Hackers, slackers, and shackles: el futuro del software libre

Créditos

» **Autor:** Matt Barton
» **Traductor:** Daniela López Seco
» **Fecha:** Sábado 3 de Junio del 2006

Introducción

Figure 1: Type This by Seb Brassard © 2005

Hasta ahora, nosotros, los jugadores de video juegos no nos hemos percatado de la continua batalla entre los desarrolladores de software libre1 y propietario. Después de todo, la mayoría empezamos a jugar ya sea en máquinas cerradas en oscuros locales de videos juegos o por consolas misteriosas con igual de misteriosos cartuchos. Pocas personas tenían alguna idea de cómo operaba el software, manejando esos maravillosos artefactos. Aquéllos de nosotros, que venimos de un ambiente de juego de computación estábamos mejor

preparados. Aunque las primeras computadoras personales como la Commodore, Atari, o Apple venían con los sistemas operativos propietarios conectados al hard, y mucho del software sólo estaba disponible en forma binaria, incontables jugadores de computadora de los años setenta y ochenta invertían sus tardes tipeando y a veces aprendiendo del código fuente impreso en revistas como Creative Computing (Informática Creativa). Mientras no es extraño oír de viejos entusiastas insistiendo con los buenos y viejos días" de buscar aquellos errores tipográficos críticos, hasta las primeras horas de la mañana solamente para averiguar un mes después que el problema era culpa de la revista algunas personas habrían estado más que contentas si pudieran haber comprado el programa en forma binaria y ahorrarse de relizar una operación de Túnel Carpiano. Por supuesto, uno tiene que tener en mente, que aquellos eran los días en que la producción en masa y la distribución del software no estaba tan desarrollada ni era tan eficiente como la industria de las revistas. Era más barato imprimir una colección de software en un libro o revista que incluir copias en cintas o diskettes. Además, en aquel tiempo, la mayoría de las campañas de marketing de computación se enfocaban en los aspectos educacionales y creativos de sus productos. La razón por la que los padres compraban a sus hijos una computadora en lugar de una maquina de juegos, era para que pudieran aprender lo suficiente para obtener un trabajo bien remunerado en la industria de la tecnología que estaba en ascenso. Si las computadoras pudieran conseguir llevar a algunos jóvenes americanos blancos a la luna, había una buena razón para creer que ellas podrían conseguir que Johnny y Jenny entraran a la universidad y aún más.

Hay algunos puntos que vale la pena remarcar. Primero, las computadoras como el Commodore 64 venían estándar con una versión de Microsoft BASIC (sí, Bill Gates ya tenia la mano en la lata de galletitas desde aquel entonces). BÁSIC significa Código de Instrucción Simbólico para todo Propósito para Principiantes, que es una manera elegante de decir que, básicamente, BASIC es bastante básico. Cualquier chico inteligente podría en poco tiempo deducir cómo hacer algunas cosas bastante ingeniosas con BASIC. Como podría esperarse, la mayoría de los chicos hicieron dos tipos de programas: Juegos y más juegos. Como Eugene Jarvis, creador del popular pero difícil juego Defender explicó: "El único uso legítimo para una computadora es jugar juegos". Me inclino a estar de acuerdo, aunque yo cambiaría el "jugar" por "crear".

El grueso de los primeros libros de programación estaba dedicado solamente a los juegos, y por una buena razón: los juegos eran, y continúan siendo, la mejor y más creativa manera de aprender sobre computadoras, software, matemática, ciencia, e incluso sobre la vida misma ciertamente no es coincidencia que la sustancia básica de la vida, el ADN, resulte ser expresada en código. De más formas que la mayoría de nosotros, tristes especimenes de homo lúdico comprendemos, la línea entre la realidad y lo virtual se está volviendo bastante delgada. Creo que la mejor evidencia de esto es, que la mayoría nos sentimos bastante cómodos con la noción de que la mayoría de nuestra riqueza monetaria existe como bits invisibles que viajan de un lado al otro en lo que nosotros esperamos que sean bases de datos electrónicas terriblemente seguras. Pasamos del estándar del oro, al estándar del papel, al estándar digital. ¿Puede imaginarse intentando convencer a un empleado de una tienda en el 1800 que sus descendientes pagarían por la mercadería con pequeñas tarjetas de plástico con bandas magnéticas en la parte de atrás? Ni siquiera entremos en el concepto de PayPal. Hemos recorrido un largo camino, y, crease o no, tenemos que agradecer a los videojuegos por la mayoría de esto.

Ahora, aprender nunca es una batalla cuesta arriba; o lo disfruta o nunca llega a ser bueno. Todo buen maestro sabe que los estudiantes apasionados o interesados aprenden mejor que los aburridos. Una manera de asegurarse que estén interesados es: presentar el tema en forma de un juego. Las oportunidades están, si usted no puede hacer un juego de él, no vale la pena aprenderlo entonces. Los buenos juegos, como las buenas novelas, hacen más que entretener: nos enseñan habilidades y nos dan una visión de la vida, del universo, y de todo.

El punto que quiero remarcar en este artículo es que los videojuegos deberían jugar un papel principalmente activo en hacer de nosotros individuos más inteligentes. Debemos evitar los juegos que ofrecen solamente una

experiencia fuermente controlada y completamente pasiva. Para llegar allí, debemos empezar a alejarnos del software propietario y adoptar el software libre2. También debemos prestar más atención a movimientos que convierten " jugadores pasivos en jugadores creativos, es decir, proyectos que permitan a jugadores tener un papel vital en el desarrollo presente y futuro del juego. Para reforzar esta idea comentaré un poco acerca de la historia del desarrollo de software y describiré algunos proyectos de software libre y de videojuegos de código abierto realmente interesantes en desarrollo hoy.

Como usted probablemente se habrá dado cuenta, estoy bastante entusiasmado acerca de las posibilidades que se abren para el desarrollo de algún videojuego realmente innovador no como los monolíticos dinosaurios de Electronic Arts; sino realizados por equipos de personas creativas, jóvenes y viejos, que se están reuniendo por todo el mundo para asegurarse de que la próxima generación de videojuegos no será solo más divertida para jugar, sino más plausible para cambiar e innovar.

Los juegos que los Hackers jugador:

Cualquier historiador de videojuego digno de su código ASCII sabe que no tendríamos un décimo del maravilloso software que tenemos hoy si no fuera por los hackers. De hecho, me sorprendería si tuviéramos incluso computadoras personales. Ahora, es importante remarcar que por "hackers" no me estoy refiriendo a delincuentes que se divierten irrumpiendo en websites corporativos u otros con nada mejor que hacer con su tiempo que "crakear" la protección del software propietario. Este tipo de hackers hicieron muy poco para estimular el progreso y de hecho lo han limitado tanto como el más avaro y estúpido CEO (chief executive officer, gerente o presidente de una compañía, en referencia a que solo persiguen el lucro). No, los hackers que tengo en mente son personas como Steve Russell, uno de los responsables de Spacewar. Spacewar era uno de los legados que pasó a nosotros por el Tech Model Railroad Club (Club del Ferrocarril de Modelo Técnico), un primer grupo de hacking con base en el MIT 3 (Massachusetts Institute of Technology). Cuando Russell estaba programando el Spacewar, no había computadoras personales, ninguna consola de juegos, y una idea muy remota de que el software era algo que podría llegar a hacer a alguien rico. Después de todo, había muy pocas computadoras en existencia, y bultos enormes como el PDP-1 costaban más de cien mil dólares y no lucían muy bien en tu living.

