De la programación hogareña al primer videojuego comercial latinoamericano. Análisis y estudio del Truco Arbiser.

Gustavo Del Dago

Universidad Nacional de José C. Paz
Pcia. de Buenos Aires
Argentina
gustavo.deldago@docentes.unpaz.edu.ar

Resumen El Truco Arbiser constituye uno de los casos más relevantes en la historia del desarrollo de videojuegos en Argentina. En el año 1982 Enrique y Ariel Arbiser (tío y sobrino respectivamente) iniciaron, como hobby, un proyecto cuyo desarrollo se extendió a lo largo de casi una década; al cabo de la cual fue tomando crecientes grados de sofisticación y complejidad materializándose en una serie de versiones compatibles con diversas plataformas. Mirando este proceso en perspectiva, vemos un auténtico proceso de aprendizaje no sólo respecto de las tecnologías computacionales sino también de las técnicas y prácticas propias del campo de los videojuegos. La comercialización de los resultados que exigió convertir las ideas y programas en productos cerrados, ocurrió en un contexto donde las personas copiaban y compartían el software. El mercado local estaba conformado mayoritariamente por "casas de computación" que vendían copias de los programas. El enfoque de trabajo bajo el que se lleva adelante esta investigación exige la búsqueda de objetos materiales y el análisis pormenorizado de los mismos. La intención que motiva y guía el esfuerzo es generar nuevos aportes y conocimientos sobre la historia del desarrollo de videojuegos en nuestro país.

Palabras claves: Videojuegos · Truco · QuickBASIC · PC/DOS · Preservación · Emulación · Arqueología computacional · Ingeniería inversa

Gustavo Del Dago

De esta forma ingresamos los argentinos en la era no solo de la denominada “INTELIGENCIA ARTIFICIAL”, sino también inauguramos la “PICARDIA ARTIFICIAL” con la creación del TRUCO ARBISER.[1]

1 Introducción

1.1 El objeto de estudio

La versión del Truco Arbiser que abordaremos aquí (Plataforma PC/DOS) es el resultado de un proceso de desarrollo que comienza como pasatiempo. Tiene lugar en un espacio de encuentro familiar entre Enrique y Ariel Arbiser. Enrique, tío de Ariel, hizo sus primeros pasos en la programación durante el año 1961 trabajando con los lenguajes Assembler y COBOL en equipos IBM-1401, IBM System/360 System/370 [5]. Tiempo más tarde, durante el año 1981, Ariel descubrirá la programación al tomar contacto con el lenguaje BASIC en una computadora TRS-80 Model I [5]. Durante el tiempo compartido, tío y sobrino proyectaban distintos juegos. En el año 1982, por idea de Enrique, fanático del truco, tomaron la decisión de programar una versión del juego destinada a la computadora TRS-80. El proyecto comenzó con el diseño y programación de las instancias del juego más simples (flor y envido) para avanzar luego con la incorporación del juego de truco. Prácticamente desde el inicio [1] incluyeron un sistema de versos sofisticando, de esta manera, la interfaz de usuario y ofreciendo una experiencia de juego muy a tono con la temática. De acuerdo al manual en formato digital [1] hallado durante este trabajo de investigación junto a una distribución no oficial, el programa de Truco tuvo una importante difusión en algunos medios masivos de comunicación: periódicos, revistas y programas de televisión. En el año 1983 la empresa Texas Instruments Argentina adquiere los derechos del videojuego y contrata a los autores para escribir una versión compatible con la computadora Texas Instruments TI-99. Circunstancia que convertiría a sus creadores en los productores del primer videojuego comercial de Argentina [8] cuyo reconocimiento formal se debe a la labor de la Asociación de Desarrolladores de Videojuegos Argentina (ADVA) en el marco de la Exposición de Videojuegos Argenitna (EVA 2013) [10]. Si bien el trabajo sobre la versión compatible con la plataforma PC/DOS se extenderá entre los años 1984 y 1992, podemos afirmar, siguiendo el historial presentado en la página de uno de los autores [2], que el grueso del trabajo tuvo lugar en el período 1984-1986.

1.2 Motivación

Truco Arbiser es, en nuestra opinión, un videojuego icónico a nivel nacional. Disponer de una distribución oficial y libre del juego (algo que debemos a la

generosidad de sus autores) habilita, además de nuevas experiencias lúdicas, la posibilidad de convertirlo en objeto de estudio. Las diversas fuentes de información disponibles nos permiten conocer una parte importante de su historia; sin embargo, creemos que el enfoque de trabajo que aquí seguimos podría aportar nueva información. En este sentido, la disponibilidad de objetos materiales se vuelve un requisito clave en la medida que se convierten en insumo de un trabajo de ingeniería inversa mediante el cual se intentará reescribir el código fuente. Estudiar el código fuente resultante de este proceso nos permitirá, cuando tengamos certezas respecto de su equivalencia semántica y semejanza sintáctica, abordar instancias de análisis a fines de develar sus modos de funcionamiento interno y, eventualmente, algunas técnicas de programación empleadas por los autores o funcionalidades que, mediante el mero uso, sería difícil o laborioso descubrir.

