UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

MAESTRÍA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PROGRAMACIÓN AVANZADA

PROGRAMA: CONCURRENCIA E HILOS

ALUMNA:

RESÉNDIZ BENHUMEA GEORGINA MONTSERRAT

DR. CARLOS GERSHENSON GARCÍA

DR. DANTE PÉREZ MÉNDEZ

FECHA DE ENTREGA: 04-09-2018

SEMESTRE 2019-1

PROGRAMA: CONCURRENCIA E HILOS

RESÉNDIZ BENHUMEA GEORGINA MONTSERRAT

INTRODUCCIÓN

En computación, un proceso es una instancia de un programa de computadora que se está ejecutando. El proceso está compuesto por: un programa ejecutable, datos asociados al programa (variables, espacios de trabajo, etc.) y el contexto de ejecución del programa (Estado del proceso).

Un hilo (thread) es una entidad dentro de un proceso que puede programarse para su ejecución. Es la unidad de procesamiento más pequeña que se puede planificar en un Sistema Operativo (OS). Es decir, un hilo es una secuencia de instrucciones dentro de un programa que se puede ejecutar independientemente de otro código.

Los hilos de un proceso pueden compartir la memoria de variables globales. Si una variable global es modificada en un hilo, este cambio será válido para todos los hilos. Así mismo, un hilo puede tener variables locales.

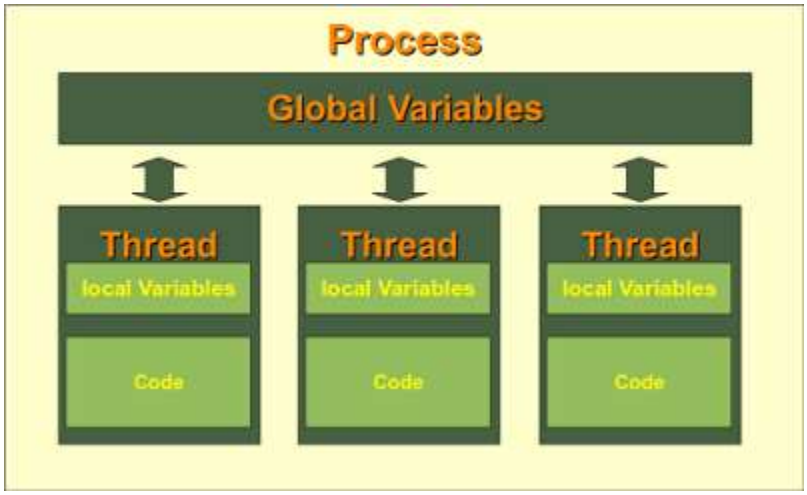

Figura 1. Proceso e Hilos

Un hilo tiene un comienzo, una secuencia de ejecución (code) y una conclusión. Así también, tiene un puntero que lleva un seguimiento de dónde se está ejecutando actualmente dentro de su contexto.

Concurrencia es la propiedad de los sistemas que permiten que múltiples procesos sean ejecutados al mismo tiempo (paralelismo) y que puedan interactuar entre sí.

La concurrencia es simulada a través de los hilos, donde el procesador se ocupa de cada uno de ellos de forma alternada en pequeños intervalos de tiempo, de esta manera se simula que se están ejecutando simultáneamente.

OBJETIVO

- Implementar The Game of Life mediante el concepto de concurrencia a través de hilos, utilizando el lenguaje de programación Python 3.7 y Pygame.

DESCRIPCIÓN DEL PROGRAMA

Se desarrolló una ventana con Python 3.7 y Pygame, donde se muestra la implementación de The Game of Life. La matriz de células se modifica de acuerdo a los siguientes eventos:

- **Presionar Tecla '0' - Clear Matrix**
  Genera una matriz de células muertas (valor de celda 0).

- **Presionar Tecla '1' - Random Matrix**
  Genera una matriz aleatoria (valores de celda 0 ó 1).

- **Presionar Tecla '2' - Template Still Lifes**
  Genera una matriz con la plantilla de Still Lifes, para observar el correcto funcionamiento del mismo.

- **Click mouse - Draw Living Cell**
  Dibuja una célula viva (valor de celda 1) al dar click sobre ella con el mouse.

