

Software de simulación y estudio de una Mini-Evolución Artificial

Resumen. En el presente trabajo se buscó simular de manera simple y representativa la evolución de una población de microorganismos, que viven en un mundo plano, en el cual se mueven, alimentan y reproducen. El diseño y desarrollo del mismo se realizó con una metodología orientada a objetos. Se utilizaron tarjetas de clases, responsabilidad y colaboración (CRC) y diagramas de jerarquía y colaboración. La implementación se realizó en lenguaje C++11. El resultado final incluye una interfaz de usuario que permite modificar ciertos parámetros iniciales de la población de microorganismos, y ver, de manera gráfica, los movimientos de los mismos y su interacción con otros objetos del mundo donde viven. El software permite almacenar datos de la simulación y facilita el análisis de la evolución de los organismos, mediante histogramas y curvas gaussianas, que reflejan los distintos niveles de inteligencia alcanzadas en el proceso evolutivo.

Palabras Clave: informática · biología · evolución · simulador · programación orientada a objetos.

1 Introducción

Este trabajo surge del proyecto integrador grupal y obligatorio propuesto para la regularización de una materia de programación. El objetivo es generar un programa para simular un mundo artificial bidimensional en el que se desarrolla una población de microorganismos que se mueven en función de la información que contiene su genoma, se reproducen, y comen el alimento depositado por sembradores. Se pretende así generar un software, que mediante la modificación de ciertos parámetros poblacionales, permita analizar la evolución de estos entes artificiales.

Para llevar adelante el diseño e implementación de este proyecto, desde la cátedra, se nos guió en la metodología básica de diseño y desarrollo orientados a objetos, poniendo en práctica los conceptos asociados a las fases del desarrollo colaborativo de un software. Además se implementaron técnicas como programación genérica, aplicando las herramientas presentes en la librería estándar de C++, graficación, con el manejo de la librería OpenGL, y métodos numéricos, entre otras.

A su vez, nuestro equipo propuso cambios a la idea base del programa. En primer lugar, se incluyeron colonias de microorganismos, las cuales presentan distintas características y compiten entre sí por el alimento que hay en el territorio. En segundo lugar, para visualizar y analizar gráficamente el aumento de la inteligencia, se generaron histogramas y curvas gaussianas. Finalmente, se incorporó una interfaz de usuario, la cual permite una interacción fluida con el programa y la posibilidad de modificación de ciertos parámetros generales, útiles para un análisis más profundo de los diferentes caminos evolutivos.

2 Solución

Planteado el problema y sus requisitos funcionales, se comenzó con el diseño del programa. Se buscaron posibles clases que lleven a cabo las funciones propuestas, plasmando las ideas obtenidas en las tarjetas CRC. Tras discutir y analizar posibles escenarios, y habiendo revisado las clases, se comenzó con la implementación.

Se plantearon los encabezados de cada clase, a modo de visualizar las interacciones entre ellas, y los métodos y atributos que permitirían la realización de sus tareas. Durante el desarrollo del código también se hicieron algunas revisiones del diseño original, incorporando ciertas clases y reformulando otras. Con la generación de pruebas para validar el funcionamiento de cada módulo del programa. El progreso fue presentado y discutido semanalmente con los docentes.

La codificación fue realizada en el Entorno de Desarrollo Qt Creator, y se incorporó, además, la utilización de GitLab, instalado en un servidor propio de la Facultad. Esto con el fin de realizar un trabajo colaborativo, bajo un sistema de control de versiones, donde se registren y documenten los cambios.

La primera impresión de las clases que podrían componer el programa, fue:

- Microorganismo (MO): modela a un microorganismo y su comportamiento.
- Sembrador: modela a un sembrador y su comportamiento.
- Mundo: genera un espacio de interacción y modela las acciones a realizar.
- Comida: almacena datos del alimento.
- Cromosoma: almacena los datos genéticos del MO.
- Tiempo o época: modela el paso del tiempo.
- Archivador: genera archivos para comparar distintos escenarios.
- Gráficos: genera una ventana con la representación del mundo y sus componentes.