Figura 1. Truco Arbiser. Pantalla de inicio.

Este trabajo se estructura en tres partes principales. En primer lugar se describen los artefactos digitales disponibles (versiones de los programas y documentación) señalando algunas relaciones entre objetos y testimonios. A continuación, se presenta la tarea de reescritura del código fuente de la versión del Truco Arbiser para plataforma PC/DOS mediante el cuál se obtuvo una versión semánticamente equivalente al original. Finalmente, se presentan los primeros hallazgos producidos durante el estudio del nuevo código fuente.

Gustavo Del Dago

> Aunque parezca mentira, y según palabras recibida(s) de usuarios a través de BBSs hace algún tiempo, el TRUCO Arbiser fue el programa más "pirateado" después del DOS.[2]

2 Artefactos

De acuerdo al enfoque metodológico que seguiremos en este trabajo de investigación, los artefactos constituyen una de las piezas de mayor importancia. Tomaremos como base el material disponible en el sitio WEB de uno de los autores [2], al que denominaremos de aquí en adelante "oficial", e incluiremos los hallazgos realizados mediante la exploración de archivos y repositorios disponibles en Internet. De esta forma, estaremos en condiciones de analizar tanto las características de la versión oficial, cosa que haremos mediante técnicas de ingeniería inversa, como las eventuales diferencias entre ésta y las distintas versiones de circulación no oficial que se han hallado hasta el momento. En la página oficial se refiere el lanzamiento o liberación de cuatro versiones para la plataforma PC: 1984 Versión inicial (GWBASIC), 1985 Versión definitiva (BASCOM), 1986 Pocas modificaciones y recompilado y 1992 Agregado módulo de voz humana. Las distribuciones no oficiales presentan diferentes números de versión, fechas de producción o publicación y modificaciones en los datos de autoría. El material hallado se catalogó empleando identificadores únicos que denotan el archivo o repositorio de origen. Con intención de lograr una clasificación que permita comparar los programas y archivos se optó por considerar el tamaño del archivo ejecutable (TRUCO.EXE) como atributo clave. En el Cuadro 1 se puede ver el detalle de las distribuciones analizadas en este trabajo.

Del análisis del conjunto de programas y archivos señalados surgen algunos datos que es importante considerar. En primer lugar, encontramos que la versión oficial publicada (OFICIAL) es la 5.5 y no la 5.4 como se menciona en la página oficial [2]. Se trata de un programa generado mediante el compilador Microsoft QuickBASIC 4.0 que incluye, además, el módulo de voz humana. Si bien en el archivo comprimido disponible no se encuentran los archivos de audio correspondientes, estos se pudieron incorporar y funcionaron de acuerdo a lo esperado. Teniendo esto último en cuenta, podríamos suponer que se trata de la versión denominada "1992 Agregado módulo de voz humana". Los archivos ejecutables distribuídos en TOSAC223 y TDC2 fueron generados mediante el programa IBM Personal Computer BASIC Compiler 2.0, tienen una longitud de 50688 bytes y difieren entre sí únicamente en las cadenas de caracteres de los textos de autoría. Considerando el tamaño de los archivos ejecutables y suponiendo que las modificaciones realizadas por los distribuidores se limitaron al cambio de cadenas de caracteres, es posible conjeturar que se trata de la versión "1986 Pocas modifi-

Distribución	Tamaño	Mensaje de la pantalla inicial
OFICIAL	72101	(c)1982-1986 Por Ariel Arbiser y Enrique Arbiser VERSION 5.5
TOSAC223/TDC2	50688	ITALSOFT 21-4535 SOFT. Blandengues 1217 Capital Federal... VERSION 5.3
TOSAC419/TDC4	47104	POR ARIEL Y ENRIQUE ARBISER (c)1982, 1983, 1984, 1985, 1986 VERSION 5.2
TDC1	47104	By TRANSWORLD SYSTEM- Rep.Argentina (c)1983, 1984, 1985, 1986, 1987 VERSION 3.4
TOSAC607/TDC3	47109	By ITALSOFT COMPUTACION Te:21-4535 (c)1988, 1989, 1990, 1991, 1992 VERSION 6.4

Cuadro 1. Distribuciones del Truco Arbiser analizadas. La columna "Tamaño" muestra la longitud del archivo TRUCO.EXE expresada en bytes.

