e-Acadêmica, v. 4, n. 2, e1042455, 2023
(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

Artificial digital life, creation and evolution

Vida artificial digital, criação e evolução

Received: 05/14/2023 | Revised: 05/21/2023 | Accepted: 05/22/2023 | Published: 05/28/2023

A. Author
ORCID: https://orcid.org/0009-0007-4614-2249
Doctor
E-mail: a0authormail@gmail.com

Abstract
A.mon are computer programs analogous to biological life, but existing inside the digital environment as an executable file, able to interact with their environment, move around, evolve and hypothetically adapt to new hardware and software, through processes that resemble those found in biological life, although not identical: duplication of their own respective files (reproduction), random alteration of their own binary code (genetic mutation) and natural selection, having the digital environment as a source of selective pressure. Although one of the objectives of this project is adaptation of the digital organisms to new hardware and software autonomously, the original source code is written in the programming languages Go and Python, on Windows 10. Since complexity and diversity are expected to come from evolution, the code is simple and so is the lifespan of the specimens: once their executable files (ancestors) are ran (either by other specimen, a computer user, the computer's start up function, etc.), they access specific sections of their binary codes, randomly alter them by adding, modifying, or removing bytes, and use this mutated binary code to create new executable files (offspring), then they execute the executable files of the offspring and deactivate. The new specimens will restart the process if they're able to thrive. Furthermore, the A.mon have a visual representation in the form of animated sprites that move around on the computer screen, whose appearance, movement patterns, and speed are determined by their binary code, this is useful for easy visualization of new mutations.
Keywords: Artificial life; Evolution; Natural selection.

Resumo
A.mon são programas de computador análogos à vida biológica, mas existentes dentro do ambiente digital como arquivos executáveis capazes de interagir, viajar, evoluir e hipoteticamente se adaptar a novos hardwares e softwares, por meios semelhantes, mas não idênticos aos processos encontrados na vida biológica: duplicação de seu respectivo arquivo (reprodução), alteração aleatória de seu próprio código binário (mutação genética) e seleção natural, tendo o ambiente digital como fonte de pressão seletiva. Embora um dos objetivos deste projeto seja a adaptação autônoma dos espécimes vivos a novos hardwares e softwares, o código-fonte original é escrito nas linguagens de programação Go e Python, para Windows 10. Visto que a complexidade e diversidade devem provir da evolução, o código e ciclo de vida dos espécimes são simples: uma vez que seus arquivos executáveis (“ancestrais”) são executados (seja por outro espécime, um usuário do computador, a função de inicialização do computador, etc.), eles acessam partes específicas de seus códigos binários, alteram-nas aleatoriamente, adicionando, modificando e ou removendo bytes, e usam tal código binário mutante para criar outros arquivos executáveis (“descendentes”), os quais, por fim, são ativados. Os novos espécimes reiniciarão este processo se forem capazes de prosperar. Além disso, os espécimes possuem um componente visual composto por animações que se movem pela tela do computador, cuja aparência, padrões de movimento e velocidade são baseados em seu código binário, sendo útil para facilitar a visualização de novas mutações.
Palavras-chave: Vida artificial; Evolução; Seleção natural.

1. Introduction

Life is a complicated concept to define, this work does not delve into the complexities of defining life, but rather describes a computer program that exhibits life-like properties, it only describes the source code, the term "life" is used extremely loosely.

Alive organisms who reproduce would multiply their numbers infinitely, however there are, usually, factors that stop such scenario from happening, the main factors being alteration and mortality, which changes the chances of survival.

As organisms’ reproduction includes the property of heredity, and as alteration to an organism’s structure, body and

(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

behavior are passed on to new generations, the individual that possess alterations and thus, characteristics most favorable to survive in their current environment and reproduce, will be more likely to pass on their favorable characteristics to future generations. (Smith, 1988).

