Soy Adrián, apasionado de la programación y los videojuegos, actualmente trabajando como Lead Programmer en un proyecto Triple A. Recientemente inicié un canal en YouTube donde trato de divulgar sobre temas interesantes de la industria del videojuego.

EL ÉXITO DE UN VIDEOJUEGO, ¿A QUÉ SE DEBE?

Existen muchos factores para que un videojuego tenga el éxito que sus desarrolladores tanto desean. Un arte espectacular, una música atractiva, una historia absorbente, unas mecánicas divertidas, un marketing efectivo, una producción eficiente, una traducción bien lograda. Pero ¿hace falta algo más? ¿El código? ¿Qué pasa con el código? ¿Debe el código ser bonito o de calidad?

Hay quien piensa que el código da igual, mientras funcione. Bueno, vamos a verlo.

¿QUÉ ES EL CÓDIGO?

Lo primero que tenemos que entender es, ¿qué es el código para un videojuego? ¿qué función cumple? El código se encarga de aportar la interactividad con el usuario, así como la comunicación entre los elementos del juego. No hay ningún otro elemento del videojuego que haga eso, todo es estático, excepto el código, el único encargado de reaccionar a los estímulos que el usuario envía a través de los periféricos (teclado, ratón, gamepad, etc.). Bien.

EL CÓDIGO ES COMO UNA FÁBRICA

Imaginen que nuestro código, es como una nave industrial, donde se reciben pedidos (inputs del usuario). Imaginemos a PackageX, una innovadora startup sobre gestión de pedidos, tienen mesas de ping pong, horario flexible, buen rollo, seguro médico, fruta gratis, gimnasio, son jóvenes, dinámicos, y no, no podéis mandar vuestro CV, recordad que es ficticia.

En la fábrica las personas que allí trabajan procesan los pedidos y activan las palancas necesarias para que se produzca el resultado que el usuario pide. Supongamos un caso de uso: el usuario pulsa el botón de atacar, ahora el pedido es recibido por nuestro personal (el código). ¿Quién gestiona el pedido? ¿Una o varias personas? ¿Cualquiera puede encargarse? En una nuestra fábrica bien organizada, el pedido de ataque sería gestionado por una persona especializada en peticiones de ataque.

Esa persona tiene todas las herramientas necesarias y conoce a qué lugares de la fábrica debe dirigirse para activar las palancas correctas, en el menor tiempo posible, gastando el menor número de recursos posibles. Sin colisionar con el resto de las personas que allí trabajan y gestionando los errores o problemas que puedan surgir de forma inteligente.

Fantástico ¿verdad?, pero es bien sabido que no todas las fábricas funcionan forma tan buena, existen malas organizaciones, malas asignaciones de tareas y de recursos, jefes que no saben delegar en otras personas o personas que hacen más de lo que deberían. Pues bien, siguiendo con nuestro ejemplo, pongámonos ahora en una situación mucho peor. Os presento a Gestión de Paquetería Puerto Hurraco. Esta empresa está compuesta solo por 3 personas: una directora, su ayudante (quien resulta ser su hermano y no da pie con bola) y un becario.

El pobre becario es el encargado de recibir todos los pedidos y avisar a la directora para empezar el proceso, cuando la directora es avisada, esta avisa a su hermano, para que gestione el pedido, el cual gasta un más recursos de los necesarios para procesar un pedido, el cual no queda procesado del todo porque tiene que a avisar a la directora para deshacer parte del trabajo y volver a procesarlo, para poder cerrar el pedido correctamente. Mientras todo eso ocurre el becario envía una carta al usuario donde le dice:

«Por favor espere, no apague la consola. Estamos procesando su pedido: Cargando...1%»

A estas alturas los lectores que se dediquen a la programación ya deberían haber entendido el símil. Pero vamos a seguir un poco más allá con el ejemplo, para meter a todos en contexto y porque me está quedando bastante gracioso.

Tras meses en funcionamiento, PackageX detecta algunos fallos en su proceso de gestión de pedidos. Vaya, un fallo lo tiene cualquiera. Así que deciden revisar su proceso y cambiar o mejorar aquellos puntos en los que el proceso flaquea. Y vaya si lo consiguen. Dado que su proceso está bien definido, cada persona tiene unas responsabilidades bien delimitadas y la comunicación es buena, PackageX solo ha tenido que aplicar un pequeño esfuerzo a mover algunas cosas de sitio y renombrar algunas secciones. Y a seguir facturando millones.

