Retruco: un intérprete para TIMBA

Retruco: an interpreter for TIMBA

Álvaro Frías Garay
FAMAF - Universidad Nacional de Córdoba,
Córdoba, Argentina
alvaro.frias.garay@mi.unc.edu.ar

Resumen: Presentamos Retruco, un intérprete para TIMBA (Terrible Imbecile Machine for Boring Algorithms); un lenguaje de programación creado en Argentina durante la década del 80 y que fuera utilizado en varios puntos del país, principalmente en materias de ciclo básico, o en los primeros años de una carrera en Ciencias de la Computación, como herramienta para presentar la programación estructurada. También analizamos herramientas similares a TIMBA. Se presenta una investigación respecto a las y los actores involucrados en su creación y quienes tuvieron contacto directo con el lenguaje, además del contexto previo a la creación del lenguaje, y su uso posterior.

Palabras Clave: lenguaje de programación, intérprete, historia, enseñanza de las Ciencias de la Computación en Argentina, computación desenchufada.

Abstract: We introduce Retruco, an interpreter for the TIMBA programming language (Terrible Imbecile Machine for Boring Algorithms); it was created in Argentina in the 80's and it was used mainly in High School education for an introductory course in programming, also as a tool to teach structured programming. Alongside this investigation we analyzed programming languages that have resemblances with TIMBA. An investigation is presented regarding the actors involved in its creation and who had direct contact with the language in addition to the context prior to the creation of the language, and its subsequent use.

Keywords: programming language, interpreter, history, teaching of Computer Science in Argentina, unplugged computing.

[1] Introducción

La preservación de los lenguajes se vuelve necesaria no solo en documentos con especificaciones, dado que esta nueva forma de literatura incorpora, además de los tradicionales emisor y receptor, un tercer actor: la computadora [1]. En este trabajo, entonces, se hace un esfuerzo en reconstruir el contexto histórico donde el lenguaje se creó y utilizó y, sumado a la falta de un compilador/intérprete disponible al público, se presenta el intérprete Retruco, el cual sigue las especificaciones originales del lenguaje, adoptándolas a las tecnologías actuales.

Received January 2022; Accepted March 2022; Published Jun 2022

[2] Trabajos Relacionados

Algunas herramientas educativas para la introducción a la computación son CARDIAC y Little Man Computer (LMC) que se originaron en la década de 1960 y comparten rasgos significativos con TIMBA, como ser herramientas básicas para enseñar conceptos de programación y computación. Asimismo estos programas no cuentan con una computadora digital convencional para funcionar, por ende disponen de una gran cantidad de simuladores implementados en computadoras actuales para implementar los programas en dichos modelos [2] [3].

La principal contribución de este trabajo, al igual que los simuladores de los lenguajes similares a TIMBA antes mencionados, es proporcionar un software para preservar el lenguaje y que resulte posible utilizarlo hoy en día.

[3[Contexto Histórico

La historia de los lenguajes de programación en Argentina se inicia junto con la llegada al país de la computadora Ferranti Mercury, también conocida como *Clementina*, la cuál contaba con un lenguaje de programación de alto nivel conocido como AUTOCODE, el cual era un lenguaje muy próximo al lenguaje matemático que servía para abstraerse de los detalles técnicos de la máquina en sí, dado que usaba un modelo de máquina ideal [4]. Posteriormente, debido a la poca capacidad expresiva del lenguaje, sobretodo en lo que concierne a cálculo de matrices, se creó el que puede ser considerado el primer lenguaje de programación argentino: COMIC (COMpilador del Instituto de Cálculo) [5].

Con la Dictadura militar de 1966 y tomando como punto de partida el trágico episodio de La Noche de los Bastones Largos, debido a los exilios de científicos e investigadores y la intervención militar a las universidades, llevó a que la investigación en computación en Argentina se viera pausada. Tomemos como ejemplo que el desarrollo de COMIC se viera detenido y nunca pudiera completarse, la falta de actualización en el Instituto de Cálculo, llevando a Clementina a una sobrevida y decadencia, y los escasos grupos de investigación durante la dictadura, que servían como apoyo a otras áreas como Economía o Física, y no sería hasta una década después que podría retomarse el curso de investigación en Ciencias de la Computación [6].