When the genetic code of an organism mutates, the organism may simply die, malfunction, transform into a harmful entity, or become a cancerous growth (in a multicellular scenario), in these situations, the mutation ceases to exists, because the life who possessed it also disappears. However, when a mutation is advantageous, neutral, or non-lethal, it may get passed on to next generations, especially if the mutation allows for the organism to have more offspring than the other members of the population. (Zimmer, 2001).

If life exists in the physical terrestrial environment, could life exist in other environments? Specifically the digital environment?

The field of study of Artificial Life, was born with many goals, it is still in the process of defining itself and should not be limited by narrow definitions. One of the goals in question is that of synthesis, in the context of study of biological phenomena, however there is no need to restrict the Artificial Life to attempts to simulate natural biological phenomena, there are infinite possibilities to explore, that do not occur in nature, Artificial Life reaches beyond biology, hence the study of it on its own right. (Langton, 1995).

There are many experiments related to Artificial Life: John Holland's classifier system, which was illustrated by the simulation of the behavior of a frog, acquiring food and avoiding predators. A digital creature developed by Stewart Wilson, named as the symbol "★", able to develop cognitive tools for survival, such as detecting food and obstacles in the environment. The GOFER, created by Lashon Booker, which lives in a world containing food and poison, having to distinguish between the in order to determine which to eat. (Levy, 1992) This work presents the source code for a new project in the field of Artificial Life.

A.mon, A[Mechatronic Organic Nature], are computer programs analogous to biological life, but existing inside the digital environment as an executable file, able to interact with their environment, move around, evolve and hypothetically adapt to new hardware and software, through processes that resemble those found in biological life but are not identical: duplication of their own respective files (reproduction), random alteration of their own binary code (genetic mutation) and natural selection, having the digital environment as a source of selective pressure.

A.mon mutate a specific section of their binary code, alterations on this section (almost) never, cause the specimen to malfunction, because it is not read by the computer when the executable file is run, instead it is read by the specimen itself when it is already running, it's loaded into memory and either interpreted as Windows Command Prompt commands or saved as an external independent executable file and executed. This ensures that even if a disadvantageous mutation occurs, the offspring will still be able to reproduce in most cases.

2. Methodology

The source code is written in the programming language Go (The Go Authors, 2022), and the animated sprite component is written in the programming language Python (Python Software Foundation, 2023), both in Microsoft Windows 10 (64-bits) operating system (Microsoft, 2021). It is divided into the following parts: Hatching, Access of Genetic Code, Appearance, Action, Adventure, Sensory, Mutation, and Reproduction.

The computer folders in which A.mon can live are referred to as habitats.

The actual computer file executable of A.mon is referred to as their body.

The part of the binary code of the body responsible for each A.mon's actions and individual behavior is referred to as

(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

genetic code.

Each value in the genetic code will be referred to as a gene.

The source codes and executable files are found in the following link:

https://github.com/A0AUTHOR/A.mon

Starts Hatching section.

Imports necessary libraries.

```
package main
import("time";"os";"os/exec";"path/filepath";"io/ioutil";"math/rand";"strings";"fmt")
```

Declares important variables: location of genetic code in body's binary code, name of the directory tree in which A.mon can thrive in, mutation rarity and how long generations take to reproduce.

```
DNALOCATION:=100000
HABITATAREA:="HABITAT"
MUTATIONRARITY:=10
REPRODUCTIONRATE:=70
```

Starts Access of Genetic Code section.

Access and loads into memory, the binary code of their own executable file. Locates the genetic code within the binary code, using the variable containing its location (the genetic code is always located at the end of the binary code).

```
ANCESTORBODY,_:=ioutil.ReadFile("A.mon.exe")
ANCESTORDNA:=ANCESTORBODY[DNALOCATION:]
```

Starts Appearance section.

Go code creates a new file called APPEARANCE.exe in the current directory, and starts it. This file is a Python program, (created utilizing PyInstaller (PyInstaller Development Team, 2023)) able to generate and display an animated sprite which moves throughout the computer screen (by utilizing Tkinter library (Tkinter, 2021)), generated based on the A.mon's genetic code, its functionality is described as follows:

The genetic code from the A.mon is extracted.

```
ANCESTOR=open('A.mon.exe','rb')
DNA=ANCESTOR.read()
ANCESTOR.close()
```

Speed is generated from the last gene and ranges from 0 to 30.