caciones y recompilado". La comparación a nivel binario de los archivos de las versiones halladas en las distribuciones TOSAC419, TDC4, TDC1, TOSAC607 y TDC3 no presentan más diferencias que las ya mencionadas en los textos de autoría y, en los casos de TOSAC607 y TDC3 la adición de la cadena de caracteres "MsDos" al final del archivo. Todas las versiones fueron generadas mediante el mismo compilador IBM Personal Computer BASIC Compiler 2.0. Lo anterior nos permite suponer que el archivo que forma parte de TOSAC419/TDC4 es la versión 5.2 oficial, correspondiente a la denominada "1985 Versión definitiva" distribuída de maneras no oficiales. Es curioso observar que la fecha del archivo TRUCO.EXE es en todos los casos exactamente la misma (04/07/1987), especialmente cuando se la compara con la fecha de la distribución OFICIAL (20/06/1986). Al observar los textos de autoría de las diferentes distribuciones se evidencia que la intervención de los programas a fines de cambiar mensajes y textos era una práctica común por parte de algunas personas involucradas en la distribución no oficial de los programas. Además de constituir un hecho de apropiación de autoría, al realizar modificaciones en los números de versión y años de producción se están presentando como "novedades" programas que son exactamente iguales. El hallazgo de este conjunto de distribuciones aporta un dato concreto que apoya la hipótesis sobre la amplia circulación por vías no oficiales que tuvo el Truco Arbiser. A los fines de este trabajo de arqueología computacional, la preservación y conservación de las versiones apócrifas constituyeron un aporte sustancial en la medida que nos permitió, entre otras cuestiones disponer de: una variedad de programas ejecutables de diferentes versiones, el manual del juego en formato digital, los archivos de audio necesarios para probar el módulo de voz humana, un programa de edición de archivo de versos (posible herramienta utilizada por los desarrolladores originales), archivos de lotes de

comandos, utilidades y configuraciones para ejecutar el juego en sistemas con placas de video Hércules. Este último asunto nos parece muy significativo en la medida que permite apoyar una conjetura sobre la configuración de los equipos en el ámbito local ya que todas las distribuciones están preparadas para lanzar automáticamente el programa SIMCGA necesario para simular las funciones de las placas CGA (requisito del juego que estamos analizando) generando gráficos compatibles con las placas Hércules, ampliamente distribuídas en nuestro país.

3 Reescritura del código fuente

La tarea central de este proyecto de investigación consiste en la reescritura del código fuente del Truco Arbiser. El esfuerzo tiene por objetivo aportar nuevos objetos materiales (código fuente en un lenguaje de programación de alto nivel) que faciliten el análisis de las producciones a fines de develar y explicar las técnicas de construcción y los modos de funcionamiento evidenciando funcionalidades ocultas o difíciles de advertir mediante el uso y trazas de la evolución histórica del proceso de desarrollo. Se partió del código binario de la versión OFICIAL. El primer análisis del archivo TRUCO.EXE (72101 bytes) permitió confirmar que se encontraba "empaquetado", una técnica muy común en la época en que este desarrollo tuvo lugar. El proceso de desempaquetado dió como resultado un nuevo archivo de 82880 bytes de longitud que luego fue utilizado como base para el proceso de desensamblado. El código binario del programa bajo estudio fue generado mediante el compilador Microsoft QuickBASIC 4.0 cuyo compilador produce código binario puro mediante un proceso que, hasta donde se tiene conocimiento, no se encuentra documentado. Esta situación exigió que se aborde el estudio de la generación de código binario. Para ello, se configuró un entorno de trabajo conformado por un sistema operativo (MS-DOS 3.30), la herramienta de desarrollo (QuickBASIC 4.0), un depurador (CodeView 3.14) y algunas herramientas y utilidades básicas para la transferencia de archivos. Todas las aplicaciones y programas se instalaron en entornos de emulación DOSBox 0.74-3 y MAME 0.226, ambos ejecutando sobre un sistema GNU/Linux. Durante el proceso de desensamblado y análisis fue fundamental separar datos de código. El programa ejecutable contiene algunos recursos internos (principalmente textos de la interfaz de usuario y datos correspondientes a los sonidos y melodías) y gestiona datos residentes en archivos externos (los versos o cantos, los gráficos de los naipes y el gráfico de la portada). En una primera instancia se investigó el formato de estos archivos de datos y estructuras de datos internas con intención de explorar la posibilidad de lograr programas semánticamente equivalentes y sintácticamente semejantes al que dió origen al código binario desde el que se partió. Los resultados obtenidos durante esta instancia confirmaron la factibilidad técnica del proyecto y constituyeron una experiencia de acercamiento a la intimidad del Truco Arbiser.