De ellas se descartó la clase Tiempo, ya que consideramos que su tarea podía ser ejecutada por el mundo. Además, a partir de la clase Archivador, se derivaron dos clases, Archivador_t, que se encarga de la lectura y escritura de archivos de tipo .txt, y Archivador_b, que se encarga de la lectura y escritura de archivos binarios.

A continuación se confeccionaron las tarjetas CRC, las cuales permitieron visualizar las responsabilidades que tendría cada clase, para luego definir los métodos y atributos correspondientes a cada una.

Adicionalmente, para un control preliminar más accesible de ciertos parámetros, como las cantidades de microorganismos y sembradores, se introdujo un header de constantes con información utilizable por las clases del programa para llevar a cabo sus responsabilidades.

Las clases surgidas tras éste procedimiento se detallan a continuación.


2.1 MOs

Se crean con la instanciación de cada colonia, con cierta energía inicial y desarrollan sus actividades siempre y cuando dicha energía sea superior a 0, es decir, mientras estén “vivos”. Primero se ejecuta la ingesta y luego el movimiento. La energía ganada

es igual al alimento ingerido, el cual es tomado de la casilla en la cual están situados. Su energía tiene un nivel máximo. El desplazamiento por el mundo es de un paso, moviéndose una casilla a la vez, y en función del contenido de su cromosoma y la disposición de comida a su alrededor. La reproducción se da luego de una cantidad preestablecida de épocas, cruzando los cromosomas de dos microorganismos vivos, con una pequeña posibilidad de mutación en un gen. La cantidad de microorganismos por colonia es siempre constante.

2.2 Cromosoma

Cuenta con 8 genes, uno por cada posición que rodea al microorganismo, los cuales le indican qué dirección tomar en función de la casilla en la que se detectó comida. Define la inteligencia en función de cuantas coincidencias hay entre la casilla que se detectó y la dirección que se tomó. Es responsable de llevar a cabo, durante la reproducción, la cruce de información genética, incluyendo también, con cierta probabilidad, la mutación de un gen del cromosoma generado.

7	0	1
6		2
5	4	3

Cromosoma Ideal:

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Fig. 1. Casillas que rodean a un microorganismo.

2.3 Sembradores

Se los considera inmortales y no consumen energía. Depositán alimento cada vez que visitan una casilla del territorio. Su desplazamiento es de a un paso a la vez, manteniendo una dirección, con cierta probabilidad de cambio..

2.4 Mundo

Es un territorio finito, con los límites prefijados. Les indica a los microorganismos y sembradores qué hacer y cuándo hacerlo. Es decir, le ordena a los sembradores moverse y depositar comida y a los microorganismos moverse, comer, crecer y reproducirse. Produce nuevas generaciones de ambos tipos de seres, sin superar una cantidad límite preestablecida. Conoce la distribución de comida sobre las casillas de su superficie. Está a cargo del registro del paso del tiempo, debiendo conocer la época actual dentro de la simulación y causando su avance.

4

2.5 Comida

Administra el depósito y extracción de alimento de cada casilla del territorio. Hace respetar las cantidades límite de alimento en cada casilla.

2.6 Archivador

Clase abstracta, cuenta con métodos virtuales puros que emplean sus derivadas para llevar a cabo sus tareas de acceso a memoria secundaria.

2.7 Archivador_b y Archivador_t

Ambas clases sobrescriben los métodos heredados de la clase Archivarador, cada una permitiendo la lectura y escritura de archivos en su formato específico. La información almacenada puede referir a la genética de los microorganismos, la cantidad de comida y distribución de sembradores en el mundo, o a datos de las poblaciones de microorganismos. La información es agregada al final de cada archivo, a modo de preservar escrituras previas. El Archivador_b manipula archivos binarios, y el Archivador_t de texto. Éste último permite la inclusión de la fecha de realización del registro.

2.8 Gráficos

Representa gráficamente los actantes de la simulación, mostrando los sembradores como cuadriláteros color violeta, los microorganismos como romboides color verde azulado con una “cabeza” que indica la dirección en la cual se dirige el individuo y la colonia a la cual pertenece en base a su color.