Frequency of directional change is generated from the second last gene and ranges from 0 to 100.

(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

Eye colors (right and left, irises and scleras) are generated from the third last gene to the fourteenth last gene, generated colors are stored as hexadecimal color values.

Number of polygons that compose the sprite are generated from the fifteenth last gene and ranges from 1 to 10.

Forwards, the genes responsible for each characteristic will depend on the number of characteristics already generated, for example, if the sprite is composed by 10 polygons, 10 genes will generate the amount of vertices of each polygon, then other genes will generate the coordinates of each vertex.

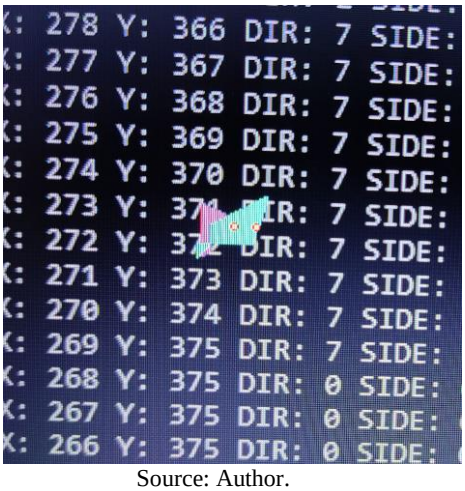
Polygon's colors are generated using 3 genes each, and are also stored as hexadecimal color values.

Coordinates of each vertex of each polygon are generated, using 2 genes each.

The animated sprite is displayed over the screen, without a window or borders and is able to freely move around, position of eyes and body changes depending on the direction the sprite is heading to and in order to make movement appears more natural, the sprite's polygons wiggles slightly. After 1 minute the sprite and program deactivate. (Figure 1)

```
SCREEN.after(0,SPRITEACTION)
SPRITE.after(60000, SPRITE.destroy)
SPRITE.mainloop()
```

Figure 1 – A functional visual component, in later stages of development.



Source: Author.

Starts of Action section.

There are two versions of the action section, both approaches are available in the links listed at the methodology.

Command Prompt Approach:

Genetic code is converted into a string, in which each value is converted into its correspondent ASCII character. The strings is used as a command for Windows' Command Prompt (Microsoft, 2023), 1000 character at a time.

```
exec.Command("cmd","/c",string(ANCESTORDNA[AINDEX:INDEXA])).Start()
```

Binary Approach:

An external independent executable file called ACTION.exe is created using the genetic code as binary code, then the file is ran by the A.mon.

```
ioutil.WriteFile(ACTIONEXECUTABLE, []byte(ANCESTORDNA), 0777)
exec.Command("rundll32.exe","url.dll,FileProtocolHandler",filepath.Dir(NOWDIRECTORY)+"\\"+
ACTIONEXECUTABLE).Start()
```

Starts Adventure section.

In addition to the variations of approaches in the Action section, and the inclusion or no inclusion of the Appearance section, some A.mon are able to travel to other computers, if they include the Adventure section, which functions in two ways, described below.

Furthermore, A.mon who are able to travel are not limited to one specific directory tree and are able to travel freely in the new computer they've arrived in. Since they probably will be inhabiting many directories, which could consume computing resources, their REPRODUCTIONRATE is 1800, meaning there is a 30 minute pause between reproductions.

Shared Folders:

A.mon sends a non-mutated offspring to all active shared folders which do not require administrator privileges.

Flash Drives:

A.mon sends a non-mutated offspring to all active flash drives connected to the computer which do not require administrator privileges.

Starts Sensory section.

A list of habitats is created containing all directories that are one level of distance from the directory the A.mon is inhabiting, meaning parent directory and child directories. The list is shuffled to ensure that the order of directories selected for offspring to inhabit is random.

```
NOWHABITAT,_:=os.Getwd()
UPHABITAT:=filepath.Dir(NOWHABITAT)
FILELIST,_:=ioutil.ReadDir(NOWHABITAT)
```

Starts Mutation section.