El trabajo consistió en una serie de pasos que se siguieron de manera iterativa:

1. Ejecutar el videojuego en entorno de depuración tomando nota del comportamiento observado, del estado interno del programa y generando capturas de pantalla.
2. Generar código ensamblador. Identificando secciones de código, datos y estructuras.
3. Realizar el análisis estático del código ensamblador generado.
4. Efectuar algunas conjeturas sobre las posibles estructuras de datos y código fuente originales.
5. Escribir los programas correspondientes de acuerdo a las conjeturas realizadas.
6. Compilar los nuevos programas generando tanto los archivos binarios como los listados en lenguaje ensamblador correspondientes.
7. Comparar los fragmentos de códigos binarios y listados en lenguaje ensamblador.
8. Ejecutar los programas resultantes cotejando los resultados obtenidos.
9. Realizar el análisis dinámico a fines de alcanzar niveles de comprensión más profundos sobre el comportamiento del programa original. Instancia que favorece la elaboración de nuevas conjeturas.
10. Realizar los ajustes o modificaciones necesarios.

Siguiendo el esquema descrito se logró reescribir una variedad significativa de fragmentos del programa original. Estos fragmentos de código fuente reescrito constituyen un nueva fuente de información que permite validar los hallazgos que se comparten en el siguiente apartado.

4 Primeros hallazgos

4.1 De los saltos computados a los procedimientos

El código fuente del programa (ver Figura 2) presenta gran variedad de saltos computados, solución muy utilizada en las versiones de BASIC que no cuentan con la posibilidad de definir y utilizar procedimientos. Los saltos computados exigen expresiones numéricas y una serie de etiquetas o números de línea para cada uno de los valores. Durante la reescritura del código fuente no es posible determinar qué se empleó originalmente (etiquetas o números de línea). Encontrar la definición y uso de subrutinas identificadas mediante nombres, conviviendo con fragmentos de código organizado mediante saltos computados, nos hace suponer que los autores incorporaron las nuevas facilidades del lenguaje durante sus últimas intervenciones. Desde la perspectiva de nuestro estudio, constituye una prueba que apoya el testimonio de los autores respecto de las múltiples adaptaciones a las que fue sometido el código del programa. Algo que refuerza la hipótesis de que Enrique y Ariel transitaron un camino de aprendizajes compartidos sobre las herramientas de desarrollo.

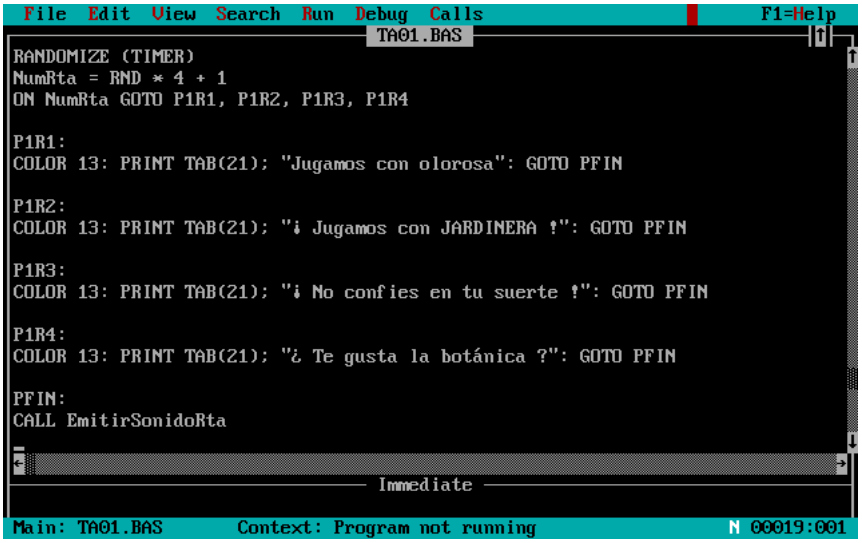

Figura 2. Fragmento de código fuente que emite una respuesta seleccionada de manera pseudoaleatoria entre cuatro posibilidades. Se aprecia la coexistencia de saltos computados e invocación a subrutinas.

Además, fueron nuevamente diseñados el dibujo de las cartas; el lenguaje de computación; el diagrama de flujo y las rutinas de Inteligencia Artificial (1), Picardía Artificial, movimiento animado de naipes, música, efectos visuales y sonoros, etc.[1]

4.2 Efectos de animación

El proceso de reescritura del código fuente develó la existencia de algunas características que habían pasado desapercibidas durante la ejecución del videojuego bajo el entorno de emulación (DOSBox). El juego incluye efectos de animación asociados a los cambios de posición de los naipes durante la dinámica de la partida. Por otra parte, presenta un efecto de transición entre pantallas cada vez que se termina una mano y una pantalla con el marcador que resume los puntos parciales obtenidos por cada jugador. En todos los casos se trata de rutinas y fragmentos de código que incluyen retardos logrados mediante la invocación de un procedimiento parametrizado. La implementación de dicho procedimiento se realizó mediante una estructura de repetición fija donde la cantidad de