Muchos hackers han remarcado cuánto más rápido ocurrió la innovación del software sin abogados. Steven Levy, autor de Hackers, nos dice que en esos días, inclusive los programas extremamente importantes (impresos en cintas o tarjetas perforadas) eran simplemente guardados en un cajón abierto y estaban disponibles a cualquiera que pudiera beneficiarse con ellos. Los hackers ganaban el respeto de sus pares mejorando esos programas; si el "hack" era lo suficientemente bueno para ser incorporado en el creciente cuerpo del software, el hacker ganaba prestigio. El resultado final fue que aquel software mejoró rápidamente mientras otros hackers se esforzaban para impresionar a sus colegas con su hechicería. Cada hack exitoso incrementaba la apuesta. Spacewar es un ejemplo de lo que ocurre cuando los programadores de juegos no son limitados por derechos de propiedad o acuerdos de confidencialidad. Aquí, Steve Russell opinando sobre del tema:

Empecé con un pequeño prototipo que solo hacía volar las naves espaciales . Pete Sampson agregó un programa llamado Expensive Planetarium (Planetario Caro) que mostraba estrellas como fondo. Dan Edwards hizo algunas cosas muy inteligentes para obtener el tiempo necesario para que nosotros pudiéramos calcular la influencia de la gravedad en las naves espaciales. La última versión de esto se hizo en la primavera de 1962. (De Kent, 19)

El Spacewar no fue el único juego que se benefició del desarrollo del software libre. En 1972, un programador

llamado William Crowther tuvo una idea brillante: escribiría un juego de rol que pudiera jugarse en una computadora. Escrito en Fortran para las computadoras PDP-10, Crowther´s Adventures (Las Aventuras de Crowther) viajó por Arpanet (el precursor de la Internet) y pronto estuvo intalado en máquinas de todo el país. Cuatro años más tarde, otro programador llamado Don Woods encontró el código y decidió hacerle algunas mejoras. Se las arregló para localizar a Crowther, quien estuvo mas que contento al darle a Woods su permiso. Finalmente, un tercer programador llamado Jim Gillogly puso sus manos en el código y, con la bendición de Woods y Crowther, portó el código a C. Desgraciadamente, en 1981 la colaboración sufrió un duro golpe cuando un cuarto colaborador, Walt Bilofsky, pasó el código a IBM-PC. Bilofsky, rechazando el concepto de software libre, decidió que su migración debería ser registrada y venderse el software. Después de acordar pagarle a Crowther "una pequeña regalía". Bilofsky y Woods sacaron una "versión oficial. En su website personal, Bilofsky se jacta: "aunque muchas versiones de Aventure eran vendidas o distribuidas libremente, y aunque el juego generó un nuevo segmento en la industria del software, fuimos la única compañía que alguna vez les pagó algo a Crowther y a Woods. El no se molesta en mencionar que Crowther nunca pidió nada. Aun así, no seamos tan duros; por lo que sé, ni Bilofsky ni Woods hicieron un real esfuerzo para impedir que las personas copien y compartan el código del juego, y que las personas todavía estuvieran jugando con el código fuente hasta 1996, cuándo Paul Muñoz-Colman sacó una versión punto 370 para DOS.

Los fans de Linux podrían sorprenderse al saber que el sistema operativo en el que se basa, UNIX, es un producto derivado de los esfuerzos de un hacker para jugar su juego favorito. En 1969, un programador de Laboratorios Bell llamado Ken Thompson tenía algunos inconvenientes para ver apropiadamente un juego llamado Space travel (Viaje Espacial) en su GE 635. Thompson decidió migrar el juego a la computadora PDP-7. En el proceso, Thompson desarrolló un "paquete aritmético de punto flotante, la especificación correcta de los caracteres gráficos para la visualización, y un subsistema debugging que continuamente desplegaba el contenido de ubicaciones de errores en la esquina de la pantalla, innovaciones que conllevarían directamente a UNIX. Esta historia muestra cuan critica puede ser la conexión entre jugar y la creatividad, y cómo el tener acceso al código fuente --en este caso el código del juego Space Travel (Viaje Espacial)es a la innovación tecnológica 4.

Sorprendentemente, aunque la historia de la programación de software indica que la innovación y el desarrollo estaban floreciendo, Bill Gates fue capaz de convencer gradualmente a aquéllos no tan conocedores del tema, que toda esa libertad y acceso al código fuente estaban retrasando la producción del software bueno y de calidad. En su libro The Road Ahead ("El camino adelante") Gates cuenta una historia muy diferente de la historia temprana del desarrollo del software:

> En los primeros años de venta del Altair BASIC, nuestras ventas habían sido mucho más bajas que el uso masivo de nuestro software indicaba que debería ser. Yo escribí una ampliamente conocida "Carta Abierta a los Hobbyists", pidiéndoles a los primeros usuarios de computadoras personales que dejaran de robar nuestro software para que nosotros pudiéramos ganar dinero que nos permitiera construir más software. Pero que mi argumento no convenció a muchos hobbyists a pagar por nuestro trabajo; a ellos parecía gustarle y lo usaron, pero prefirieron pedírselo prestado a otros.

La "Carta Abierta" a la que Gates se refiere es un trabajo de sofisticación retórica. En esta carta, se puede ver una gran cantidad de retórica poderosa que le permitiría a él y a los de su tipo erosionar gradualmente las libertades que disfrutaban los hackers. A diferencia de los incontables hackers que vinieron después de el, Gates veía a la programación puramente como una manera que hacerse asquerosamente rico. En lugar de compartir su conocimiento para mejorar la sociedad, Gates quería esconderlo y protegerlo- tanto funcionalmente guardando los secretos de su funcionamiento para el, como éticamente convenciendo a otros entusiastas de las computadoras que era incorrecto compartir. Como la mayoría de los hobbistas deben saber, Gates escribe. la mayoría de ustedes roba su software. El hardware debe ser pagado, pero el software es algo

para compartir. ¿Quién se preocupa si a las personas que trabajaron en él se les paga? De hecho, casi 30 años después, todavía seguimos haciéndonos esa pregunta. De hecho, las únicas personas que aparentan saber la respuesta correcta a esto, son billonarios como Gates, cuyos productos mal hechos e inestables como Microsoft Windows y Microsoft Office han consumido la parte del león del mercado e inhibieron severamente las innovaciones publicas en la industria. Gates quería ponerle grilletes a los hackers, domesticar su Joven exuberancia y canalizar su energía creativa en una manera enferma de la cual el pudiera exprimir la ganancia máxima.