A continuación, se muestra la ventana de la implementación de "The Game of Life":

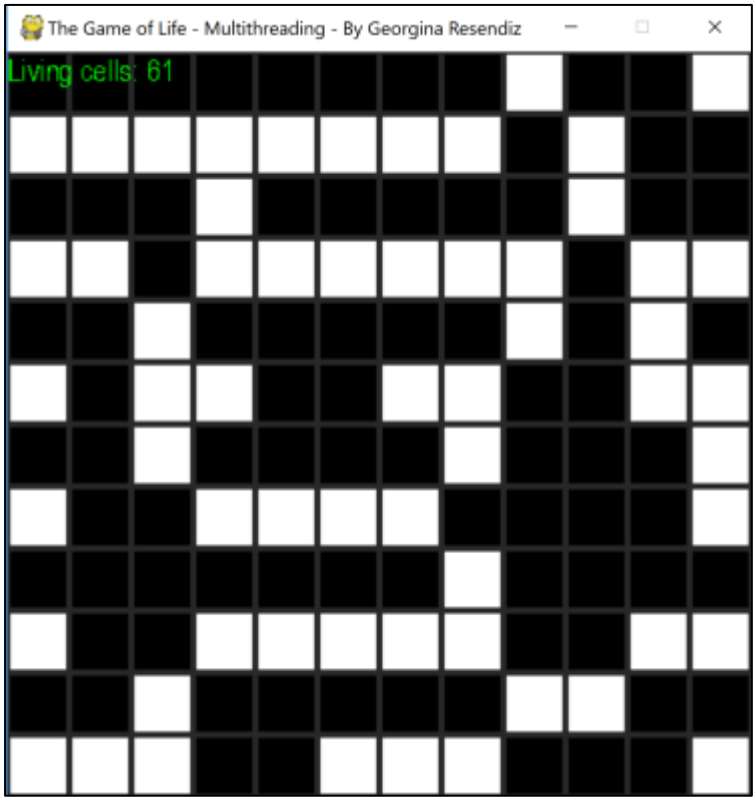

Figura 2. Ventana de implementación de “The Game of Life”
Python 3.7 y Pygame

FUNCIONAMIENTO DEL PROGRAMA

El programa funciona de acuerdo con lo siguiente:

1. Se comienza con una matriz aleatoria y posteriormente se realiza la creación de objetos de la clase Cell, los cuales heredan los atributos y métodos de la clase “Thread” del módulo “threading”, es decir, cada una de las células de la matriz son un hilo (thread).

   Una vez generados los hilos (células de la matriz), éstos ejecutan su código de forma independiente a los demás hilos. Este código realiza la suma de los 8 vecinos de alrededor y la obtención del nuevo valor de la célula a partir de las reglas siguientes:

- Cualquier célula viva (valor 1) con menos de dos vecinos vivos muere (valor 0), a causa de subpoblación (fallecimiento).

- Cualquier célula viva (valor 1) con más de tres vecinos vivos muere (valor 0), a causa de sobrepoblación (fallecimiento).
- Cualquier célula viva (valor 1) con dos o tres vecinos vivos se mantiene viva (valor 1) para la siguiente generación (supervivencia).
- Cualquier célula muerta (valor 0) con exactamente tres vecinos vivos se convierte en célula viva (valor 1) (nacimiento).

  Cabe señalar que los valores de la matriz son compartidos por todos los hilos (threads), por lo tanto, para generar sincronía entre ellos se requiere utilizar de la clase Lock, el método acquire() que permite a un hilo (thread) realizar modificaciones a los valores de la matriz, mientras tanto los demás hilos (threads) tendrán que esperar a que éste lo libere mediante el método release() y entonces se encuentre disponible para otro hilo (thread).

  También se muestra el número de células vivas (Living cells) en la esquina superior izquierda de la ventana.