El documento más antiguo que se ha encontrado que menciona al lenguaje TIMBA es un cuadernillo de Computación para el CBC de la UBA (1985), donde se especifica que el lenguaje fue creado por un equipo de trabajo de la Universidad Nacional de San Luis, dirigido por el profesor Ing. Hugo Ryckeboer [7] como lenguaje para la enseñanza de la programación.

Es importante resaltar el contexto de la computación a nivel mundial en dicha época. La década de 1980 resultó en el boom de la microcomputación. Compañías como Apple e IBM en Estados Unidos comenzaban a comercializar las Apple][y PC, así como sus contrapartes inglesas Acorn y Sinclair vendían las Atom y ZX81 respectivamente. Todas las computadoras listadas anteriormente contaban con un intérprete de BASIC. En Argentina la clase media podía acceder a ellas aunque estaba

más apuntado a hobbistas. Esto puede explicar la principal peculiaridad del lenguaje; el uso de un mazo de cartas españolas, elemento común que se puede encontrar en cualquier casa, como principal fuente de datos para el lenguaje, y que el UCP, también llamado ejecutor, es decir el elemento encargado de ejecutar las órdenes que se conforman a partir de las reglas del lenguaje, sea uno mismo, [9] como se muestra en la Figura 1.

Figura 1. Una representación del UCP de TIMBA [8]

Aunque BASIC y TIMBA compartieron época, los dos lenguajes se encontraron en veredas opuestas en cuestiones fundamentales de su construcción: primeramente BASIC fue pensado como un lenguaje de propósito general de alto nivel para las computadoras de tiempo compartido de Dartmouth [10], mientras que TIMBA fue creado como un lenguaje de enseñanza, con construcciones más limitadas aunque igualmente expresivas, con un lenguaje que no necesita una computadora digital ejecutarse. Otra diferencia notable es que TIMBA se basa en el paradigma de programación estructurada, el cual se inició en la década de 1950 con

ALGOL, y hace uso de las estructuras principales de condición y repetición, mientras que BASIC, si bien tenía una estructura de *for*, también hacía uso de los *go to* y el *if* se usaba como un salto condicional, prácticas del lenguaje que ya para la época se consideraban dañinas [11].

TIMBA puede considerarse como un lenguaje enmarcado dentro de los *CS Unplugged*[1], es decir, un conjunto de actividades que se utilizan para enseñar conceptos de Ciencias de la Computación sin el uso de una computadora digital [10]. Hay que remarcar que TIMBA es un precursor de dichas actividades dado que su formalización se dio en la década de 1990, por lo menos diez años después de la aparición del lenguaje argentino, aunque comparte el mérito con otros lenguajes *unpluggled* como lo son CARDIAC y Little Man Computer (LMC).

CARDIAC fue creada en los laboratorios Bell en 1969 como una herramienta para demostrar el funcionamiento interno de una computadora. Consistía de una cartulina con piezas móviles que representaba una computadora de arquitectura Von Neumann, con CPU, memoria, I/O y un conjunto de instrucciones, contando con mnemónicos para escribir programas en lenguaje ensamblador [12] [13].

LMC presenta un paradigma similar a CARDIAC pero variando esta vez en que la computadora se presenta como una habitación con un hombrecito el cual cuenta con una calculadora de bolsillo para hacer cuentas y cajas de correo donde almacenar números, además de dos cajas donde recibe instrucciones mediante códigos de operación (*opcodes*) y donde deposita resultados [14].

Retomando TIMBA, durante esa época el lenguaje se portó a la computadora PECOS en un proyecto educativo que involucraba hardware, software y material didáctico [15]. Se ha encontrado evidencias de un libro titulado "Programa Educativo Para Colegios Secundarios. Área Computación", creado por Viviana Rubinstein el cual contaba con un *diskette* de "TIMBA", posiblemente un intérprete del lenguaje para la computadora, aunque no se ha podido recuperar dicho diskette [16][2].

TIMBA continuó siendo usada en el CBC de la UBA por un par de años y también se incorporó como herramienta la Universidad Nacional de Río Cuarto. Finalmente el lenguaje persistió en la Universidad Nacional de San Luis, hasta 2019 [8] como un pseudo-lenguaje simple que permitía la definición de algoritmos basado en el paradigma de la programación estructurada para la materia Programación I [17].

En la actualidad no hay referencias al lenguaje en el material de estudio de la materia [18].

[3.1] TIMBA a través del PECOS

Gracias a los esfuerzos de preservación de distintos actores actores comprometidos con el legado histórico de la computación argentina, es posible narrar en esta sección que hay PECOS todavía funcionando, y también hay diskettes con programas de PECOS, entre ellos TIMBA.