¿Como pudo Gates convencer exitosamente a tantos hackers quienes se habían beneficiado personalmente por la apertura del software, que encadenarlo era una buena idea? La respuesta es, por supuesto, que el no pudo. El blanco de Gates no eran los hackers quienes habían cortado los dientes en maquinas como la PDP-1 y sabían de primera mano cuan rápido el software mejoraba cuando se le permitía ser libre. Gates le debe su éxito a los fabricantes de computadores personales y a los cientos sobre cientos de nuevos programadores que fueron introducidos a la computación en la forma de código cerrado, sistemas operativos propietarios como Microsoft Basic y MS-DOS. Estos nuevos programadores no tenían idea de lo que se estaban perdiendo, ¿porque el software debía ser libre? Los viejos hackers miraban tontamente a estas oleadas de nuevos usuarios de computadoras con desprecio, después de todo, sus lamentables pequeñas cajas de consumidores palidecían en comparación con los gigantes mainframes con los que estaban acostumbrados a trabajar. Mas tarde, cuando se hizo obvio que el software podía venderse tan seguramente como el hardware, la mayoría de los hackers abandonaron sus ideales y saltaron en el vagón, firmando acuerdos de confidencialidad tan rápido como sus nuevos jefes preparaban contratos.

Sin embargo, no todos los hackers fueron tan rápidos para renunciar a su libertad. Ahí, entre las torres de la catedral que caían estaba parado un joven indignado con razón cuya integridad personal y filantropía siempre triunfaría sobre el deseo básico de mera riqueza física. Ese hombre era Richard Stallman, cuya trascendencia sólo ahora estamos empezando a entender. Stallman ha llegado a representar el Movimiento de Software libre, cuyos principios están en rígida oposición al pantano legal que impregna y retrasa al desarrollo de software moderno. El argumento de Stallman es que el software debe ser libre, no libre como costo cero, sino libre como la libertad de expresión. Rechazado por muchos como un soñador sin esperanzas, y despreciado como un rebelde peligroso, Stallman ha resistido las presiones puestas en el para aceptar el software propietario. Ahora, el éxito de GNU/Linux y otros incontables proyectos de software libre han incitado a muchos programadores a tomar la filosofía de Stallman mas seriamente. Este articulo es un intento por explorar las implicancias del movimiento del software libre para los juegos modernos.

La historia del software a mediados de los 80 es una demostración de la ineficacia del desarrollo de software propietario. Cuando las acaudaladas compañías de software inundaron con más dinero el Congreso esa vergonzosa práctica de soborno avalada por el estado que denominamos hacer lobby -- y finalmente se sancionó la tiránica "Ley de propiedad intelectual", y la innovación sufrió. La innovación se volvió la excepción en lugar de la regla. Los programadores retrocedieron y los trajes sastre avanzaron. El desarrollo del software era ahora un negocio cuyas ganancias dependían de mantener el código fuente secreto, restringiendo el acceso a sólo aquellos deseosos y capaces de pagar por él, y eliminando la competencia afianzando los monopolios avalados por el estado en forma de "Derechos de propiedad intelectual". Las personas que compraron este software se convirtieron en "usuarios finales" o para decirlo más correctamente, los "usuarios atrapados". Se les permitía usar sus programas solamente pagando, y podían usarlos sólo como sus dueños corporativos dictaran.

Desde luego, hay muchos lectores que me llamarían acertadamente para aclarar este punto. Yo me detendré por un momento aquí y trataré lo que yo espero serán las objeciones más comunes. "¿Usted habla en serio cuándo dice que no ha habido ninguna innovación reciente? ¿Usted está ciego? ¡Mire cuan lejos han llegado

los gráficos!"

Si comparamos un juego como el Spacewar con Half-Life 2, no puedo negar que ha habido una importante innovación, y no es mi intención. Ha habido innovaciones significativas en el Hardware a través de los años que nos permitieron a los diseñadores de software hacer sus imágenes más realistas. Seguramente, no podemos llamarlo un avance del software cuando es el hardware el que hace la diferencia. Lo que es aun más critico aquí es darse cuenta que la industria del hardware esta libre de las restricciones a la creatividad impuestas por la ley de derecho de autor. Eric S. Raymond, autor de The Catedral and the Bazaar, lo expresa mejor:

El mejor ejemplo en el mundo de los beneficios de la libertad en los negocios es comparar el negocio del hardware de computadoras con el del software. En el hardware, donde reina la libertad tanto para los proveedores como parar los consumidores de igual manera a escala global, la industria genera la innovación más rápida en valor de producto y cliente que el mundo haya alguna vez visto. En la industria del software, por otra parte, el cambio se mide en décadas

Por supuesto, alguien puede bien objetar que el hardware esta protegido por otro cuerpo de ley poderoso llamado sistema de patentes, las que son de alguna forma aún mas draconianas que la ley de derecho de autor. Sin embargo la historia nos muestra que las computadoras con "estándares abiertos", tales como la computadora compatible con IBM, finalmente triunfa sobre las computadoras con estándares propietarios tales como la Commodore Amiga o la Apple Macintosh. La razón de esto es fácilmente explicada bajo los principios elementales de la economía capitalista. Bajo el capitalismo, la innovación florece solamente cuando la maneja la competencia. Una compañía sin competencia no tiene razón de malgastar sus recursos en mejorar su producto, mientas que una compañía con una competencia dura tiene que trabajar mucho para bajar sus costos y a la vez mejorar la calidad del producto. Los consumidores se benefician inmensamente cuando hay una "mejor oferta" enfrente del circuito de la city. El estándar abierto de la IBM compatible triunfó porque a los fabricantes de hardware se les permitió competir entre ellos y así ofrecer a los consumidores un mejor trato. Si lo mismo hubiera pasado con el software todo este tiempo, estoy convencido de que estaríamos jugando en nuestros propias holocubiertas en este momento, y algunos de nosotros podríamos estar haciéndolo en una nave intergaláctica.

Reto a cualquiera que defienda el status quo para mostrarme algunos títulos nuevos innovadores de los principales desarrolladores si quitamos a esos juegos embarazosos que deben su existencia a una "licencia" de una película taquillera (o de un bodrio) y las terribles secuelas de secuelas, me atrevo a decir que hay pocos juegos que podríamos describir llanamente como "innovadores". Usualmente se escucha mucho acerca de ellos porque son tan raros y por lo tantos dignos de ser celebrados. La mayoría de los títulos son miembros casi idénticos de "géneros" pobremente definidos. Otro juego en primera persona salió a la venta y porque se gastaron millones de dólares en su desarrollo y promoción, se supone que debemos aclamarlo como un milagro de la ciencia moderna. Si se rompieran verdaderos paradigmas en Half-Life 2, Halo 2, y Doom 3, me gustaría enterarme.

Comencé este articulo con una discusión del comienzo de las computadoras personales y como servia principalmente a una función educativa. Mas de un chico entusiasmado encontró el error de programación y luego creó nuevos videojuegos y programas deslumbrantes. Desafortunadamente, muchos de estos chicos habían aprendido a guardar sus códigos y a no compartir sus innovaciones con otros. Los codificadores conocedores de negocios aprendieron a lanzar sus juegos solo como archivos ejecutables, y luego, mas tarde como archivos ejecutables de copia protegida. Habían tomado la idea de los juegos de consola. Vale la pena hacer una pausa aquí para reflexionar sobre las diferencias entre los juegos de consola y de computadora.