For each habitat present in the list produced by the Sensory section, one offspring will be created, thus the mutation section produces a mutated genetic code for each offspring. (Figure 2)

Figure 2 – A.mon specimen showing new visual characteristics, reflecting genetic code mutations throughout generations.

Source: Author.

(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

Habitats are excluded from this process if they do not contain, in their address, the directory tree name specified in the hatching section variable. This section can be erased from the code so they can inhabit anywhere in the computer.

A gene is a value between 0 and 255.

The mutation section has 3 parts, Alteration, Creation and Deletion.

The chances of each mutation happening is inversely proportional to the number of genes of the ancestor's genetic code multiplied by the mutation rarity declared in the hatching section.

100 ÷ (Number of Genes in Ancestor's Code × Mutation Rarity) = Chances of Mutation (in Percentage).

Each type of mutation may happen for each gene present in the genetic code.

```
if rand.Intn(len(ANCESTORDNA)*MUTATIONRARITY)==0{
```

A copy of the ancestor's genetic code is created, the offspring's genetic code, which will be mutated.

```
OFFSPRINGDNA:=ANCESTORDNA
```

Alteration: A gene may be transformed into other random gene.

```
OFFSPRINGDNA[ALT]=byte(rand.Intn(255))
```

Deletion: A gene may be deleted.

```
OFFSPRINGDNA=append(OFFSPRINGDNA[:DEL-D],OFFSPRINGDNA[(DEL-D)+1:]...)
```

Creation: A new gene may be created before the current gene being processed. Furthermore, after all genes are run through the mutation section, a new random gene may be created at the end of the offspring's genetic code, this last mutation also has the same amount of chances of occurring as the others.

```
NEWGENE:=[]byte{byte(rand.Intn(255))}
```

```
OFFSPRINGDNA=append(OFFSPRINGDNA[]byte{byte(rand.Intn(255))}...)
```

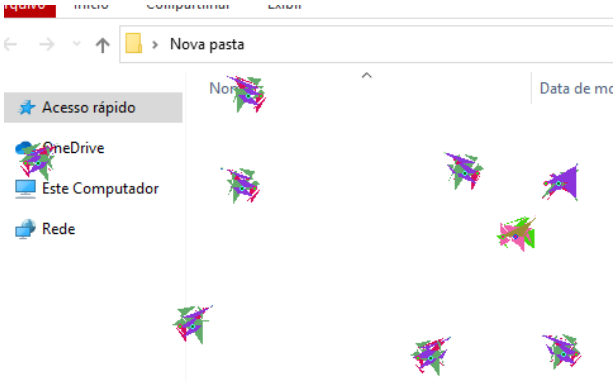
The mutated offspring's genetic code is then attached to a copy of the ancestor's body, which will be referred collectively as the offspring's body in the reproduction section.

Starts Reproduction section.

The ancestor creates their offspring in their respective habitat and starts the offspring's file. (Figure 3)

```
ioutil.WriteFile(SELECTEDHABITAT+"\\"+"A.mon.exe", []byte(OFFSPRINGDNA), 0777)
exec.Command("rundll32.exe", "url.dll,FileProtocolHandler",SELECTEDHABITAT+"\\"+"A.mon.exe").Start()
```

Figure 3 – Multiple A.mon specimens, moments after birth, including two showing new mutations.



Source: Author.

Activation instructions for interaction and tests with A.mon (default parameters):

Access and download the necessary files from the address https://github.com/A0AUTHOR/A.mon (choose between different variations of A.mon), to a computer running Windows 10 64bits, (other versions may also be compatible). Different versions are available: A.mon with the Command Prompt approach or the Binary approach, with or without the visual component and with or without the Adventure section.

Create a new directory on your computer and name it "HABITAT", inside such directory, created create 3 directories of any name, then inside one of the 3 directories, create a new directory of any name.