---

1 En español, Computación desenchufada.

2 Agradecemos a Gustavo del Dago por aportar la referencia.

Uno de los tantos que se cruzó con la PECOS fue Roberto Mandracchia [19]. Empezó su relación con la computadora en la escuela secundaria, una de las cuales formó parte del proyecto PECOS de escuelas secundarias, y varios años después lograría recuperar una, aunque ya se habían descartado varias junto con software, entre ellos diskettes de TIMBA.

Tras años de tenerla juntando polvo, en 2020 Roberto decidió revivir la vieja PECOS. Comenzó reconectando la corriente, pero le faltaba el software, dado que únicamente contaba con el BASIC tipo MSX.

Mediante una cadena de conocidos, integrada por nombres que ya se han nombrado anteriormente en esta investigación, Roberto contactó con Nicolás Wolovick que lo contactó con Gustavo del Dago que tenía en su poder diskettes de PECOS que tenía Viviana Rubinstein, y junto con algunos cambios en la ROM fue posible hacer correr el Sistema Operativo CP/M de la máquina, aunque la ROM de PECOS todavía no estaba completa.

Figura 2. Una PECOS preservada corriendo en la actualidad [19]

Finalmente, Roberto Etchenique despachó a Roberto Mandracchia una ROM original de PECOS y pudo conseguir el CP/M y los programas originales para los que fue hecho. Finalmente hay en Argentina una PECOS corriendo como hace más de 20 años. En conjunto con este trabajo de preservación de la máquina física, Gustavo del Dago continua trabajando creando un emulador basado MAME (*Multiple Arcade Machine Emulator*), un proyecto que se encarga de crear emulaciones para distintas arquitecturas, entre ellas la arquitectura Z80, de la cuál PECOS utiliza, con el fin de preservar la arquitectura de la micro-computadora en software.

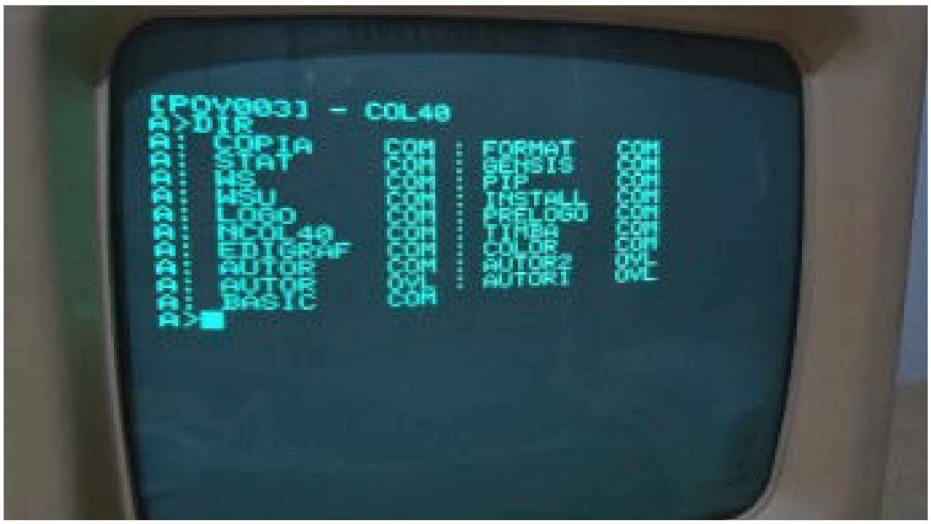

Figura 3. Software de la PECOS preservada, entre el cual se encuentra TIMBA [19]

[4] Lenguaje e Intérprete

[4.1] TIMBA

El lenguaje TIMBA está estructurado y tiene como única estructura de datos a la pila. Un programa típico de TIMBA se divide en dos: una parte de instrucciones al UCP y la otra encargada de definir las pilas que habrá al principio del programa y las cartas, si las hay, en cada una de ellas. Además de las pilas, el UCP en cualquier momento puede tener una carta en su "mano", una analogía de un registro, tomadas de una de las pilas, por lo que siempre comienza con la mano vacía.