Los juegos de consola pueden ser mejor descriptos como "juegos cerrados". Los sistemas son propietarios y los juegos son típicamente entregados en formatos propietarios en medios propietarios. Todo esto esta diseñado para limitar lo que el usuario puede hacer con el sistema. La discusión es, por supuesto, si tal restricción es necesaria para el desarrollo de software. Después de todo, si la industria no puede evitar que la gente haga sus propias copias de los juegos, nadie va a querer gastar $50 en los nuevos títulos y los desarrolladores van a perder dinero. Se necesita una dura e impenetrable protección contra la copia para evitar que personas sin autorización accedan al modelo de software "pagar por jugar" que muchas personas asumen que es una gran motivación para la innovación. Después de todo, el costo de desarrollar un título clase A puede superar los $5 millones. Tales costos de producción increíbles solo pueden costearse atrayendo a inversores ricos y deseosos. Los inversores sobreviven minimizando el riesgo y maximizando la ganancia. Descarte los mitos de los capitalistas aventureros que se muestran como temerarios que corren riesgos y deseosos de subirse a cualquier nuevo proyecto prometedor que aparece en su puerta. Cualquier idea, sin importar su potencial de innovación, es detenida en el momento que parece riesgosa para los inversores: "Solo tome este derecho de propiedad de una gran película que acabamos de comprar y haga otro HALO, los chicos lo amarán."

Otra vez, tal cuestión genera muchas preguntas. Primero, gran parte de los $5 millones de costo de producción es gastado en maneras que a gatas promueven la innovación, como licenciamiento, marketing, seguros, y otro tipo de "costos corporativos" que no serían un impedimento para un programador de software libre. Otros costos, como contratar a un locutor o a músicos profesionales, pueden agregar algunos adornos interesantes, pero no se puede decir que realmente agregan innovación al desarrollo del juego. Pregúntele a un programador si alguna vez algún miembro del personal de marketing le propuso alguna sugerencia innovadora- pero no cuando está tomándose una Coca Cola. La industria moderna de los juegos es tan buena en hacer videojuegos como la industria moderna del cine en hacer películas. ¿Se acuerdan de Crossroads? (Nota del traductor: para el que no lo sabe, la película de Britney Spears, puaj!!!)

Se ve cada vez menos innovación en el Mercado de las consolas precisamente por su seguridad en aumento. El público esta inhibido de hacer copias ilegales de los juegos, pero también están inhibidos de aportar sus ideas para el desarrollo de nuevos juegos. El público es silenciado y se les niega opinar; no se les permite ver como los juegos que aman tanto están hechos. Históricamente, al público no le gusta que se le escondan secretos, particularmente cuando esos secretos juegan un papel tan critico en producir su cultura. Un buen ejemplo de lo que sucede cuando el público se cansa de tal tiranía ocurrió en 1517, cuando un monje enojado clavó una crítica mordaz a la corrupta y arrogante iglesia católica, en la puerta de una iglesia en Wittenberg, Alemania. El mismo hombre terminó para siempre con la naturaleza propietaria de la Cristiandad traduciendo la Biblia católica del latín, que solo era entendido por curas, al idioma local, que podía ser entendido por cualquiera que pudiera leer. Su nombre era Martin Luther, el Richard Stallman de su era.

Reivindicando la computación creativa

Hoy por hoy, una gran cantidad de "usuarios " están redescubriendo las raíces fundamentalmente liberadoras y potenciadoras de sus computadoras. Están empezando a entender al software como algo para interpretar como ellos lo aprueban, y aprendiendo a codificar y modificar programas como así a ejecutarlos. Estos nuevos hackers son, como cualquiera se puede imaginar, radicalmente diferentes de aquellos anteriores a ellos, no obstante tienen algunas poderosas ventajas: hay muchísimos más de ellos, y tienen Internet. Hay muchos libros buenos y algunos completamente malos por ahí sobre como Internet cambiará nuestra sociedad. Algunas de estas afirmaciones son tan ridículas como la ciencia ficción mas descabellada. A pesar de esto, una de las maneras en las que Internet nos cambió a muchos de nosotros que no se puede negar - ha acercado a muchas personas para compartir sus ideas e información, personas quienes de otra manera nunca se hubieran

conocido, menos aún construido juntos software de calidad comercial. Tal vez los grupos que se han beneficiado más visiblemente de esta nueva manera de comunicarse son los hackers modernos, mas comúnmente conocidos como programadores de software libre.

Es dudoso que Linus Torvalds supiera realmente en que se estaba metiendo cuando anunció que estaba "haciendo un sistema operativo libre" en un grupo de USENET en 1991. Con un grado de humildad y despreocupación que se ha convertido en una característica de los altos sacerdotes del movimiento de software libre, Torvalds ha llegado a representar todo lo que es bueno y esta bien el desarrollo moderno de software. Bajo su gentil representación, Linux ha crecido de un pequeño proyecto de hobbie en fuerza destructora de software libre que tiene incluso a Bill Gates temblando dentro de sus botas. El abrumador suceso de Linux ha inspirado a incontables otros programadores a unirse a el en retomar el control del desarrollo de software libre de las manos codiciosas de las mega-corporaciones. La mayoría de las nuevos programas publicados en Freshmeat, un boletín de anuncios para proyectos de software libre, son utilidades y aplicaciones, no obstante los juegos se están haciendo más comunes. Uno de los proyectos que se pueden encontrar ahí es Vega Strike, un proyecto de simulador espacial fundado por los hermanos Daniel y Patrick Horn.