Mientras tanto, Gestión de Paquetería Puerto Hurraco, también ha decidido mejorar su proceso. O intentarlo. Se reúnen la directora y su hermano, mientras al becario le han dado un par de horas libres para que vaya a la cantina. Tras esas dos horas deliberando, llegan a la conclusión de que necesitan contratar a 25 personas más para dividir correctamente las responsabilidades, despedir al hermano y ascender al becario a gestor líder. Tras ello, la directora prefiere cerrar la empresa y con todo el capital ahorrado volver a fundar la empresa desde cero. Te echaremos de menos Gestión de Paquetería Puerto Hurraco.

LA INDUSTRIA DE LOS VIDEOJUEGOS Y LA CALIDAD

Durante mi tiempo como programador, he visto código y he leído y escuchado opiniones de muchos programadores/as, y aún me sorprende como, en muchas empresas desde las más altas capas de dirección hasta la capa de programación, aún hay gente que no se toma en serio la calidad del código.

Imagina ser propietario/a de una empresa de videojuegos y descubrir que el código de tu juego funciona como Gestión de Paquetería Puerto Hurraco, ¿asusta cierto? Un buen código ahorra tiempo en bugs, y el tiempo es dinero, ¿verdad? Pues tal vez deberías tener una reunión con tu equipo de programación para asegurarte de que eso no ocurre.

En la industria de los videojuegos ocurre algo que nunca puedo etiquetar como bueno o malo, simplemente ocurre. Y es, que los desarrolladores llegan a esta industria bajo la premisa de la pasión que tienen por los videojuegos, y eso es fantástico en primera instancia. Cada persona que trabaja y siente pasión por lo que hace, lo hace más feliz y mejor, y eso es increíble. La pasión es lo que te hace avanzar y mejorar.

Pero, ahora bien, tras hablar con muchos desarrolladores, algunos más grandes, otros más indies, he podido comprobar que existe una falta de técnica y metodología muy grande. Existe una falta de gusto por la calidad y las cosas bien hechas, no solo hechas con pasión, sino también con técnica, y ahí es a donde voy.

He estado en empresas de software muy diferentes, desde Apps para Android, Desarrollo Web y SEO, Realidad Virtual, Videojuegos de Casino y Videojuegos, digamos canónicos (?). Y si tengo que comparar, diría que cuanto más alejada estaba la empresa de los Videojuegos (canónicos), mejor era el proceso de desarrollo, era más estable. Parece que ahí las buenas prácticas sí tenían cabida.

He visto y aplicado muchas buenas prácticas en el desarrollo del software: clean code, patrones de diseño, TDD, técnicas para hacer un buen comentado del código, refactoring, code reviews, control de versiones, y temas más transversales como metodologías ágiles, scrum, kanban, etc. Y con diferencia, en la programación de videojuegos es donde he visto que menos buenas prácticas se aplican, y menos se persigue la calidad, solo sirve el hecho de que funcione.

¿POR QUÉ?

Bueno, esto se puede deber a muchos factores, los cuales trataré de plasmar y analizar según mi observación del ecosistema de la programación de videojuegos.

Primero, la calidad del código no importa, siempre que funcione lo suficiente, como para ejecutar el juego de principio a fin. Es simple. Y en general, en el desarrollo del software se ve como un ahorro de tiempo y dinero. Más adelante demostraremos que no es así.

Segundo, el personal que trabaja programando en la industria de los videojuegos, por lo general (insisto), no entran a la industria por su pasión por programar, ni por programar videojuegos, sino, por los videojuegos en sí mismos, la diversión que nos ofrecen, la emoción de sus historias, sus mecánicas, compartirlo con amigos/as, los buenos momentos. Entiéndase, que amo los videojuegos, pero también me siento en la responsabilidad de fabricar un software de calidad y fiable, y eso también me apasiona.

Tercero, el hecho de que un videojuego es software parece ser invisible a ojos de la industria. Es un videojuego, correcto, pero software al fin y al cabo, y como tal, lleva un proceso de desarrollo, al cual se le pueden aplicar muchas buenas prácticas, que facilitan mucho más, bueno… todo.

Cuarto, los tiempos de entrega y los crunches. Dos factores que nunca ayudan. La industria de los videojuegos es muy competitiva, por tanto, los tiempos de entrega son cada vez más ajustados, siempre tienes que lanzar el producto antes que tu competidor, aludiendo al primer punto, nos da igual como salga, solo tiene que salir. En cuanto a los crunches, no es algo que quiera comentar en este artículo, pero todos sabemos lo perjudiciales que son para el equipo. Solo diré que se han estado asumiendo como una estrategia en la industria, cuando realmente son una desgracia dentro de la misma y un fallo de gestión.