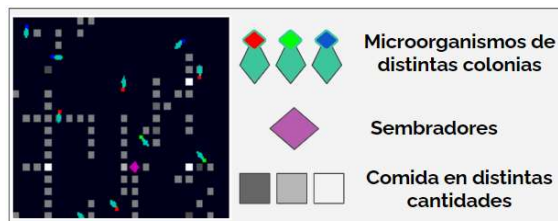


Fig. 2. Elementos del Mundo representados en el Gráfico.

Una vez codificada toda la parte inicial, y habiendo verificado su funcionamiento, se incorporaron ciertas mejoras, que suplementan requisitos funcionales y no funcionales:

1. En la sección de gráficos, se añadió un histograma que refleja la cantidad de microorganismos con una cierta inteligencia. Junto con una curva gaussiana que permite analizar de mejor manera la distribución de inteligencias.
2. Se planteó que los microorganismos pertenecen a colonias, las cuales presentan distintos comportamientos y luchan entre sí, transfiriendo la mitad

de la energía del perdedor al ganador. Éstas evolucionan de forma distinta, por lo que se incorporaron histogramas adicionales en función de la cantidad de colonias. Además se incluyeron nuevos sembradores por cada colonia, ya que al haber más microorganismos es lógico que se necesite más alimento.

3. En el graficador se incluyeron botones para pausar, reproducir, almacenar datos y finalizar la simulación. En una etapa posterior, para mayor simplicidad, dichos botones fueron reemplazados por herramientas propias de los formularios del entorno de desarrollo.
4. El genoma fue modificado para que contenga 2 genes más. El primero fija la cantidad de comida que puede ingerir un microorganismo, de 0 a 10 unidades, contemplando la posibilidad de que un organismo nazca con una mutación que le impida comer. El segundo fija el gasto de energía por movimiento, de 1 a 10 unidades, considerando que todo ser vivo consume energía para moverse.
5. En base al cambio anterior, y ante la complejidad que representa plasmar estos datos en la inteligencia, se incorporaron dos columnas a la derecha del histograma. Las mismas representan la cantidad de microorganismos, por colonia, que (1) consumen más energía de la que gastan y (2) gastan más energía moviéndose que alimentándose.
6. Las colonias presentan comportamientos dispares, siendo los mismos fijables en las configuraciones. En primer lugar está la colonia que coopera, en la cual, si dos o más de sus MOs se encuentran en una misma casilla, aumentan su energía en 2 unidades. En segundo lugar está la colonia que pelea, la cual, al cruzarse con cualquier MO, sea compañero o no, lo ataca. La tercera es la colonia default, la cual solo ataca a los MOs de otras colonias y no genera energía al compartir casilla.

3 Resultados

Para la completación del proyecto, un total de 12 clases fueron generadas. Dos de ellas se basan en las clases de interfaz del entorno Qt Creator: “Interfaz” y “Opciones”.

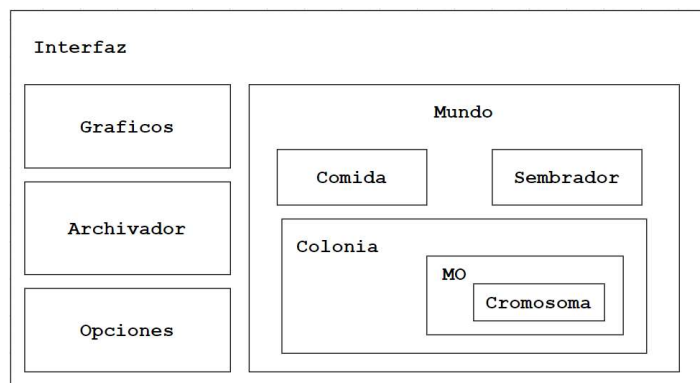


Fig. 3. Diagrama de Contención

6

Originalmente, la interacción con el usuario se realizaba mediante una interfaz creada a partir de los gráficos generados con las herramientas de OpenGL, en el mismo espacio que la representación de la simulación.

Tras aprender a aplicar las funciones de interfaz de Qt Creator, se descartaron parte de los gráficos previos, a modo de ceder responsabilidades de *Gráficos* a *Interfaz*.

Para incluir los nuevos factores evolutivos del cromosoma, como ya se mencionó, se sumó una nueva sección al histograma.