El lenguaje cuenta con instrucciones operativas para mover cartas de la mano a las pilas y viceversa, e invertir la posición de la carta que se tiene en mano, pudiendo estar boca arriba o boca abajo. Cuenta también con instrucciones de condición (SI) y de repetición (MIENTRAS), ambas construcciones que pueden encontrarse en un lenguaje imperativo típico, contando los dos tipos de instrucciones nombradas anteriormente con proposiciones para obtener valores de verdad sobre la carta que se tiene en mano: la posición de la carta (boca arriba o boca abajo); el valor numérico de la carta comparándolo con un valor dado o con el valor de la carta que está en el tope de una pila; el palo de la carta, pudiendo hacer comparaciones similares a las del valor numérico: comparándolas con un palo dado o con el palo de una carta que está en el tope de una pila, así como también proposiciones sobre el estado de una pila dada: si está o no vacía. Dadas las proposiciones anteriores estas pueden combinarse usando los operadores de conjunción (Y) y de disyunción (O).

El lenguaje puede generar errores *e.g.* tomar una carta de una pila vacía, invertir la carta que se tiene en mano cuando no hay una carta en mano, evaluar una

proposición sobre la carta cuando no se tiene una carta o esta está boca abajo, etc. [9]. A continuación se presentan dos problemas resueltos en el lenguaje. El primero es un problema que sirve para mostrar todas las construcciones que posee el lenguaje. El segundo es un problema tomado de [7].

El primer problema es así: dada una pila de cartas llamada "PRINCIPAL"de la que se sabe que solo contiene cartas de palo Espada u Oro y, sin saber si están boca arriba o boca abajo, hay que repartir las cartas en dos pilas, llamadas "DE ESPADAS" y "DE OROS". En el programa se da una pila predeterminada aunque el programa es válido para cualquier pila de cartas que solo contengan cartas de palo de Oro o Espada en cualquier posición.

```
DEFINICION DE PROGRAMA
MIENTRAS LA PILA PRINCIPAL NO ESTA VACIA
TOME UNA CARTA DE LA PILA PRINCIPAL
SI LA CARTA ESTA BOCA ABAJO
INVIERTA LA CARTA
SINO
NADA MAS
SI LA CARTA ES DEL PALO ORO
DEPOSITE LA CARTA EN LA PILA DE OROS
SINO
DEPOSITE LA CARTA EN LA PILA DE ESPADAS
NADA MAS

REPITA

UCP EJECUTE CON LAS SIGUIENTES CARTAS
PILA PRINCIPAL TIENE 1 DE ORO ↑ - 1 DE ESPADA ↑ -
                    2 DE ORO ↑ - 2 DE ESPADA ↑ -
                    7 DE ORO ↑ - 7 DE ESPADA ↑
PILA DE OROS NO TIENE CARTAS
PILA DE ESPADAS NO TIENE CARTAS
```

Las sentencias de "DEFINICIÓN DE PROGRAMA" y "UCP EJECUTE CON LAS SIGUIENTES CARTAS" sirven como identificadores de las secciones del programa mencionadas anteriormente.

El segundo problema dice así (sic): Dada una pila B con al menos 2 cartas extraer el tope y la base y generar una pila con ellas (las demás cartas deben quedar en el mismo orden en la pila B). Una posible solución al problema se presenta a continuación.

```
DEFINICION DE PROGRAMA
TOME UNA CARTA DE B
DEPOSITE LA CARTA EN RESPUESTA
MIENTRAS LA PILA B NO ESTA VACIA
TOME UNA CARTA DE B
```

```
DEPOSITE EN AUXILIAR
REPITA
TOMA UNA CARTA DE AUXILIAR
DEPOSITE EN RESPUESTA
MIENTRAS LA PILA AUXILIAR NO ESTA VACIA
TOME UNA CARTA DE AUXILIAR
DEPOSITE EN B
REPITA

UCP EJECUTE CON LAS SIGUIENTES CARTAS
PILA B TIENE 1 DE ESPADA ↑ - 4 DE ESPADA ↑ -
               4 DE ORO ↑ - 4 DE BASTOS ↑ -
               4 DE COPA ↑ - 1 DE BASTOS ↑
PILA AUXILIAR NO TIENE CARTAS
PILA RESPUESTA NO TIENE CARTAS
```

El programa anterior no es trivial, y si bien es posible hacer un modelo mental del proceso de movimiento de las cartas en una pila, o usar una pila de cartas físicas, ambas alternativas se vuelven engorrosas, sin tener en cuenta además lo propenso a errores que es el ser humano en lo que refiere a realizar computaciones tan repetitivas y monótonas. Por esto, y teniendo en cuenta la masividad de las computadoras hoy en día, es que Retruco sirve como una herramienta en el proceso de creación y ejecución de programas de TIMBA.