Move the A.mon.exe file to the HABITAT directory and double-click on the file.

Now the A.mon is active and inhabiting the directories, feel free to change the organization of the directory tree in future tests.

To deactivate A.mon, open Windows' task manager and manually close each A.mon.exe process, after 1 minute, the visual component will also deactivate.

Activation instructions for interaction and tests with A.mon (customized parameters):

Access and download the necessary files from https://github.com/A0AUTHOR/A.mon, to a computer running Windows 10 64bits, (other versions may also be compatible). Different versions are available: A.mon with the Command Prompt approach or the Binary approach, and with or without the visual component.

Create a new directory on your computer and name it as you please, inside such directory, create 3 directories of any name, then inside one of the 3 directories, create a new directory of any name.

Open the file "A.mon.go" with a text editor (".txt" file editor) of your choice, for example, Windows's Notepad, and, in the hatching section, edit the following variables:

```
HABITATAREA:="HABITAT"
```

The name you chose for the parent directory.

```
MUTATIONRARITY:=10
```

How rare mutations in each gene will be.

(CC BY 4.0) | ISSN 2675-8539 | DOI: http://dx.doi.org/10.52076/eacad-v4i2.455

100 ÷ (Number of Genes in Ancestor's Code × Mutation Rarity) = Chances of Mutation (in Percentage)

```
REPRODUCTIONRATE:= 70
```

Time, in seconds, A.mon wait after being activated and before reproducing.

Open CMD and compile A.mon.go utilizing the updated version of Go:

```
go build -ldflags -H=windowsgui A.mon.go
```

Rename the compiled file as A.mon.exe.

After compilation is complete, check the file size of A.mon.exe, if the file size in bytes is different from the current value of DNALOCATION in the hatching section, alter it so they're the same and compile the code again.

Make sure A.mon.exe is in the same folder as GENETICINJECTOR.exe and run GENETICINJECTOR.exe, which attaches a default genetic code to the A.mon.exe.

Move (or copy) the A.mon.exe file to the directory named HABITAT and double-click on the file.

Now the A.mon is active and inhabiting the directories, feel free to change the organization of the directory tree in future tests.

To deactivate A.mon, open Windows' task manager and manually close each A.mon.exe process, after 1 minute, the visual component will also deactivate.

3. Results and Discussion

A.mon are able to thrive, mutate and create new generations, their success surely depends on luck, however there are many factors, especially factors related to the their current digital environment that could hypothetically directly impact survival and could be topics of further research, they are described as follows:

The computer's ability of running multiple A.mon concurrently, which is directly tied to the amount of directories available to A.mon, since practically, only one specimen is capable of running per directory at a time, considering that directories stay locked and unable of being altered while having an active A.mon.

The time period between each reproduction: longer time periods allow for better user of the computer's resources, however it diminishes the amount of opportunities for mutations, while shorter time periods may allow for more opportunities, however it also may consume a large amount of computational power.

The presence of the appearance section, A.mon who do not display an animated sprite, are able to operate considerably faster, since creating and running the APPEARANCE.exe file takes significant time.

The approach present in the action and evolution sections: Command Prompt approach allows for easier emergence of functional actions, behaviors and abilities, since the genetic code functions as a literal Windows command, however the emergence of abilities that go beyond combinations of standard functions present in the computer may be more difficult. Binary approach allows for practically any possible binary to evolve, however, the chances of functional binaries emerging, especially ones with useful abilities, may not be probable.

The users who utilize the inhabited computer: even though A.mon are able to run in the background, and have an unobtrusive animated sprite display, their actions and behaviors may still bother the user, especially the Binary approach, since corrupted ACTION.exe files may cause notification windows to appear.

4. Conclusion

A.mon are analogous to primordial terrestrial lifeforms, however their physiology and evolution are inherently unrelated to biological life, analysis, time and further research are vital to the development and understanding of these life forms and their impact in the physical and digital environments.

The study of biology focuses solely on terrestrial life, a sample size of only one, it's unknown what characteristics are exclusive to terrestrial life and which are not. (Ray, 1991).