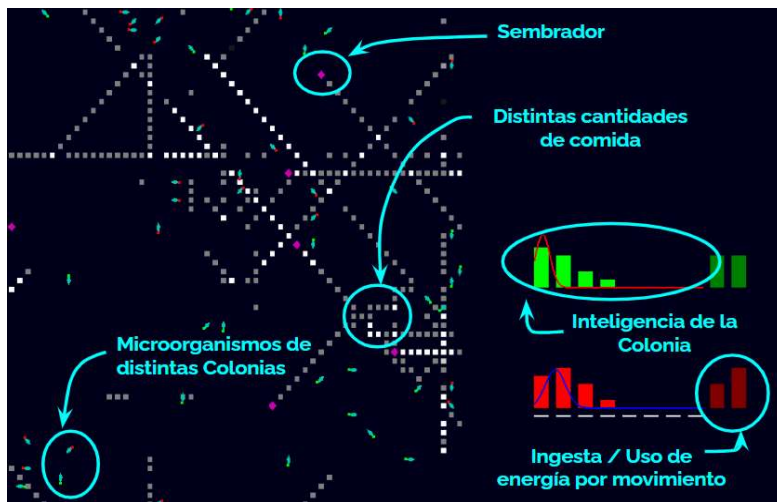


Fig. 4. Gráficos de la simulación, mostrando dos colonias distintas marcadas por los colores rojo y verde.

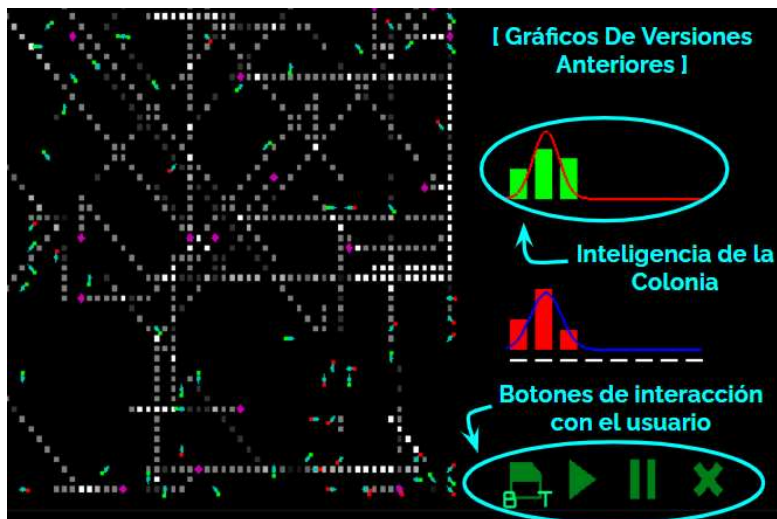


Fig. 5. Gráficos de versiones anteriores; los botones a la derecha abajo fueron reemplazados por los botones propios del entorno de desarrollo Qt.

Además, con la forma facilitada de interacción con el programa, se habilitó la modificación de ciertos parámetros anteriormente constantes.

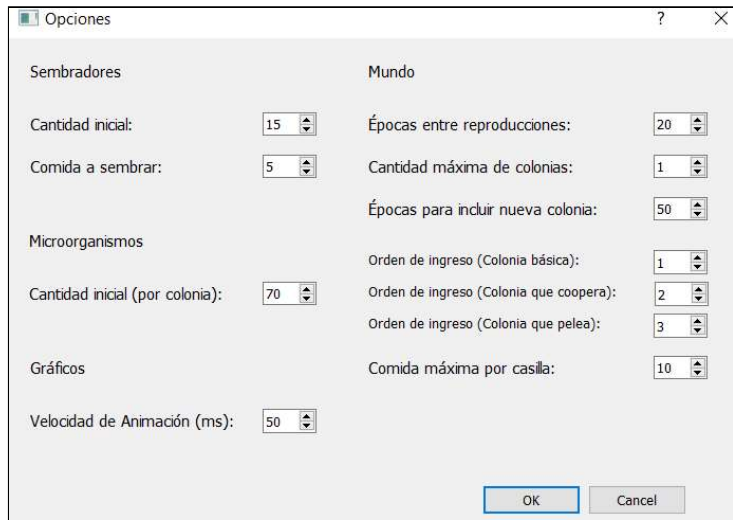


Fig. 6. Ventana de Opciones, permite la modificación de ciertos parámetros que rigen la simulación.