[4.2] Retruco

El Intérprete que se ha desarrollado para TIMBA se lo ha llamado Retruco y ha sido implementado usando una máquina virtual (apodada TIMBA2000), una puesta en práctica similar a los intérpretes BASIC de los años '80, el cual cuenta con una serie de *opcodes*[3] de largo variable, que siguen una relación uno a uno con las principales construcciones del lenguaje. Por ejemplo, la instrucción de Tomar una carta de una pila se define como el *opcode* OS donde $S \in N \cup \{0\}$.

Como se vuelve engorroso escribir en *opcodes* de la máquina virtual se desarrolló un prototipo de editor que sirve también como generador de *opcodes*. Como se muestra en la Figura 4, el lenguaje es casi idéntico pero con pequeñas modificaciones al lenguaje original; las cartas que componen una pila se escriben por separado o pueden intercalarse, la división de sentencias no es mediante "," sino con saltos de línea, etc. Debido a que se pensó al lenguaje como introductorio a la programación, y siguiendo el diseño de lenguajes como *Scratch*, se limitó la entrada del usuario solo mediante botones, y otras restricciones.

Además del editor anterior también se está desarrollando un compilador de TIMBA a la máquina virtual.

---

3 Abreviatura en inglés para referirse a códigos de operación de la CPU.

Figura 4. El editor Retruco de TIMBA .

El intérprete es Software de código abierto y se encuentra todavía en desarrollo; puede encontrarse en [20], junto con una explicación más detallada de los *opcodes* del lenguaje y el funcionamiento de la máquina virtual.

[4.3] Preservación mediante el uso

Si bien se ha logrado la preservación del lenguaje mediante un intérprete que funciona en arquitecturas modernas así como la preservación del intérprete original de TIMBA en PECOS, la disponibilidad del lenguaje no implica que vaya a preservarse si también es olvidado. Tal es el ejemplo de tantos lenguajes extintos con idiomas. Es por eso que se plantean formas de volver a revitalizar el lenguaje TIMBA adaptarse a los tiempos modernos.

Si bien TIMBA comenzó y terminó su vida como un lenguaje de programación enfocado a la enseñanza, también es un producto de su tiempo. Las construcciones del lenguaje son similares a BASIC y se nota una excesiva verborragia del lenguaje para cualquier declaración de la gramática, similar en este sentido a COBOL. A continuación se realiza el ejercicio de reformulación de la gramática para adaptarla a la programación actual.

Sentencias Operativas. Si bien las sentencias operativas no son muy verborrágicas se podrían reducir a palabras simples.

```
TOME UNA CARTA DE LA PILA XX
```

podría reducirse a

```
TOME PILA XX
```

La operación de depositar una carta;

```
DEPOSITE UNA CARTA EN LA PILA XX
```

a

```
DEPOSITE PILA XX
```

La operación de invertir una carta en mano;

```
INVIERTA LA CARTA
```

a

```
INVIERTA
```

Esta simplificación de oraciones es similar a como se trata el código en lenguaje ensamblador manteniendo solamente estructuras básicas de Operandos y mnemónicos.

Sentencias de Selección son las sentencias de la forma

```
SI condición ENTONCES

{sentencias de selección/operativas}

SINO

{sentencias de selección/operativas}

NADA MAS
```

y

```
MIENTRAS {condición}

{sentencias de selección/operativas}

REPITA
```

Si bien estas no son largas en sí mismas, dado que sus delimitadores (SI, SINO, MIENTRAS, REPITA, NADA MAS) son similares a lo que se puede encontrar en lenguajes actuales, las condiciones son posiblemente las sentencias más largas de todo el lenguaje. Por ejemplo, la siguiente condición compara el valor de la carta en mano con la de una en una pila dada o su negación.

```
LA CARTA {ES/NO ES} DE

{IGUAL/DISTINTO/MENOR/MAYOR/{MENOR/MAYOR} O IGUAL} VALOR
QUE TOPE DE PILA XX
```

Para una especificación completa de las condiciones se refiere a [9]. Las condiciones en TIMBA definen valores de verdad principalmente sobre 3 características; el valor numérico, el palo o la posición de una carta dada, ya sea una carta que se tiene en mano o una carta que está en el tope de pila. Para disminuir su verborragia primero definiremos 3 funciones que supondremos son parte del estándar del lenguaje: VALOR, PALO y POSICIÓN. El dominio de las tres funciones será $MANO \cup \psi$. Donde MANO es el *string* que hace referencia a la carta que se tiene en mano y ψ es el conjunto de nombres de pilas definidos en la sección de definiciones de pilas del programa. El rango de VALOR será el conjunto de los valores de las cartas españolas exceptuando, 8's 9's.