iteraciones está parametrizada. Si bien el cuerpo de la estructura no contiene instrucciones, la variable índice es de tipo punto flotante. La demora se produce, en consecuencia, a partir de las operaciones de incremento y comparación de dicha variable. La implementación de bucles de espera mediante estructuras de repetición vacías es una técnica usada frecuentemente en programas escritos en lenguaje BASIC y destinados a computadoras hogareñas. En dichos escenarios, donde la frecuencia de operación de los equipos es fija, es posible establecer relaciones entre el número de iteraciones y el tiempo efectivo de proceso. En cualquier caso, es importante señalar que los resultados serán sensibles a la velocidad de operación de la máquina donde se ejecuten los programas. La plataforma PC se caracteriza por la amplia variedad de sistemas y configuraciones. La frecuencia del reloj es una de las características que ha mostrado cambios generalmente incrementales. Así, el número de iteraciones que en las computadoras disponibles en 1986 generaba retardos apreciables, se ha convertido gradualmente en un proceso que nos resulta instantáneo. Ya en la documentación que acompaña al Microsoft QuickBASIC 4.0 [6] se indican las limitaciones de emplear este tipo de técnicas a la que se menciona como propias de programas escritos para la versión BASICA del mismo lenguaje. Las consecuencias de este modo de programar retardos no afectó de manera exclusiva al Truco Arbiser. De hecho, constituyó un problema compartido por una gran cantidad de los primeros programas y videojuegos compatibles con la plataforma PC. Con el propósito de ofrecer una solución general a este tipo de problema, se diseñaron y desarrollaron diversos programas utilitarios con capacidad de regular la velocidad de operación de los equipos[1]. En la documentación del lenguaje BASIC [9] de la computadora TRS-80 (recordemos que se trata de la computadora que utilizaron Enrique y Ariel para programar la primera versión del juego) no solo se ilustra la técnica sino que se presenta la relación entre iteraciones y tiempo de proceso real. Culminada la instancia de análisis del código fuente y a fines de confirmar los hallazgos, se configuró el entorno de emulación ajustando la cantidad de ciclos de ejecución. Aún cuando el emulador DOSBox no permite, por su propia arquitectura interna, ajustar la frecuencia de emulación de manera precisa, fue suficiente para observar las animaciones, transiciones y el resumen de puntuación entre manos.

> Si Ud. necesita insultarla para descargar su bronca, puede hacerlo... pero no se extralimite !... (Pruebe ofenderla con palabras insultantes u obscenas y verá como se enoja !!).[1]

[1] En el entorno de trabajo configurado para el estudio del Truco Arbiser utilizamos *AT-SLOW (Version 4.10) PC-AT High Performance Slowdown By David Keil* (1992-1997)

4.3 Palabras insultantes

La interfaz de usuario está basada en el ingreso de sentencias escritas que son analizadas en búsqueda de comandos específicos como “carta 1”, “envido” o “truco”. El mismo manual nos invita a probar con otras alternativas como “truque” o “envidito”. Mediante el análisis del código fuente se pudo elaborar la lista completa de palabras reconocidas. Las palabras de la categoría insultante (ver Figura 3) nos permiten pensar sobre qué usos y significados se daban o atribuían a este conjunto de términos a mediados de los años ochenta en Argentina. Podemos afirmar que, de alguna manera, interactuar con un programa que reconoce órdenes emitidas empleando el lenguaje natural (aún cuando internamente coteja la aparición de términos específicos) y que responde a los términos obscenos o insultantes presenta un comportamiento de cierta sofisticación. No conocer de antemano la lista de términos, invita a ensayar con diversas frases y sentencias prestando atención a las respuestas. El carácter antropomórfico que se confiere a la máquina a lo largo del manual refuerza la idea de estar jugando frente a una entidad con algún grado de inteligencia y carácter; incluso, capaz de picardías. Creemos que, de esta manera, se habilita una experiencia de juego donde el uso del lenguaje natural favorece el involucramiento y la inmersión.

```
IF INSTR(Entrada$, "put")   OR INSTR(Entrada$, "mierd")  OR
   INSTR(Entrada$, "pij")   OR INSTR(Entrada$, "conch")  OR
   INSTR(Entrada$, "bolud") OR INSTR(Entrada$, "pelotu") OR
   INSTR(Entrada$, "caraj") OR INSTR(Entrada$, "chot")   OR
   INSTR(Entrada$, "fuck")  OR INSTR(Entrada$, "garch")  THEN
```

Figura 3. Fragmento de código que identifica palabras obscenas o insultantes.

4.4 Mensaje secreto

La introducción de la palabra mágica “XYZZY” en la aventura conversacional Colossal Cave Adventure por parte de Don Woods [11] y [7] dió inicio a una tradición en el mundo de la programación de videojuegos: la inclusión de palabras o comandos secretos. A modo de tributo, múltiples aventuras conversacionales incluyen la misma palabra. El análisis del código fuente del Truco Arbiser, nos permitió descubrir que éste también incluye un comando o palabra secreta. El programa responde a la palabra “!&^$v” cuyo reconocimiento produce el mensaje “Gráficos, música, adaptaciones a PC y coautoría: Ariel Arbiser” (ver Figura 4). Luego de analizar todas las versiones preservadas estamos en condiciones de asegurar que se trata de una característica disponible sólo en la versión 5.5. Es posible suponer que se trató de un agregado con intención de reforzar los mecanismos de protección autoral.