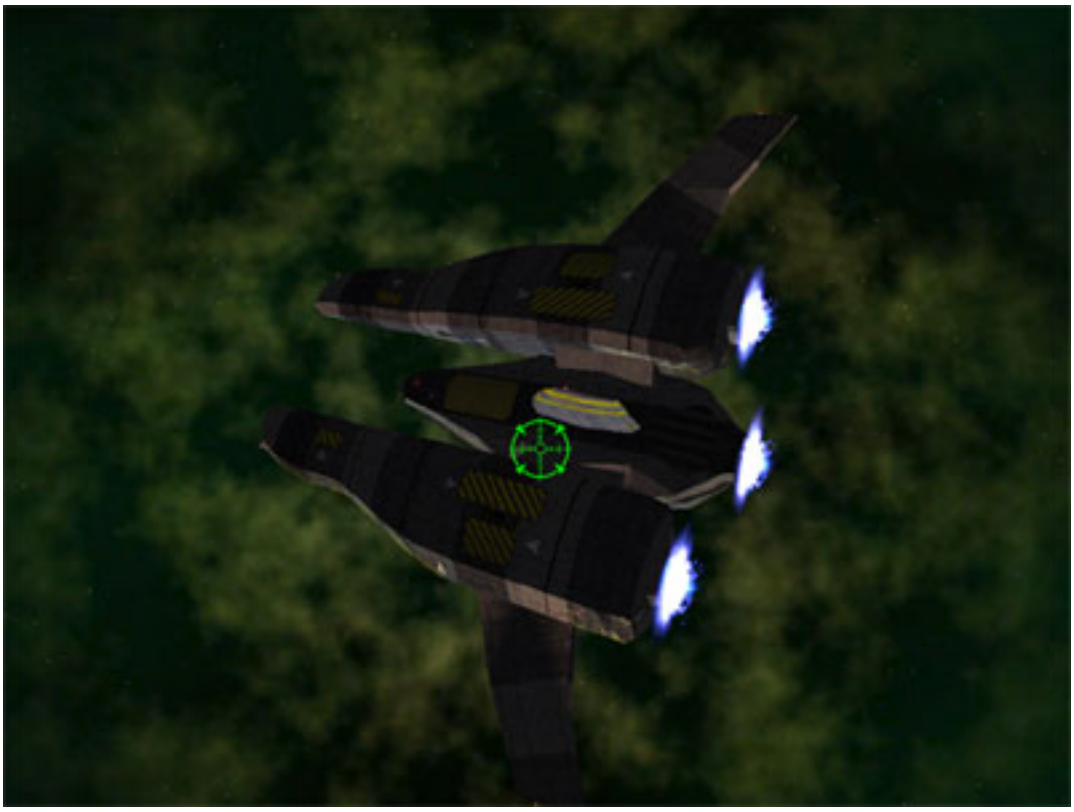

Figure 2: Vega Strike screenshot

Daniel dice que obtuvo la inspiración para Vega Strike del juego de DOS Privateer, un juego que aquellos con un poco mas de entorno histórico reconocerán como descendiente de Elite, el clásico tráfico espacial y simulador de combate Firebird. Todos estos juegos siguen una premisa similar- el jugador se convierte en el capitán de una nave intergaláctica que puede ser usada para el bien o para el mal - es realmente la decisión del jugador si jugar el juego como un honesto comerciante o un voraz pirata, es más que un pequeño aporte que uno de los grandes proyectos de código abierto esté basado en uno de los juegos "de juego abierto?". Algunas de las innovaciones de Vega Strike al género incluyen una ponderosa opción de multijugador, la habilidad de manejar una flota entera de naves espaciales, y muchas interesantes misiones, Pero lo más innovador es que, a diferencia de los juegos previos del genero, Vega Strike permite a los fans no solamente jugar el juego, sino

también ayudar con el desarrollo en progreso!

Originalmente, Daniel no tenia intención de que Vega Strike fuera de código abierto. Como muchos jugadores que se convirtieron en programadores, había crecido con los juegos propietarios de código cerrado, y tenia sueños de vender "bits secretos". Todo esto empezó a cambiar, sin embargo, cuando Daniel descubrió Linux y el desarrollo de software de código abierto. "Mientras aprendía sobre Linux", Daniel me contó, "Me di cuenta de que así debía ser. Esto seria lo que me diferenciaría de la competencia". Como todos los desarrolladores de software libre, Daniel hizo un trato con sus aficionados - les daría acceso a su código fuente, y, a cambio, ellos lo ayudarían a mejorar el juego y expandirlo. No tardó mucho tiempo para que el proyecto atrajera a fuertes talentos, incluido otro desarrollador de software libre que había trabajado en otro simulador espacial abierto. "El probó mi juego, se dio cuenta de que estaba muy avanzado, y se unió a mi proyecto," dijo Daniel. A pesar de que la participación fluye y refluye, Daniel cree que siempre habrá al menos 3 o 4 codificadores trabajando en mejorar, depurar o corregir errores en el programa. Sin embargo, muchas de estas contribuciones no vienen de codificadores, sino de hechiceros artistas gráficos que quieren usar Vega Strike para realizar sus visiones artísticas. Otros contribuyen escribiendo historias, componiendo música, inventando misiones, o trabajando en el manual, que está guardado en un tipo especial de sitio web llamado wiki. El wiki permite a cualquiera que lo desee editar el manual. Le pregunté a Daniel si él sentía que había dinero para ganar con el desarrollo de software. " si yo quisiera hacer dinero, tendría que empezar de nuevo. Probablemente borraría todo los archivos descargables y sacaría el código fuente y sólo lo ofrecería en CD. Las personas lo re-distribuirían, pero probablemente no lo harían". Este modelo funcionó para muchos otros desarrolladores de software libre, que se dieron cuenta de que muchas personas apreciaban más la conveniencia de ordenar un CD de calidad que bajar archivos tan grandes (particularmente si están limitados por conexiones a Internet lentas). Aún con el código abierto y archivos disponibles gratis online, Daniel ha logrado vender CDS valuados en $200. La táctica de vender CDS y varios accesorios (manuales, libros, remeras, tazas, etc) para patrocinar el software libre es una muy común; Richard Stallman la adoptó tempranamente - para patrocinar la Fundación de Software Libre y varios proyectos GNU.

Otra posibilidad que Daniel se planteó era lanzar el código bajo una licencia de software libre, pero registrando las imágenes, archivos de sonido y la historia del juego. "Toma mucho trabajo hacer un juego comercial solo para un motor" dice Daniel. "Tener un buen set de datos es la diferencia entre un buen juego o un mal juego." Sin embargo, hay un problema en este esquema: Muchos artistas actualmente trabajando en el proyecto muy probablemente no ofrecerían sus contribuciones si supieran que alguien lucraría con ellas. Le mandé un correo electrónico a Richard Stallman y le pregunté que pensaba de este modelo. El respondió con esta afirmación.

Un escenario de juego puede ser considerado arte / ficción más que software. Entonces está bien dividir el juego en motor y escenario, entonces considerar el motor como software y el escenario como Arte/ ficción.

Este compromiso entre el software libre y el software propietario parece tener sus ventajas. Los desarrolladores podrían seguir lucrando con su trabajo mientras comparten el código base. De esta manera, programadores principiantes tendrían la oportunidad de estudiar de los maestros. Aun si estos programadores decidieran crear y vender sus propios juegos basados en este código, sus juegos tendrían gráficos, sonidos, e historias substancialmente diferentes y así no competirían con el original en forma justa. En el caso que un programador haya deliberadamente copiado este material, el o ella podrían ser demandados por incumplimiento a la ley de derecho de autor.

Finalmente, otro modelo que Daniel propuso consistía en lanzar inicialmente el juego bajo una licencia propietaria y mantener el código cerrado, para más tarde, después de agotar su potencial de ventas, liberar el código al público. Daniel cita a Id software como un ejemplo de esta política. Id lanzó su juego original Doom como un demo de código cerrado; se les exigía a los fans que compraran la versión completa y no se les

permitía compartirla. Eric S. Raymond menciona el mismo juego en su ensayo "The Magic Cauldron" (El caldero Mágico):

Las tendencias técnica y de mercado elevan las ganacias por abrir el código; la apertura de las especificaciones de Doom y el estimular a los agregados de terceros, incrementó el valor percibido del juego y creó un mercado secundario a ser explotado. En algún punto, las curvas de beneficios traspasaron el límite y se hizo económicamente racional para Id pasar a ganar dinero en ese mercado secundario (con productos como antologías de juegos basados en un escenario) y después liberar el código del Doom. Después de este punto, realmente sucedió. El código completo de Doom fue liberado a fines de 1997.