El rango PALO será el conjunto de palos de las cartas españolas. El rango de POSICIÓN serán las *strings* ARRIBA y ABAJO. Reemplazando ahora las palabras comparativas por operaciones de comparación comunes en todos los lenguajes de programación modernos, es decir ==, !=, <, >, <=, >= y reemplazando la negación definida por la oración "NO ES"por ! podemos reescribir condiciones de una forma sintética. Por ejemplo, la condición conjunta

```
LA CARTA NO ES DE MAYOR O IGUAL VALOR QUE TOPE DE PILA
XX Y LA CARTA ES DE IGUAL PALO QUE TOPE DE PILA XX
```

a

```
!VALOR(MANO) >= VALOR(XX) Y PALO(MANO) == PALO(XX)
```

Una condición específica de las pilas es si está o no vacía. Para este caso vamos a definir otra función LARGO, que tiene como Dominio ψ y rango $N \cup 0$. La condición del largo de pila

```
LA PILA XX {ESTA/ NO ESTA} VACÍA
```

puede reescribirse a

```
LARGO(XX) {== / !=} 0
```

A continuación se reescribe el primer programa de la sección anterior con las reglas para sintetizar definidas:

```
MIENTRAS LARGO(PRINCIPAL) != 0
TOMAR PRINCIPAL
SI POSICIÓN(MANO) == ABAJO
INVIERTA
SINO
NADA MAS
SI PALO(MANO) == ORO
DEPOSITE DE OROS
SINO
```

```
DEPOSITE DE ESPADAS
NADA MAS
REPITA
```

Es de notar la reducción de verbosidad en el programa, haciéndolo así más legible y fácil de entender.

La inclusión de las funciones nombradas anteriormente también abren un abanico de posibilidades en cuanto a expresividad de condiciones del lenguaje. Por ejemplo, en lo que refiere al largo de una pila, la especificación original del lenguaje solo permite determinar si la pila está vacía o no, mientras que utilizando la nueva función junto con los operadores de comparación y valores en los naturales permite realizar expresiones mucho más ricas, por ejemplo

```
LARGO(XX) < 5 Y LARGO(XX) >= 2
```

El análisis y cambio de la sintaxis de TIMBA es un tema profundo a analizar a ser tratado en otra investigación.

[4.4] La TIMBA como juego

TIMBA ha sido un lenguaje educativo pero en la línea de preservarlo y que forme parte del presente, posiblemente se vea anticuado. La premisa de tener un lenguaje que maneje pilas de cartas es innovador incluso para estándares modernos, es por esto que TIMBA encaja mejor en tiempos modernos como un juego.

En la actualidad la ludificación de la programación es un tema ampliamente abordado y existe una gran variedad de videojuegos en donde la mecánica principal consiste en programar dado un lenguaje que existe en el propio juego, muchas veces con características acordes al mundo del que el juego es parte.

El propio lenguaje de TIMBA es en sí mismo una referencia a un objeto físico que se utiliza para jugar, el propio mazo de cartas, y una propuesta es que TIMBA forme parte de un videojuego como mecánica principal siendo la de escribir programas en TIMBA para manipular pilas de carta.