Figura 4. Truco Arbiser. Detalle del efecto producido al ingresar la palabra secreta

Autores del Programa de Computación "Truco Arbiser": Ariel y Enrique Arbiser.[1]

4.5 Protección de autoría

El análisis de las cadenas de caracteres evidenció algunos textos como "a:*.exe >nul" o "Congratulations !." que, a primera vista, no tendrían relación con la lógica del juego. El estudio de los fragmentos del código fuente en los que se refieren dichas cadenas de caracteres develó la existencia de un subsistema de protección tal que, cuando se han modificado los datos de autoría, intenta eliminar una serie de archivos del juego y del sistema operativo de la máquina donde se está ejecutando. En las Figuras 5 y 6 se muestra la reescritura del código fuente de la rutina que implementa el subsistema de protección. Para facilitar la lectura se suprimieron fragmentos de código que difieren sólo en el archivo destino de la modificación o eliminación. El efecto de esta rutina, cuando se detecta la alteración de los nombres de los autores, consiste en la modificación (destructiva) del intérprete de comandos del sistema operativo (COMMAND.COM), de los archivos ejecutables y de recursos del juego (TRUCO.EXE, TRUCO.COM, MVYTRUC@, MWYTRUC@.BAS y MXYTRUC@) y en la eliminación de todos los archivos con extensiones EXE, COM, BAT y SYS en las unidades A:, C: y D: (ver Figura 7).

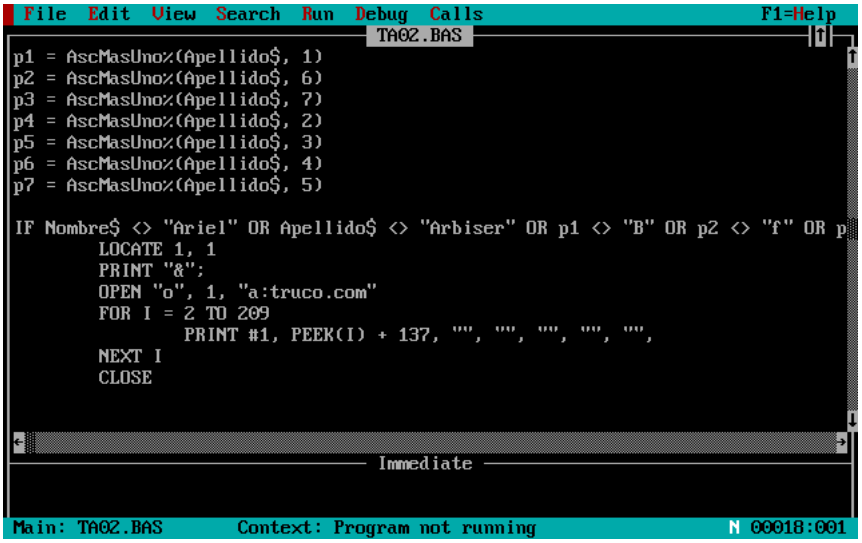

Figura 5. Control de datos de autoría. Detalle del fragmento que determina la alteración del nombre o apellido del autor y destruye el archivo TRUCO.COM

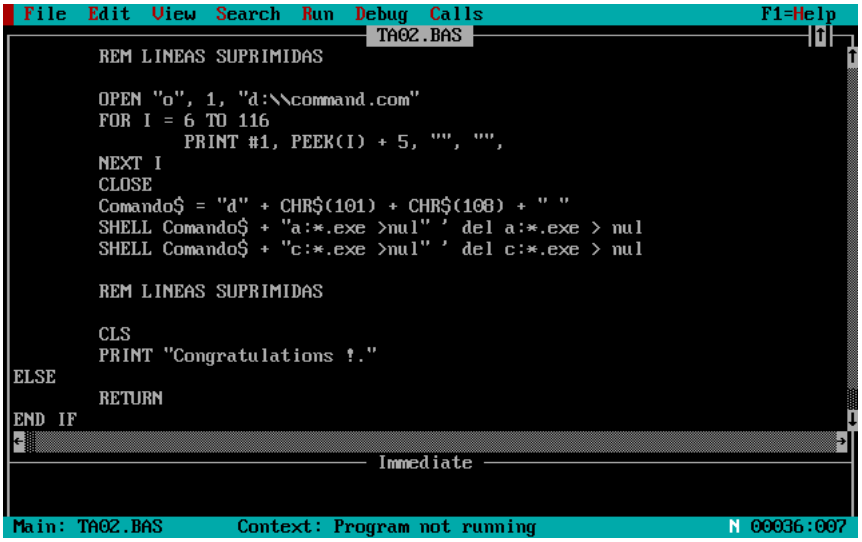

Figura 6. Control de datos de autoría. Detalle del fragmento de código que destruye y/o elimina archivos del sistema operativo.