Las consecuencias de la variación en los parámetros pueden ser observadas durante la ejecución del programa, y sus resultados pueden ser capturados en archivos para su posterior acceso, permitiendo la comparación de los caminos evolutivos que toma cada colonia.

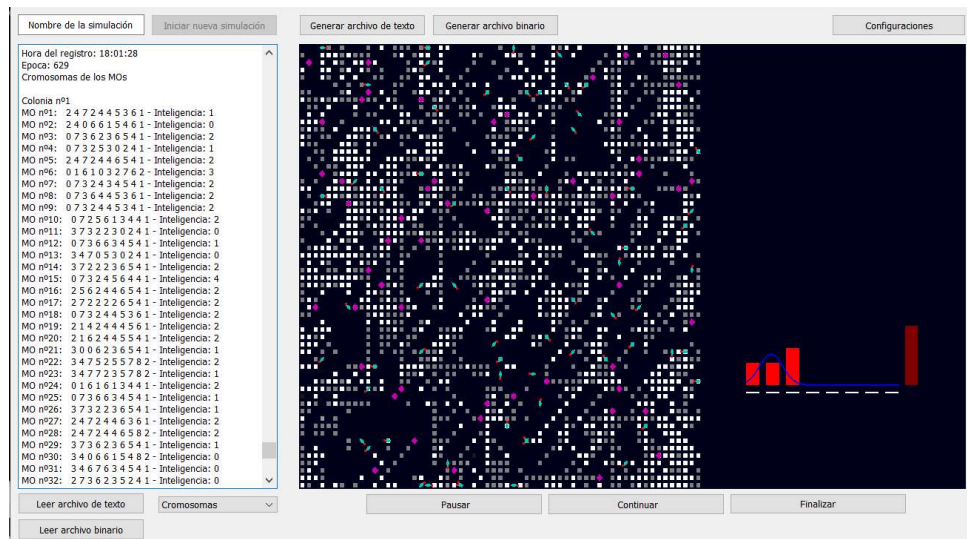


Fig. 7. Prueba 1: Ante una configuración que genera comida en exceso, los microorganismos presentaron una evolución casi sin variaciones.

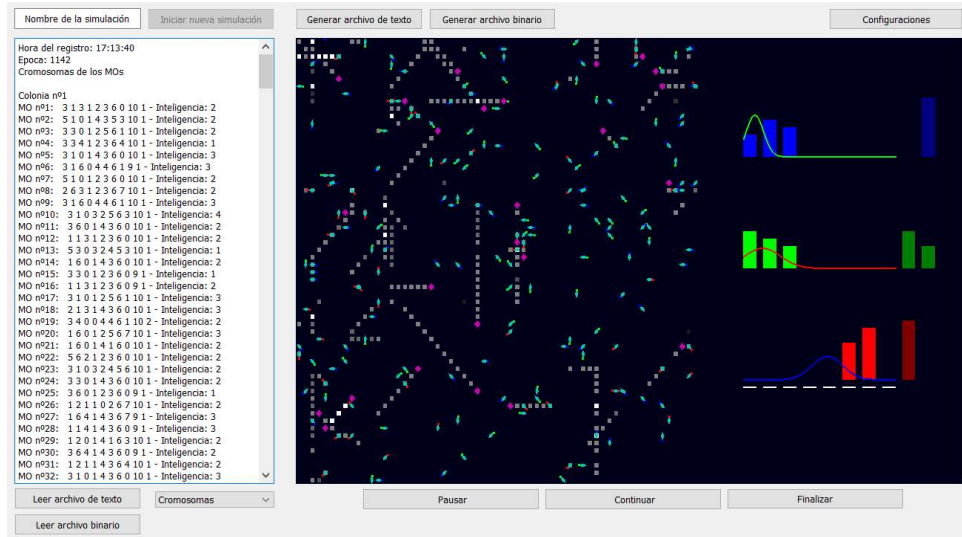


Fig. 8. Prueba 2: Ante la inclusión de mayor número de colonias, la disponibilidad de alimento baja. A su vez, se pueden observar los distintos niveles de evolución según el tipo de colonia. Azul = pelea, verde = colabora, rojo = default.