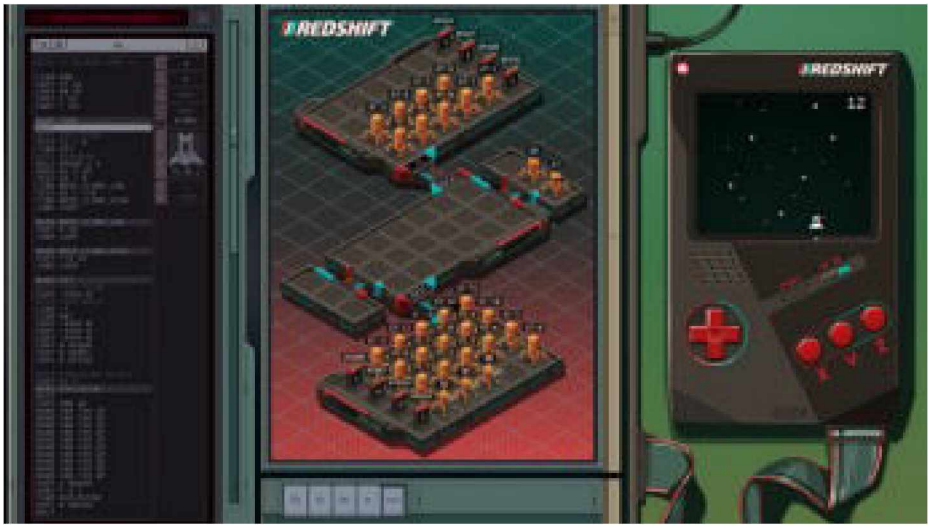

Figura 5. Ejemplo de un videojuego sobre programación, el TEC Redshift Player. A la izquierda se observa una consola con lenguaje del propio juego.

Debido a lo anterior y sumado al componente educativo del que el lenguaje que hace gala, siendo fácil de entender y de aplicar, es posible que TIMBA encuentre un hueco entre futuros programadores jóvenes como introducción en el mundo de la programación.

[5] Conclusiones y trabajo futuro

La investigación histórica que se ha realizado nos muestra la capacidad en materia de investigación de la Computación y la producción de lenguajes que Argentina demostró a principios de los ’60 con la llegada de *Clementina* que pudo habernos marcado como referentes en computación pero debido a los Golpes de Estado que desmantelaron la infraestructura científica-tecnológica nos llevó a perder esa oportunidad. TIMBA surge como un hito entre tantos otros, como la producción local de computadoras en el resurgimiento de la computación en nuestro país después de los oscuros sucesos de la dictadura. En este punto del trabajo, es posible afirmar la importancia de preservar y que sea accesible a todos y a todas no solo como una herramienta didáctica sino también como una muestra de nuestra cultura.

Cabe remarcar la labor de conservación realizada por distintos actores de arqueología computacional nacional en pos de preservar la micro-computadora PECOS, gracias a ellos se ha logrado reconstruir una de aquellas y mediante un trabajo en conjunto recuperar su software y un intérprete de TIMBA.

Aunque en este trabajo no se han aplicado técnicas tendientes a preservar el intérprete antiguo de TIMBA que funcionaba en PECOS, propias de arqueología computacional [21], sí se ha podido preservar el lenguaje en sí mismo, haciéndolo

disponible al público así como también documentación, testimonios y recopilación de diversos objetos materiales del legado cultural.

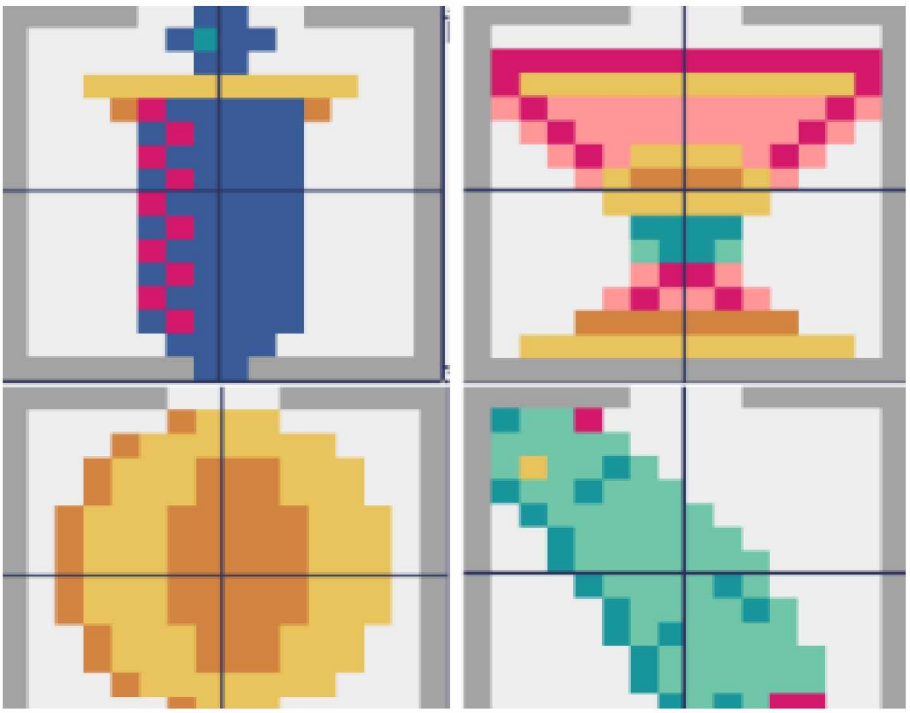

Figura 6. Primeras versiones de *sprites* de cartas para un juego de TIMBA.