Doom es claramente uno de esos juegos que pueden ser descriptos como "innovación", o, al menos "influyente". Pocos juegos pueden reclamar el honor de establecer un género entero, y, aunque los tiradores en primera persona (incluyendo el único éxito de Id Wolfenstein 3D) hacía tiempo que estaban rondando, Doom fue el primero en alcanzar realmente la verdadera masa critica. Irónicamente, Doom fue también responsable de establecer DirectX de Microsoft como una plataforma de juegos de PC válida. Esta es la historia de acuerdo a Brad King y John Borland:

La tecnología de Microsoft no tenía buena reputación con las aplicaciones multimedia; esperando quebrar el escepticismo, un talentoso programador de Microsoft llamado Alex St. John acudió a John Carmack y pregunto si la compañía podía hacer una versión de Doom que corriera en DirectX. Carmack aceptó y le dio a St. John el código fuente de Doom, y un grupo de programadores fue contratado por Microsoft especialmente para trabajar en este proyecto 7.

Quiero creer que Id se sintió culpable por ayudar a Microsoft a obtener tal dominio en el mercado de juegos de PC y por ofrecer el código fuente de Doom a Microsoft en una suplica por absolución. DirectX, como muchos de los productos de Microsoft, resulto ser un caballo de Troya para los desarrolladores. Una alternativa que todos los programadores deberían conocer es SDL, que ofrece a programadores como Bob Pendleton "una biblioteca que me permite escribir código sin tener que preocuparme demasiado por asuntos legales y no me fuerza a pagar una pequeña fortuna en derechos si desarrollaba algo que tuviera algún valor comercial".

Después de tener una fructífera discusión con Daniel Horn, decidí que seria justo obtener una perspectiva alternativa. Decidí contactar a Mike Boeh de Retro64 para obtener una perspectiva del mundo del desarrollo de juegos propietarios independientes. Me atrajo el trabajo de Mike por muchas razones. Además del hecho de que Retro64 hace algunos maravillosos juegos de calidad profesional, están comprometidos en revivir el espíritu creativo prevaleciente durante el reinado de la Commodore 64. Desarrolladores independientes como Mike pasan malos ratos compitiendo en un mercado dominado por mega corporaciones con presupuestos de millones de dólares. Desarrolladores indy exitosos sobreviven evitando la competencia directa y concentrándose en áreas que los principales desarrolladores ignoran. Retro64 se especializa en juegos simples y adictivos que no requieren hardware sofisticado ni una inversión de tiempo significativa; los tres criterios de Mike para un juego nuevo son, que sea fácil de aprender, totalmente controlables por mouse, y "adictivos o terapéuticos". Esa ultima cualidad, por supuesto, es la mas evasiva, pero no cabe duda que Mike aprendió a reconocerla e incorporarla en sus juegos. Juegos como Cosmo Bots, Platypus, y Bugatron ofrecen una experiencia de juego fundamentalmente diferente a la de los mega juegos como Half-Life 2. Mike ha ganado el suficiente éxito vendiendo sus juegos vía Internet como para mantenerse a sí mismo y a su familia; un objetivo sagrado para todos los desarrolladores indy.

Figure 3: Cosmobots Screenshot

La estrategia de desarrollo de Mike es muy diferente a la de Daniel. Los juegos en Retro64 son propietarios. En lugar de ofrecer juegos enteros gratis, Mike lanza demos jugables como un modo para que los jugadores "prueben antes de comprar". En breve, Mike hará lo que los grandes desarrolladores hacen, solo que en una escala mucho más chica. Mike argumenta que lo que mucha gente no sabe acerca de la programación de juegos es que hay mucho trabajo penoso involucrado. Muchos desarrolladores indy potenciales pierden interés poco después de desarrollar un prototipo:

La barrera de terminar un juego es del tamaño de la tareas. Hay que escribir tanto código. Cosmobots tomó unos 9 meses. Es fácil perderse en el camino. Si tenés una esposa e hijos, es difícil mantenerse concentrado. Es divertido el primer par de semanas cuando haces tu pequeño prototipo, y después el mundo real empieza. Cuando recién empecé Retro64 tenia un trabajo de tiempo completo que cambie a medio tiempo. De otra manera no podría haberlo terminado.

Aunque hay mucho que amar en la programación de juegos, el tedio te acosa a cada paso - buscando errores, o buscando errores en el código que te detienen el juego, es a menudo un proceso largo y penoso, así como lo es "el acabado final", o llevar al juego de una etapa beta tosca a completarlo. Mike también invierte mucho capital en sus proyectos contratando artistas gráficos y músicos.

Aunque Mike sea oficialmente un programador propietario, admite de buena gana que el desarrollo de software libre ha producido buenos productos, como Apache, el servidor que el usa para su sitio web, y se ha

beneficiado de compartir código con otros programadores indy. No obstante, el rechaza el desarrollo de software libre por miedo a que otros desarrolladores, menos escrupulosos lo usen para crear fácilmente imitaciones que competirían con sus propios juegos.

Aunque tuviera las licencias en orden, sería difícil. Ya tengo imitaciones. Tengo un juego llamado Z-Ball que alguien clonó. También hay gente que clonó mi sitio web. Han tomado el slogan de mi sitio web, "Donde la diversión nunca envejece", y lo cambiaron un poco para que dijera "Donde divertirse nunca envejece". También robaron el diseño y los gráficos.

Si un juego es un clon o no, o una legitima derivación es más complejo de lo que uno podría imaginarse. Por ejemplo, el juego de Mike Cosmo Bots obviamente toma muchas cosas del mas antiguo juego comercial Qix; hasta el juego es anunciado con la frase, "Los fans de Qix y Jezzball amarán a Cosmo Bots". Mike hace una diferencia entre lo que él hace y robar, una clase de clonación "sin vergüenza" en que otro desarrollador usa los mismo gráficos, diseño de niveles, historia, o música de otros juegos: "Si miran mis juegos y los juegos del género original, son tan diferentes, y tan vagamente basados en el concepto que no siento que lo que hago sea desvergonzado". De esta manera, la mayoría de los programadores, tanto libres o propietarios, reconocen que cierta libertad para tomar prestado es aceptable y muchas veces necesario.

El pecado capital de clonar es la falsa atribución, que ocurre cuando otro desarrollador trata de engañar a ingenuos usuarios haciéndoles creer que sus juegos en realidad fueron hechos por alguien más. El problema con la falsa atribución no solo es que el desarrollador original pueda perder algo de dinero en las ventas. Lo que es aún peor es que si el producto falso es de baja calidad, podría destruir o por lo menos dañar la reputación del desarrollador original. Una práctica similarmente vil es lo que Mike describe como "secuestro" o tratar de impulsar la venta de un juego dándole el mismo nombre de un juego ya establecido:

Muchas personas disfrutaron de Bugatron y trataron de crear un juego similar. Los juegos creados fueron tan malos que no importó. Estaría mas disgustado si la gente tomara el nombre. Podrían secuestrar mi juego. Si la gente nombrara a su juego Z-Ball y lograra ranking en Google, y si su juego fuera similar a mi Z-Ball, podría engañar a la gente comprando su juego en lugar del mío.

Lo que es más importante para Mike es proteger su buen nombre, el cual los clientes asocian con juegos de calidad y los lleva a probar sus nuevos juegos. La falsa atribución es también despreciada en la comunidad de software libre y se considera una seria violación a la ética de los hackers. Dudosamente, la reputación es mas importante y valorada en el desarrollo de software libre, ya que la mayor recompensa de un buen hack es el prestigio y el reconocimiento de otros hackers. Se espera que los hackers que "pinchan" un proyecto, o lo llevan en una nueva dirección a la de los desarrolladores actuales, le den un nuevo nombre, independientemente del acuerdo de licencia 8.