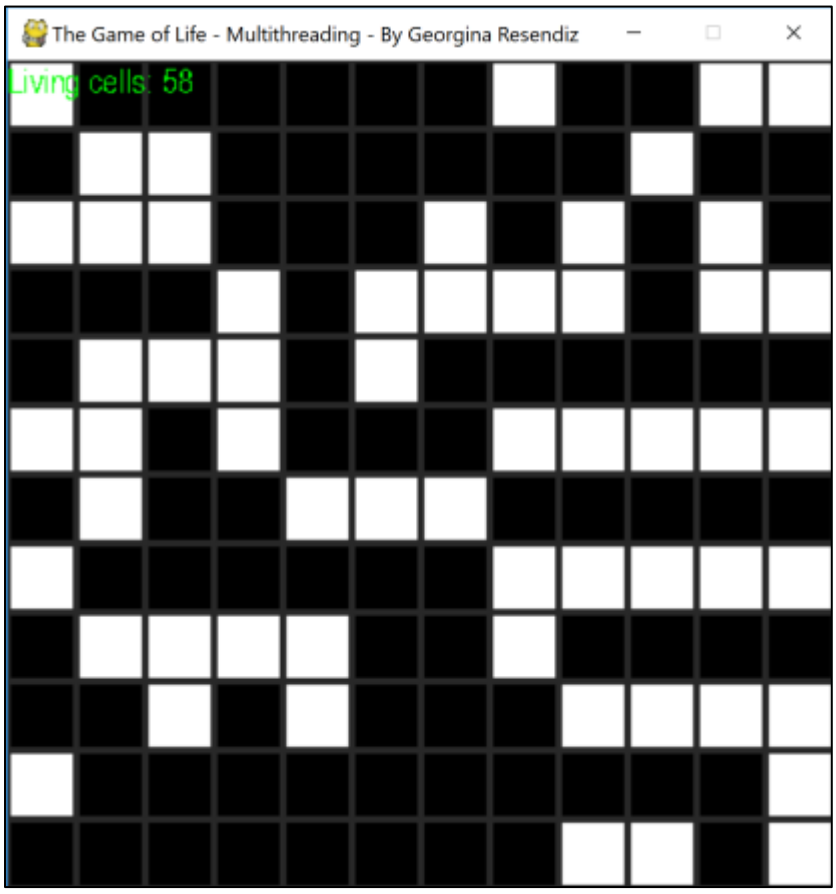

Figura 3. Matriz inicial aleatoria y muestra del número de células vivas (Living cells)

2. Se pueden generar algunos de los siguientes eventos por medio del teclado o mouse, según corresponda:

    - **Presionar tecla ‘0’:** Genera una matriz de células muertas (Clear Matrix).

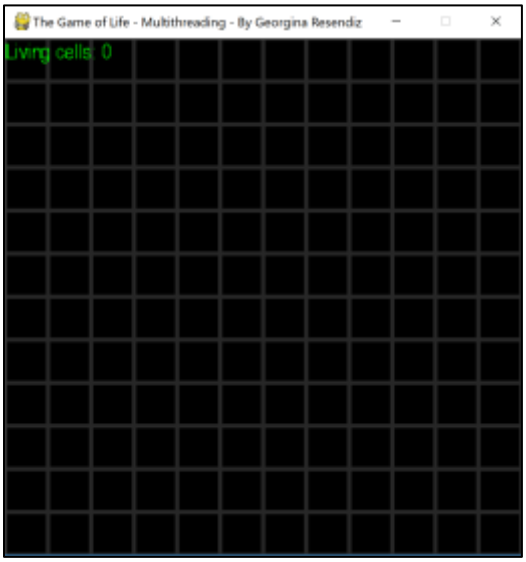

Figura 4. Clear Matrix - tecla ‘0’

    - **Presionar tecla ‘1’:** Genera una matriz aleatoria (Random Matrix).

Figura 5. Random Matrix - tecla ‘1’

- **Presionar tecla ‘2’:** Genera la plantilla de Still Lifes (Template - Still Lifes).

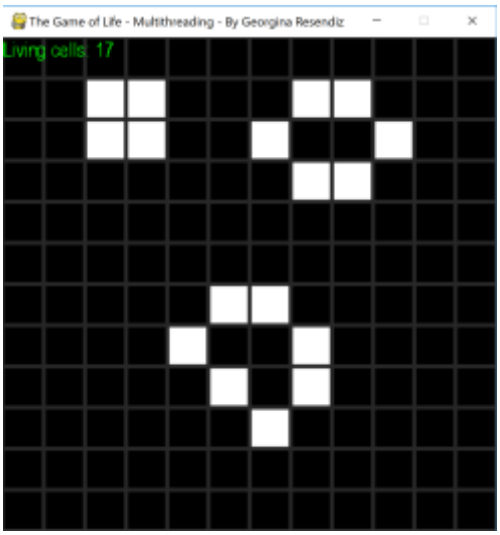

Figura 6. Template - Still Lifes - tecla ‘2’

- **Click mouse - Draw Living Cell:** Dibuja una célula viva en la posición (x,y) de la matriz al dar click sobre ella.

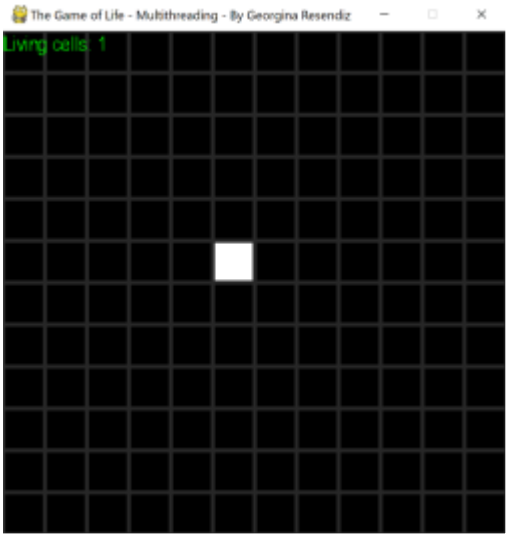

Figura 7. Dibujar célula viva - Click mouse

Cuando se dibuja una célula viva en una matriz de células muertas, ésta morirá inmediatamente porque no tiene vecinos (subpoblación).

3. Finalmente, cuando se realiza el evento de cerrar el programa, se envía ‘False’ a la variable global CELLS_RUNNING para que cada uno de los hilos terminen de ejecutarse.

Figura 8. Cerrar programa y terminar ejecución de hilos