total_muerte = 0
            vacunar() #Llamar a la funcion vacunar
    pg.display.update() # Mandar actualizar la ventana
```

Para un factor R0=4.08 y una probabilidad de vacunación del 5% se tiene que habrá un total de 170 muertes por COVID. Para realizar otra prueba se establecen los parámetros de R0=1.5 (como el mejor de los casos) y la probabilidad de vacunación al 15%.

In [9]:
```
R0 = 1.5
PV = 25
```

In [10]:
```
while True:
    pg.time.delay(SPEED) # Sleep o pausa
    it = it + 1
    if it <= 10000 and it >= 2:
        states_temp = states.copy() #Copia de la matriz
        #Recorrera la matriz
        for x in range(COLS):
            for y in range(ROWS):
                state = states[x][y]
                if state == -1:
                    pass
                if state >= 10: # Numero de dias de contagio
                    states_temp[x][y] = state + 1
                if state >= 20:
                    if randrange(99) < PM: # Genero un randomico para verificar si fa
                        states_temp[x][y] = -1 # Muere
                    else:
                        states_temp[x][y] = 1 # Cura o recupera
                if state >= 10 and state <= 20: # Rango de infectado
                    if randrange(99) < PI: # Infecto a las personas cercanas entre  1
                        neighbour = get_vecinos(x, y) #Obtenemos los vecinos a contag
                        x2 = neighbour[0]
                        y2 = neighbour[1]
                        neigh_state = states[x2][y2]
                        if neigh_state == 0: #Verifico que este sano
                            states_temp[x2][y2] = 10 # Contagia
        states = states_temp.copy()
        total_muerte = contar_muertes() # contar el numero de muertos

    pg.draw.rect(display, WHITE, (300, 30, 260, 50)) # Grafico el fondo
    textsurface = font.render("Total muertes: "+ str(total_muerte), False, (255,160,12
    display.blit(textsurface, (250, 30)) # Graficar el texto de muertes
    #Graficar el estado del paciente matriz
    for x in range(COLS):
        for y in range(ROWS):
            if states[x][y] == 0:
                color = BLUE # No infectado
            if states[x][y] == 1:
                color = GREEN # Recupero
            if states[x][y] >= 10:
                color = (states[x][y] * 12, 50, 50) # Injectado - Rojo
            if states[x][y] == -1:
                color = BLACK # Muerto
            pg.draw.circle(display, color, (100 + x * 12 + 5, 100 + y * 12 + 5), 5)
            pg.draw.rect(display, WHITE, (100 + x * 12 + 3, 100 + y * 12 + 4, 1, 1))
    #Escuachar los eventos del teclado
    for event in pg.event.get():
        if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE: #Presiona y Escape
            pg.image.save(display, "img2.png")
            pg.quit() #Termino simulacion
        if event.type == pg.KEYDOWN and event.key == pg.K_SPACE: #Presiona y espacio
            #Reiniciamos valores
            states = [[0] * nb_cols for i1 in range(nb_rows)]
            states_temp = states.copy()
            states[5][5] = 10
            it = 0
            total_muerte = 0
            vacunar() #Llamar a la funcion vacunar
    pg.display.update() # Mandar actualizar la ventana
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-10-2bf35f303647> in <module>
     47             total_muerte = 0
     48             vacunar() #Llamar a la funcion vacunar
---> 49     pg.display.update() # Mandar actualizar la ventana

error: video system not initialized
```

Como se muestra al final de la simulación, el total de muertes por COVID disminuye con los nuevos parámetros ingresados. Esto va acorde con lo esperado, puesto a que se ha aumentado la probabilidad de vacunación para evitar la muerte por COVID, y se ha disminuido el factor de intensidad del brote.

Bibliografía

Eisenberg J. (2020) Coronavirus: qué es el factor R0 con el que se mide la intensidad de un brote como el coronavirus y su potencial pandémico. BBC. https://www.bbc.com/mundo/noticias-51469198

Sousa, G., Garces, T., Cestari, V., Florêncio, R., Moreira, T., & Pereira, M. (2020). Mortality and survival of COVID-19. Epidemiology and Infection, 148, E123. doi:10.1017/S0950268820001405