> Los autores del programa "Truco Arbiser" no garantizan que estos programas estén libres de errores o que coincidan con los requerimientos específicos del usuario. [1]

El flujo de ejecución de la rutina encargada de destruir los archivos, que no contempla los posibles errores propios de esta instancia y que podrían tener su

origen en medios removibles protegidos contra escritura o unidades inexistentes en el sistema anfitrión, se interrumpe frente a la aparición del primer error. Durante el análisis del código fuente se advirtió la consecuencia más importante. En una gran cantidad de escenarios (todos aquellos que no tengan un diskette desprotegido en la unidad A:), se interrumpirá la ejecución de esta rutina y se anulará el efecto destructivo.

```
                CONTROLA_TODO_MAL
cd9b b8 01 00   MOV     AX,0x1
cd9e 50         PUSH    AX
cd9f 50         PUSH    AX
cda0 50         PUSH    AX
cda1 50         PUSH    AX
cda2 b8 04 00   MOV     AX,0x4
cda5 50         PUSH    AX
cda6 9a 13      CALLF   B$LOCT
     01 41 1d
                LOCATE 1, 1

cdab b8 f2 4b   MOV     AX,qbstr_and
cdae 50         PUSH    AX
cdaf 9a 61      CALLF   B$PSSD
     01 41 1d
                PRINT "&";

cdb4 b8 f8 4b   MOV     AX,qbstr_o
cdb7 50         PUSH    AX
cdb8 b8 01 00   MOV     AX,0x1
cdbb 50         PUSH    AX
cdbc b8 fe 4b   MOV     AX,qbstr_atrucocom
cdbf 50         PUSH    AX
cdc0 b8 ff ff   MOV     AX,0xffff
cdc3 50         PUSH    AX
cdc4 9a 2e      CALLF   B$OOPN
     01 41 1d
                OPEN "o", 1, "a:truco.com"

cdc9 b8 02 00   MOV     AX,0x2
cdcc e9 56 00   JMP     LAB_1000_ce25
```

Figura 7. Control de datos de autoría. Detalle del código desensamblado y del código BASIC obtenido mediante la tarea de reescritura.

A fines de estudiar y documentar el comportamiento de esta rutina se preparó un entorno de ejecución especialmente configurado a fines de garantizar que todos los comandos se ejecuten sin errores. Las pruebas permitieron evidenciar que, en ningún caso, terminará su ejecución de manera exitosa (ver Figuras 8 y 9). La interrupción será causada debido a un error en el código que intenta efectuar la apertura de un archivo mediante un identificador númerico que está siendo utilizado. Desde nuestro punto de vista, es interesante apreciar por un lado, el cuidado que se puso durante la programación para enmascarar la expresión

empleada en la estructura condicional que determina la activación de la rutina destructora de archivos y del armado del comando de borrado de archivos (DEL) y, por otro, el poco interés o falta de pruebas para confirmar el efecto resultante.

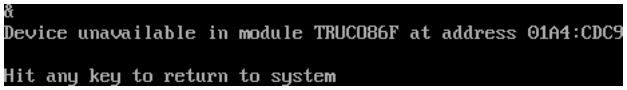

Figura 8. Control de datos de autoría. Detalle del error en tiempo de ejecución en un sistema sin medio magnético removible en la unidad A:

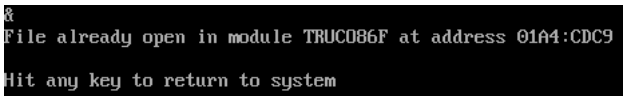

Figura 9. Control de datos de autoría. Detalle del error en tiempo de ejecución al intentar abrir el archivo TRUCO.COM estando en uso el número identificador.

5 Conclusiones

Tanto los testimonios de los autores como diversos pasajes del manual nos permiten afirmar que llevaron adelante tareas de observación y escucha atenta mediante las que lograron comprender las conductas y los comportamientos de las personas en interacción con el videojuego. Aún cuando, en este caso, nos animamos a conferirle un carácter espontáneo e informal, constituye una práctica compartida por la amplia mayoría de diseñadores y productores de videojuegos. En este sentido, pensamos que el trabajo conjunto de Enrique y Ariel Arbiser habilitó un espacio de reconstrucción de algunos conocimientos propios del campo de los videojuegos, incipiente hacia los primeros años ochenta. Las diversas implementaciones del programa a fines de lograr compatibilidad con diferentes arquitecturas y plataformas (TRS-80, TI-99, PC) exigió el aprendizaje de diversas herramientas y lenguajes. Un aprendizaje que suponemos situado y significativo. Situado en la medida que se tenía claro el dominio de un problema que presentaba ligeras variaciones (se trataba del mismo juego). Significativo dado que, al mismo tiempo, presentaba desafíos en el dominio de la solución (lenguajes de programación y plataformas) orientando de este modo el proceso de aprendizaje. El proceso de reescritura del código fuente presenta algunas pistas que apoyan la hipótesis de que el código de la última versión es el resultado de procesos de migración de código (otras formas de reescritura). Un ejemplo de lo que estamos afirmando lo encontramos en las técnicas para producir demoras en la ejecución que hemos analizado. Estas podrían haber sido aprendidas