Quinto, para aplicar esas buenas prácticas, primero hay que estudiarlas y aprenderlas. Por lo general, quien tiene tiempo de estudiarlas, no tiene ganas, y quien tiene ganas, no tiene tiempo.

Sexto, metodologías como Scrum (por poner un ejemplo), no son apropiadas para videojuegos. En realidad, sí, Scrum, puede aplicarse en empresas que desarrollen software, y un videojuego, resulta ser software. Nadie aplica Scrum puro, nadie fuerza a la estructura de su empresa a encajar en Scrum, simplemente se coge de Scrum lo que pueda servir y aportar valor a tu proceso de desarrollo, y con eso basta a priori. Hay empresas que lo reducen tanto que al final solo es un tablero Kanban en Trello, sigue siendo apañado.

Séptimo, bastante relacionado con el primero, el desarrollo de software no se quiere entender desde las capas superiores de las empresas. Todo el mundo entiende el arte, y sabe cómo lo quiere, y sabe opinar, etc. Todo el mundo sabe escuchar el audio del juego y saber si lo quiere así o lo quiere cambiar. Todo el mundo sabe ver la campaña de marketing y decidir si quiere cambiar la estrategia o no. Pero la programación sigue siendo la gran caja negra de todas las empresas. Nadie sabe lo que quiere. Y eso provoca que no exista un control de calidad externo al que realiza el equipo de programación. Y si no lo realizan ellos, no lo realiza nadie. Porque nadie más lo entiende claro.

En resumen, estas buenas prácticas son aburridas, nadie quiere aprenderlas, estudiar, romperse la cabeza, pero entonces, tienes que estar dispuesto a alargar tu desarrollo un tiempo que no tenías previsto, debido a la deuda técnica que vas a ir arrastrando, para compensar esas malas prácticas.

Y eso me plantea otra pregunta ¿Tienes el dinero necesario para alargar el desarrollo ese tiempo imprevisto?, siendo propietario de esa empresa ¿no preferirías que las cosas se hiciesen bien para ahorrar dinero? Como veis la premisa es absurdamente simple y primitiva: “Si inviertes en calidad en el software, ahorrarás dinero.”

Un hecho es que el código bien desarrollado, mediante buenas prácticas, estructurado, modular, con los patrones de diseño correctamente aplicados, testeado, asegurando su correcta construcción y funcionamiento, ahorrará un tiempo significativo en el futuro. Ya que arreglar un bug en un código bien mantenido, necesita X horas, mientras que arreglar un bug en un código mal mantenido, necesita 10X horas, si no más.

Un videojuego puede retrasarse por muchos y muy variopintos motivos, pero si reducimos ese riesgo desde el código estaremos asegurando una gran parte del porcentaje de éxito de nuestro juego.

Creación de videojuegos para todas las edades y niveles

Orígenes de la programación de videojuegos.

Programar videojuegos ha sido, desde finales de los setenta, una actividad a la que se han ido dedicando un grupo cada vez mayor de entusiastas. Muchos de ellos aficionados, que luego pasaron a profesionalizarse.

A finales de los setenta y principios de los ochenta las librerías se inundaron de revistas dedicadas a los videojuegos, pero muchas también enseñaban los aspectos básicos de creación de videojuegos.

Algunas directamente solo trataban el tema de la creación de videojuegos, usando máquinas tan emblemáticas como el Sinclair ZX Spectrum o el Spectravideo 318-328, o los posteriores MSX, sin olvidar las máquinas de Atari, Commodore... La lista es muy larga. Muchos copiábamos directamente el código que se publicaba en las revistas, sin tener ni idea de qué era aquello. A veces funcionaba, y a veces no. Pero era apasionante.

Las cosas han cambiado, y mucho. Hoy día aquellos juegos siguen siendo mitos, pero técnicamente la programación ha avanzado y es irreconocible con respecto a aquella época. Actualmente existen herramientas para hacer videojuegos, porque programarlos desde cero sería una locura. Todas las empresas usan herramientas muy sofisticadas.

La llegada de Unity y Unreal

Existen dos herramientas muy populares que tienen versiones gratuitas, y que se pueden descargar directamente para comenzar a usar de forma inmediata, mediante un simple registro, sin costes asociados. Una de ellas es Unreal 4, herramienta muy potente, con dos elementos básicos: los “blueprint” para crear juegos mediante un sistema visual muy potente, aunque con ciertas limitaciones, y el lenguaje de programación integrado, que es el archiconocido C++.