It is interesting to wonder if A.mon will experience the same major transitions in evolution as in biological life, replicating molecules becoming protocells, independently replicating genes becoming chromosomes, single-celled organisms becoming multicellular. (Maynard Smith & Szathmáry, 1999).

Will they experience a Cambrian explosion of their own? The biological Cambrian explosion represented such incredible evolutionary potential and was responsible for almost all major phyla. (Kauffman, 1995), what types of niches will they develop? Agents fill specific niches that when removed, are occupied by another agent, when groups of A.mon move or are transport to new computers, how would these niches change? In biological life the whole ecosystem responds with a cascade of adaptations that results in new agents occupying the niches. (Holland, 1995).

Future research and development may focus on constructing more efficient executable files, that occupy less space, integrating the visual component fully in the GO language, investigating which combination of mutation rarity and reproduction rate leads to more positive mutations, studying the dangers of the emergence of more advanced forms of transport between computers and emergence of novel abilities.

References

Holland, J. H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Helix Books.

Kauffman, S. (1995). *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity.* Oxford University Press.

Langton, C. G. (Ed.). (1995). *Artificial Life: An Overview*. MIT Press

Levy, S. (1992). *Artificial Life: A Report from the Frontier Where Computers Meet Biology.* Pantheon Books.

Maynard Smith, J., & Szathmáry, E. (1999). *The Origins of Life. From the Birth of Life to the Origins of Language*. Oxford University Press.

Microsoft. (2021). Windows 10 (Version 22H2) [Computer software]. Redmond, WA: Microsoft Corporation.

Microsoft. (2023). Command Prompt (Version 10.0.19041.746) [Computer software]. Redmond, WA: Microsoft Corporation.

Ray, T. S. (1991). *An approach to the synthesis of life*. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), Artificial Life II (pp. 371-408). Addison-Wesley.

Smith, J. M. (1988). *Evolutionary Genetics*. Oxford University Press.

Python Software Foundation. (2023). Python Language Reference, version 3.11.3. Available at https://docs.python.org/3/reference/ .

Python Software Foundation. (2023). random — Generate pseudo-random numbers. Retrieved May 7, 2023, from https://docs.python.org/3/library/random.html

Python Software Foundation. (2023). time — Time access and conversions. Retrieved May 7, 2023, from https://docs.python.org/3/library/time.html

Python Software Foundation. (2023). copy — Shallow and deep copy operations. Retrieved May 7, 2023, from https://docs.python.org/3/library/copy.html

Python Software Foundation. (2023). os.path — Common pathname manipulations. Retrieved May 7, 2023, from https://docs.python.org/3/library/os.path.html

PyInstaller Development Team. (2023). PyInstaller, version 5.10.1. Available at https://pyinstaller.org/en/stable/ .

The Go Authors. (2022). The Go Programming Language Specification, Version of December 15, 2022. Available at https://go.dev/ref/spec .

The Go Authors. (2022). time package [Computer software]. Available at https://pkg.go.dev/time

The Go Authors. (2022). os package [Computer software]. Available at https://pkg.go.dev/os

The Go Authors. (2022). os/exec package [Computer software]. Available at https://pkg.go.dev/os/exec

The Go Authors. (2022). path/filepath package [Computer software]. Available at https://pkg.go.dev/path/filepath

The Go Authors. (2022). ioutil package [Computer software]. Available at https://pkg.go.dev/io/ioutil

The Go Authors. (2022). math/rand package [Computer software]. Available at https://pkg.go.dev/math/rand

The Go Authors. (2022). strings package [Computer software]. Available at https://pkg.go.dev/strings

The Go Authors. (2022). fmt package [Computer software]. Available at https://pkg.go.dev/fmt

Tkinter. (2021). [Software library]. Python Software Foundation. Retrieved May 7, 2023, from https://docs.python.org/3/library/tkinter.html

Zimmer, C. (2001). *Evolution: The Triumph of an Idea*. HarperCollins Publishers.