A diferencia de la mayoría de los programadores más importantes, Mike no está preocupado por la distribución no autorizada de las versiones completas de sus juegos en redes punto-a-punto o sitios web ilegales. El fundamento es simple: Las personas que quieren copiar sus juegos van a encontrar una manera de hacerlo. "El núcleo de audiencia que compra mis juegos no quieren tomar el riesgo de seguridad e ir a sitios warez. De cualquier manera esas personas nunca van a comprar mi juego". A la audiencia elegida por Mike - mayormente hombres y mujeres de 30 a 40 años - les falta el tiempo, la energía y el conocimiento para conseguir copias ilegales, y cuando el juego cuesta solo $20, buscar una copia no autorizada raramente parece valer la pena.

Mike ve el valor cultural y social de liberar el código fuente al público, y admite que le gustaría donar al público el código fuente de sus juegos más viejos cuando no den más ganancias. "Si no se está vendiendo

bien, ¿porque no?" pregunta Mike.

Matt Matthews es un hombre que ha estado haciendo esta pregunta a los desarrolladores un montón últimamente. Su sitio web, Liberated Games (Juegos Liberados), se dedica a ofrecer bajadas gratis y legales (muchas veces con código fuente) de juegos que han agotado su potencial comercial. Matt, también conocido como Curmudgeon Gamer (Jugador Cascarrabias), es un profesor de matemáticas que reconoce el valor educacional del código, y ha castigado a desarrolladores que entregaron su código binario sin el acompañamiento del código fuente: "La regla es: libre como la cerveza no es lo suficientemente bueno como para asegurar la supervivencia de un juego en el futuro; libre como la libertad, sin embargo, puede hacer que un juego viva por siempre". Matt señaló en una entrevista conmigo que los programadores pueden aprender todavía trucos invaluables estudiando el código fuente, hasta de códigos de juegos viejos de hardware antiguo:

A través del examen del despiece de estos programas viejos, los programadores aficionados no solo han aprendido de nuevo el arte de programar el Atari 2600, sino también han aprendido algunos trucos nuevos. Mientras estos programadores sean lo suficientemente inteligentes que puedan reinventar muchas a toda máquina, no hay dudas de que se beneficiaron de tener a su disposición una extensa biblioteca de código existente en la forma de roms de Atari 2600.

Matt argumenta que los ambientes de desarrollo severamente limitados como Atari 2600 incitaron a los programadores a producir código realmente innovador: "Saber como estas proezas fueron logradas es instructivo porque muestra lo que se puede lograr si se optimiza el código y se explota el hardware al máximo". Uno de los genios mas celebrados de todos los tiempos, Isaac Newton, dijo algo muy similar en 1975 "Si he visto mas allá que otros hombres, fue erguido en los hombros de gigantes". Matt ha hechos esfuerzos para contactar a los dueños de derechos de autor y preguntarles si considerarían lanzar sus juegos bajo una licencia pública para que los nuevos programadores puedan pararse en sus hombros. El proceso es largo y tedioso, pero al menos unos pocos de los principales editores parecen estar dando la bienvenida a la idea.

Una razón por la que Matt empezó Liberated Games (Juegos Liberados) fue su aversión por los sitios "Abandonware" (Nota: programas que generalmente siguen protegidos por derechos de autor, pero que no son ya vendidos ni patrocinados por sus dueños, pero lo suficientemente populares para seguir siendo compartidos ilegalmente). Sitios de Abandonware como BH Legends y Home of the Underdogs ofrecen bajadas gratis de juegos comerciales que se han sacado de producción. Home of the Underdogs (Casa de los Oprimidos) explica sus fundamentos en su pagina Acerca de:

Creemos que proveer juegos que fueron abandonados por sus editores, aunque sea técnicamente ilegal, es un servicio valioso para la comunidad de juegos, porque estos juegos están en peligro de desaparecer en la oscuridad, y los dueños de lo derechos de autor ya no los renuevan.

Aunque una discusión completa acerca de la ética del abandonware esta mas allá del alcance de este articulo (Me guardo la discusión para la próxima vez), recomiendo a cualquiera que se interese en el tema a leer el Scratchware Manifesto (Manifiesto de Scratchware) y consideren sus puntos. La opinión de Matt acerca del "abandonware" es que, aunque es noble y discutiblemente justificable hacer esfuerzos para preservar esos juegos, hacer que estén libremente disponible para ser bajados por el público no lo es.

El software libre vende, pero ¿Quién está comprando?

Espero que este artículo haya convencido a algunos desarrolladores de juegos constituidos y ambiciosos a

que consideren los beneficios de liberar su código fuente acompañados de sus binarios. Tales suplementos no costarían demasiado, y probablemente incrementarían y extenderían la innovación a través de toda la industria del software; después de todo, los video juegos llevan típicamente a la industria a la innovación. También me gustaría pensar que algunas personas podrían reconsiderar algunos de los otros principios del movimiento del software libre; en particular, el compromiso de enriquecer nuestras comunidades y dejar que la gratitud triunfe sobre la avaricia o la impotencia burocrática. Después de todo, el público que compra juegos es lo que hace posible la industria de los juegos propietarios; ¿porque no hacer algunas pequeñas concesiones? ¿Porque no "fertilizar el suelo" con código útil? Los beneficios a largo plazo son incalculables y completamente excitantes.

Muchos desarrolladores de juegos propietarios reconocieron los deseos creativos de incontables jugadores. Respondieron liberando kits de construcción y promoviendo el crecimiento de "comunidades de modificación" en Internet. La respuesta a estas concesiones ha sido enorme y ha obviamente prolongado de gran manera la rentabilidad de muchos juegos. El trabajo despampanante realizado por algunos jugadores ha rivalizado (y discutiblemente sobrepasado) al del desarrollador; tome Counter-Strike como el ejemplo de libro de texto. Valve vio la popularidad de un módulo creado por un fan y decidió lanzarlo como un juego independiente. Mientras reconoce el "valor agregado" de tal comunidad a las propiedades del juego, pocos desarrolladores parecen deseosos de tomar el siguiente paso lógico y liberar el código fuente. Contundentemente, "el código fuente gotea", eso significa que, la distribución no autorizada de código fuente en Internet, se está haciendo mas común. Parece que la industria puede responder a esta amenaza de una de estas dos maneras: Uno, aumentando la seguridad y gobernando con un puño de acero, o yendo con la corriente y cambiando sus paradigmas.