La segunda herramienta es tremendamente popular, especialmente desde que su versión gratuita se abrió a todo desarrollador: se llama Unity, antes conocido como Unity 3D.

¿A quién quiero más, a mamá o a papá?

Y ahora viene la terrible pregunta: ¿cuál es mejor? Las respuestas que se leen por ahí están muchas veces llenas de tópicos. Podemos dejar clara ya una cosa: ambos motores van a permitir crear grandes juegos. Unity en concreto, que suele tener fama de ser peor que Unreal, tiene esa fama inmerecida. La primera: “Unity se usa para crear juegos que son clones de prototipos o demos”, lo cual hace pensar que con Unity no se pueden crear grandes juegos. No es así; que un juego sea básico no significa que Unity sea básico. Ese juego básico se podría haber creado también con Unreal.

La segunda razón por la que Unity ha tenido una fama de estar en segundo lugar es que, técnicamente, ciertas tareas eran más complejas de hacer que en Unreal. Es cierto que era así en algunos aspectos, pero aquí hay algo muy importante que decir: Unity ha cambiado de forma impresionante en estos tres últimos años. De la versión 4 se pasó a la 5, que ya añadió muchos elementos innovadores. Pero es que la versión 2019.2, que es la actual cuando se escribe esto, desafía a cualquier otra herramienta disponible como Unreal.

Unity además emplea el lenguaje C#, más cercano para muchos programadores que se inician, y que realmente da menos sustos que el C++. Actualmente Unity compila el código C# transformándolo en C++, con las ventajas que ello conlleva. Sí, se pierde algo de rendimiento, pero para muchos casos es más que suficiente. Y, por cierto, el Javascript ha sido descatalogado.

La potencia de Unity.

En el campo del prototipado y desarrollo rápido, Unity se ha convertido en una herramienta tremendamente poderosa. En la actualidad todo el proceso de desarrollo, pruebas y puesta a punto de cualquier juego se efectúa a una velocidad muy alta. Una vez se conocen los aspectos clave de Unity, se puede comenzar a trabajar en proyectos que sí, serán sencillos, no serán triples AAA, pero que tienen un aspecto y una calidad muy altas, si el programador pone su empeño en ello claro.

No es oro todo lo que reluce.

Porque aquí está el quid de la cuestión. Unity es una herramienta fantástica, pero programar videojuegos sigue requiriendo un esfuerzo tremendo, que desgasta muchísimo, siendo una actividad que consume horas y horas de sueño, hasta tener un juego con la jugabilidad y la calidad suficientes como para atraer la atención de los jugadores.

Pensemos en la gigantesca cantidad de juegos que se publican en Steam cada día. Muchos de ellos son realmente malos, algunos no están mal y a veces aparece una joya. Pero esa joya queda enterrada entre todos los desarrollos básicos, y prácticamente todos están hechos con Unity. Pero hay que insistir en la idea: Unity no es la responsable de eso. En la Wikipedia se pueden ver juegos desarrollados con Unity de alta calidad, son muchos y para todos los gustos.

Entonces, ¿me quedo con Unity?

Si eres un desarrollador que empieza, definitivamente Unity va a darte muchas ventajas para empezar a dar los primeros pasos. Sí, Unreal tiene Blueprint, pero luego se pasará al C++, con lo que se pierde esa eficacia. Por lo tanto, Unity es mejor opción.

Si estás en un equipo de tres, cuatro personas, con experiencia ya demostrada, la decisión es ya más personal que otra cosa. Lo que sí se debe aclarar es que ambas herramientas aportarán potencia de sobra para grandes títulos.

Y si es un equipo profesional, entonces la cuestión permanece, pero también te puedes plantear otros motores, que, eso sí, tienen precios bastante o muy elevados. Pero, en general, los “indies” funcionarán bien con ambas herramientas.

Pero hay que insistir en un punto: Unity 2019 es una gran herramienta con muchas mejoras incluidas en las últimas versiones, con un lenguaje potente, y que incorpora por defecto el IDE Visual Studio, en versión Windows y ahora también en Mac.

Si vas a empezar en esto, claramente recomiendo Unity. Una herramienta potente y con la que enseguida se verán los primeros resultados. Eso sí: primeros resultados. De ahí a un título de calidad hay que trabajarlo mucho. En Unity o en cualquier herramienta.

Final en página 30