durante las primeras prácticas en el entorno de programación disponible en la computadora TRS-80 y luego migradas hacia las nuevas plataformas y lenguajes. Por otra parte y aún cuando no se ha terminado la tarea de reescritura del código fuente completo, se pueden apreciar técnicas de programación previas al paradigma estructurado conviviendo con algunas propias de éste último. En este sentido, y como ya se ha señalado, podemos considerarlos elementos que registran momentos de aprendizaje sobre técnicas de programación y dominio del lenguaje empleado. Respecto de la rutina de detección de cambios en la autoría, se puede confirmar que se trata de una funcionalidad incorporada en una de las últimas versiones del programa (v5.5). Avanzando en terreno hipotético, podemos sugerir que la inclusión de dicha rutina de protección tuvo lugar no como una manera de impedir la circulación del juego y, en consecuencia, orientada a garantizar beneficios económicos, sino más bien como un mecanismo tendiente a reforzar unos derechos de autoría que, ha quedado demostrado con el análisis de diversas versiones preservadas, se veían avasallados reiteradamente. El efecto punitivo, eliminación de archivos y destrucción de archivos, parece tener como destinatarios sólo a aquellos distribuidores que intervinieran el programa atribuyéndose autoría sobre las producciones. El mensaje "Congratulations!", en nuestra opinión, tuvo desde siempre unos destinatarios específicos. Es importante recordar que todas las distribuciones no oficiales preservadas derivan de versiones oficiales previas a la inclusión de la rutina de protección de autoría.

6 Proximos pasos

> La "máquina-jugadora" no mira las cartas del jugador pero hace lo posible (cuando puede) por deducir (gracias a su inteligencia) o intuir (gracias a su "palpito") que cartas le pueden quedar al humano.[1]

Quedan muchos aspectos para seguir explorando. Entre los más interesantes se encuentra el estudio de las funciones y datos que determinan el comportamiento de la máquina. Los resultados obtenidos mediante las técnicas y métodos de la arqueología computacional aplicadas al estudio de videojuegos [4] y [3], nos animan a continuar con la tarea de reescritura del código fuente. Un esfuerzo que aportará nuevo conocimiento sobre la intimidad del Truco Arbiser. El proyecto historizador del desarrollo de videojuegos en nuestro país y en nuestra región, se enriquecerá en la medida que logremos convocar a las personas protagonistas y realizar tareas de preservación de materiales. Generando así un espacio fértil para el estudio de las producciones mediante diversas técnicas, enfoques y perspectivas.

Referencias

1. Arbiser, A.: Jugar al truco con la computadora (1987)
2. Arbiser, A.: Página del TRUCO Arbiser (1996), https://www-2.dc.uba.ar/charlas/lud/truco/
3. Aycock, J.: Retrogame Archeology. Exploring Old Computer Games. Springer International Publishing Switzerland (2016)
4. Del Dago, G.: Estudio del *Trucotron*. Aportes para la historia del desarrollo de videojuegos en Argentina. Anales del SAHTI - SHIALC 2020 II. SAHTI - Simposio Argentino de Historia, Tecnologías e Informática (JAIIO) VI SHIALC - Simposio de Historia de la Informática en América Latina y Caribe. Argentina (2020)
5. Lange, J.: El Pregón del pueblo. El Truco Arbiser (2002), http://www.nodoalem.com.ar/pregon/default.asp?verarticulo=102
6. Microsoft Corporation: Microsoft QuickBASIC 4.0. Programming in BASIC: Selected Topics (1987)
7. Montfort, N.: Twisty little passages : an approach to interactive fiction. The MIT Press Cambridge, Massachusetts London, England (2005)
8. Penix-Tadsen, P.: CULTURAL CODE. Video Games and Latin America. The MIT Press Cambridge, Massachusetts London, England (2016)
9. Radio Shack, A Division of Tandy Corporation: USER'S MANUAL FOR LEVEL 1. Radio Shack. TRS-80 (1977)
10. TecnoTrece: Ariel y Enrique Arbiser - Primer videojuego argentino El Truco (2013), https://www.youtube.com/watch?v=p6tgW2NVNws
11. del Vas, J.M.: La Aventura Colosal: Historia de las aventuras conversacionales. Plan B Publicaciones, Palma de Mallorca, España (2018)