Referencias

1.Annette Vee.: Coding Literacy, How Computer Programming Is Changing Writing. The MIT Press, (2017).

2. William Yurcik, H. Osborne. A crowd of Little Man Com puters: visual computer simulator teaching tools - ISBN: 0-7803-7307-3. - doi: 10.1109/WSC.2001.977496, (2001).

3.CARDIAC Simulator. Último acceso 2 de Julio de 2021. https://www.cs.drexel.edu/~bls96/museum/cardsim.html

4. AUTOCODE un sistema de codificación para la computadora Mercury. Instituto de Cálculo Facultad de Ciencias Exactas y Naturales. Universidad de Buenos aires. Último acceso 20 de Junio de 2021. https://elgranerocomun.net/IMG/pdf/El_Lenguaje_AUTOCODE.pdf

5. Wilfred Oscar Salvador Durán. COMIC EL LENGUAJE DE PROGRAMACIÓN Y COMPILADOR DEL INSTITUTO DE CÁLCULO EN 1965. Ediciones del Domo. Último acceso 13 de Julio de 2021. http://edicionesdeldomo.altervista.org/

6. Marcelo Carvalho. Continuidad formal y ruptura real: la segunda vida de Clementina ISBN: 978-950-665-573-0.

7. Computación Unidades 0 y 1. EUDEBA Editorial Universitaria de Buenos Aires (1985). Disponibilizado por Santiago Ceria. Último acceso 2 de Julio de 2021 https://archive.org/details/computacion-cbcunidad-0-1

8. Ricardo Medel, Comunicación Personal, junio 2021.

9. Lenguaje TIMBA. Universidad Nacional de San Luis. Último acceso 8 de junio 2021. http://dirinfo.unsl.edu.ar/servicios/abm/assets/uploads/materiales/23005-timba.pdf

10. First Basic Instruction Manual. Dartmouth College Computation Center. Mayo 1964. Último acceso 1 de Julio 2021. https://www.dartmouth.edu/basicfifty/basicmanual_1964.pdf

11. Edsger W. Dijkstra: Go To Statement Considered Harmful. Último acceso 1 de Julio 2021 https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf

11. Tim Bell, Jan Vahrenhold. CS Unplugged—How Is It Used, and Does It Work?. Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of his 60th Birthday. Springer International Publishing. 2018.

12. Drexel University. Computing and Informatics. Cardiac. Último acceso 2 de Julio 2021. https://www.cs.drexel.edu/~bls96/museum/cardiac.html

13. David Hagelbarger, Saul Fingerman. An instruction manual for CARDIAC. Bell Telephone Laboratories. Último acceso 13 de Julio de 2021. https://www.cs.drexel.edu/~bls96/museum/CARDIAC_manual.pdf

14.The Little Man Computer. Último acceso 2 de Julio de 2021. https://www.kean.edu/~gchang/tech2920/http___professor.wiley.com_CGI

15. Viviana Rubinstein, Comunicación Personal, junio 2021.

16. Catálogo Biblioteca ORT Sede Belgrano. Último acceso 3 de Julio de 2021. https://campus.belgrano.ort.edu.ar/biblioteca

17. Mauricio R. Dávila. Comparativa de Abordajes de Cursos Introductorios de

Programación. Último acceso 8 de junio de 2021. https://core.ac.uk/download/pdf/234157119.pdf

18. Programación I. Universidad Nacional de San Luis. http://proguno.unsl.edu.ar/material.html, página vigente al 09/06/2021.

19. Roberto Mandracchia. PECOS, una historia personal. Último acceso 15 de Mayo de 2022. https://homecomputer.com.ar/2022/05/10/pecos-una-historia-personal/

20. Página de GitHub del proyecto Retruco. https://github.com/qequ/retruco

21. Gustavo del Dago . Creación de un ecosistema donde preservar el primer lenguaje y compilador argentino: Un caso de arqueología computacional. Proyecto SAMCA.