Consideramos brevemente que hubiera pasado si Valve software hubiera liberado su código fuente. Tengan en mente que no estoy hablando de sets de datos como gráficos, archivos de sonido, y similares, solo el "motor", o el código que junta todos estos sets de datos para hacer que el juego sea jugable. ¿Serían las consecuencias de liberar el código fatales a las ventas de juego? Dudosamente. Después de todo, relativamente pocas personas lo mirarían de todas maneras la gran mayoría estaría más que contenta con solo comprar el juego. Las personas que estuvieran interesadas y se beneficiarían por acceder al código son los programadores ambiciosos y conformados, que se beneficiarían mucho estudiándolo. El resultado seria el enriquecimiento de la comunidad de programación que llevaría rápidamente a una oleada de ideas radicales cuando los programadores dejaran de reinventar ruedas se enfocaran en lo que todavía no se ha hecho; innovación. En breve, proyectos "de código abierto" como Half-Life 2 probablemente conducirían a juegos mucho mejores, que resultarían en mejores ventas y usuarios finales mas felices. Por el argumento que liberar el código fuente haría al juego mas susceptible a las piratería; creo que Maquiavello, autor de El Príncipe, lo dijo mejor: "La mayor fortaleza que existe es evitar ser odiado por la gente". Substituyan "protección contra copia" por "fortaleza" en esta cita y tendrán una revelación abrumadora. La mejor protección contra copia es asegurarse que los jugadores respeten y valoren a la compañía. Cada persona que RIAA (Recording Industry Association of America, Asociación Americana de fabricadores de discos) o MPAA (Motion Picture Association of America, Asociacion Americana productores cinematográficos) demandan hace que el publico se amargue más y sea más compasivo hacia los piratas. Si se reusan a modificar sus estrategias o modelos económicos, a corto plazo, si se reusan a negociar con el público dudo muy poco que eventualmente sus imperios caerán.

Para tener éxito en la economía del software que esta cada vez mas dominada por el software libre y los desarrolladores de código abierto, los desarrolladores de software propietario necesitan hacer una seria introspección. La imagen pública es importante. La reputación es importante. En breve, lo importante va a ser convencer al público que lo que están ofreciendo es un trato justo.

Veo dos maneras en las que los desarrolladores de juegos pueden florecer en la economía post propietaria. Una es cambiando a un "modelo de servicios", que es esencialmente lo que es Valve´s Steam es. El futuro de

este modelo es claro: regalen el juego (con el código) y cobren por el acceso a servidores excelentes. Si estos servicios no pueden ofrecer soporte de calidad a un tasa razonable, no habrá una razón honesta para que alguien trate de engañar al sistema. Por la misma razón esas personas honestas se enojarían si escucharan que alguien robara electricidad o el cable, pondrían presión en otros para que obedecieran la ley y no estafaran. Es importante no ignorar esta presión; es el bien más valioso de una corporación. Por supuesto, lo que se perdería si la mayoría de los principales desarrolladores de juegos cambiaran a este sistema seria el juego "único" amado por tantos jugadores de PC. Habría muy poco incentivo financiero para producir un juego como el original Baldur's Gate, por ejemplo, desde que la "venta de bits secretos", como dice Eric Raymond, no seria viable en la economía post propietaria. Para ganar dinero, los desarrolladores de juegos tendrán que vender un "servicio de juego" que trabaje como la televisión por cable.

Otra estrategia es el modelo "patrocinado por el público'. Este modelo esta basado en el antiguo sistema de "patrón noble" de las primeras épocas, cuando las personas adineradas encomendaban grandes obras de arte publico a cambio de reconocimiento y un legado a su buen gusto. Siguen existiendo formas de este sistema, por ejemplo, el gobierno federal ofrece subsidios para financiar proyectos de investigación o similares que son altamente deseables, pero no inminentemente provechosos. El Fondo Nacional para las Artes es un programa federal que específicamente premia con donaciones a los artistas. Con la clase de presión adecuada y la representación, NEA podría premiar a los desarrolladores de juegos que pudieran demostrar la relevancia cultural y la importancia de sus proyectos. Por supuesto, otra estrategia seria apelar directamente al público en lugar de buscar el subsidio federal. Un desarrollador conocido y constituido como Bioware, por ejemplo, podría publicar una descripción detallada de un proyecto en su sitio Web y pedirle a la gente que contribuyera con los fondos necesarios para su producción. Esto probablemente llevaría a una clase de "juego regateo" con editores que discutirían la necesidad o el atractivo de ciertas características, la dirección del juego, y más. El código fuente probablemente se publicaría durante el proceso, y Bioware hará bien en escuchar y aceptar buenas contribuciones de programadores talentosos. Obviamente, Bioware escucharía con más atención a los editores quienes donaran la mayoría del dinero, pero el público tendría todavía bastante poder en tales desarrollos. Cuando el juego sea finalizado, los binarios serian liberados bajo una licencia pública para que todos pudieran disfrutarlo.

Concluiré este artículo con la observación que el futuro para el desarrollo de juegos de software libre luce bastante brillante. Internet ha hecho al viejo modelo de software propietario cada vez más vulnerable. La industria de la cultura como un todo será forzada a reconocer el poder de un sistema para compartir archivos barato, fácil, bien cifrado y anónimo que finalmente destruirá a cualquier corporación que no se comprometa o cambie. La industria del software esta excepcionalmente posicionada para asumir el liderazgo en la transición a una economía libre, donde el público es alentado a distribuir trabajos bajo su propio gasto y conveniencia. Mientras tanto, los desarrolladores de juegos aprenderán como proveer servicios útiles que el público comprará feliz. Si todo sale bien, ¿quien dudará que la industria cinematográfica y disquera seguirán el ejemplo? Esperamos con ansias por una era de más libertad y una sociedad más rica y mejor.

Notas

1 - Elegí el término "software libre" en lugar del mas común "código abierto" por una razón retórica. Por una buena razón de hacer esto, vean este ensayo.

2 - Los términos "código abierto" y "software libre" son elecciones retóricas que describen estrategias de desarrollo similares. La diferencia fundamental es que "software libre" acarrea con una cierta ideología que todo el software debiera ser libre (como libertad de expresión), mientras que "código abierto" es mas neutral y discutiblemente mas atractivo para los negocios. En este artículo, use "código abierto" y "software libre" basado

en el término que el desarrollador en cuestión prefiera. Estoy muy tentado de solo usar el termino "software libre" y terminar con esto, porque lo que importa es si el código está disponible o no y que puedan hacer legalmente los usuarios finales con él.

3 - Para mas información, vea La postrema historia de los video juegos de Steven L. Kent's

4 - La información de esta sección viene de las páginas históricas de Unix de Bell Labs.

5 - Vea este artículo en CNET.com.

6 - La metáfora del Catolicismo versus el protestantismo pude ser muy útil para entender el Software libre versus el propietario. Podríamos pensar a Microsoft como la iglesia católica y a Bill Gates como el Papa. Ellos sancionan todas las interpretaciones de su Biblia, por ejemplo, el código base, cobrar diezmos, e intentar luchar contra la herejía con el recurso de la ley. Mientras tanto, podemos pensar al Movimiento de Software Libre como el protestantismo y a Richard Stallman (o a Linus Torvalds, elijan) como Martin Luther. Ahí, la idea es interpretar el código por uno mismo, siendo esta la tarea de todos los usuarios finales de tomar un rol mas activo en entender y aplicar las enseñanzas de la Biblia.

7 - King, Brad y John Borland. Dungeons and Dreamers. (Calabozos y soñadores) Emeryville, CA: McGraw Hill, p. 131.

8 - Para mas información, vea The Cathedral and the Bazaar (La catedral y el bazar) de Eric Raymond.