4 Conclusiones

En este trabajo se presentó resumidamente el proceso seguido para la realización de este proyecto. Al permitir la aplicación conjunta de los conceptos teóricos se logró un mejor dominio de los mismos, en contraste a que si únicamente se hubiesen aplicado prácticas separadas para la revisión de cada concepto. Además, el estrecho contacto con los docentes durante su realización permitió obtener una mirada más crítica sobre el código en cada etapa de su desarrollo. Accediendo de este modo a una programación más experimentada y profesional. Por otro lado, la coordinación en equipo para la desarrollo del proyecto, junto con la herramienta GitLab, permitieron desarrollar una dinámica muy adecuada para el trabajo en forma colaborativa, que resulta muy valiosa en una situación de aislamiento preventivo como la que estamos pasando a lo largo del primer semestre de 2020.

Más ampliamente, se comprendió la importancia de las herramientas informáticas. Se observó que las habilidades adquiridas hasta el momento, a partir de los contenidos vistos en la carrera, pueden ser aplicadas para el desarrollo de programas con utilidad práctica para diversas disciplinas. A partir de los mismos principios empleados para la realización del presente proyecto, es posible generar simulaciones de mayor complejidad que, con la suficiente cantidad de variables y parámetros, permitan modelar un sistema real. Ya sea para observar los caminos evolutivos de ciertos organismos con el fin de visualizar el futuro o entender el pasado, o incluso para analizar la interacción entre los elementos dentro un mismo organismo, generando resultados que, en aplicaciones médicas, podrían evitar costos y riesgos.

En lo referente a la temática abordada, observamos que los objetos, como el genoma, al inicializarse aleatoriamente, fijan un estado inicial único en las colonias, de modo que, para un número pequeño de individuos, cada nueva simulación puede llevar a un nuevo resultado.

En la práctica, el cambio del comportamiento de los individuos en una colonia, o la forma de suministro del alimento, definen el algoritmo evolutivo en el cual sobrevivirán los individuos que mejor se adapten. Por otro lado, el código genético de los microorganismos puede complejizarse tanto como se desee. De esta manera se pueden contemplar casos tales como la supervivencia en un ambiente tóxico, la posibilidad de que desarrollen un mecanismo similar a la “fotosíntesis” o el poder volar o saltar, aumentando la cantidad de casillas recorridas por unidad de energía. Finalmente destacamos, que las posibilidades de representación de múltiples situaciones o condiciones son muy amplias y aplicables a diferentes tipos de problemas, siempre y cuando los aspectos que se quieran optimizar, puedan ser representados en la información genética. Incluso en otras áreas esta técnica podría aportar al conocimiento, por ejemplo, para intentar comprender la vida y evolución de determinadas especies. Incluso, partiendo de casos más simples, como las células procariontas, se podría llegar a predecir la adaptación y supervivencia del organismos en ambientes tan extraños como los de otros planetas.

Bibliografía

1. *Sitio de referencias sobre C++*: <http://www.cplusplus.com/>
2. Deitel, H., Deitel, P.: *Cómo Programar en C++*. 4th edn. Ed. Pearson Prentice Hall (2003).
3. Budd, T.: *Programación Orientada a Objetos*, Addison Wesley Iberoamericana.
4. Wirf-Brock R.: *Designing Object Oriented Software*, Prentice Hall. (1990).
5. Börstler J.: *Object-Oriented Analysis and Design Through Scenario Role-Play*, UMEA University.
6. Sheiner, D., Woo, M.: *OpenGL Programming Guide - The Official Guide to Learning OpenGL -*, Silicon Graphics.
7. Hearn, A.: *Gráficos por computadora*. Ed. Prentice Hall (1987).
8. Wright, R.: *Programación en OpenGL- Una guía de referencia completa de OpenGL*. Anaya Multimedia (1997).