

**Documentación del Producto**

**Resumen**

Este documento resume el trabajo realizado para producir el juego “Vengeance”. Se cubren todos los aspectos relacionados con el desarrollo del mismo, explicando quienes trabajaron en el mismo, como fue la construcción de código, y las diferentes herramientas utilizadas por el grupo para organizar y facilitar la construcción del mismo, además de incluir como fue la experiencia de trabajo en el proyecto.

El documento se divide de la siguiente manera:

\*Presentación del juego y el equipo de trabajo  
\*Conceptos clave   
\*Estructura del Sistema  
\*Herramientas de Producción  
\*Herramientas auxiliares/ de comunicación  
\*Optimización e inconvenientes  
\*Conclusión  
\*Elementos adicionales  
\*Referencias  
\*Agradecimientos

**Presentación del Juego**

Vengeance es un RPG (*Role-Playing Game)*, un juego donde el jugador adopta un rol de un personaje en una historia, usualmente determinado “PJ”. En este caso, el jugador toma el rol de Gudi, quien debe recorrer los distintos mapas buscando mejorar su experiencia de combate (subiendo de nivel) para poder pelear contra el villano Jasaskar.

Lo que buscamos con este juego fue crear un pequeño juego RPG que sea disfrutable por quien lo jugué, y que al mismo tiempo represente un reto, y mantenga al jugador interesado en completarlo, buscando ser cuidadosos de que el juego no este desbalanceado y se vuelva tedioso o demasiado fácil, para que nosotros mismos podamos experimentar la situación que viven aquellos desarrolladores de juegos que nosotros solemos jugar hoy en día.

**Equipo de Trabajo**

**Martín Bombara:** Fue el encargado principal de buscar los elementos gráficos y sonoros que implementaríamos al juego, además de aportar su punto de vista en la documentación, además de realizar la IA de los enemigos del juego

**Guido Contento:** Realizó la documentación del juego, además de aportar algunas funciones puntuales del código del juego, y encargarse de mantener actualizada la tabla con las actividades que había que realizar.

**Stefano D’Alessandro:** Se encargó principalmente de las clases que conforman al juego, además de implementarlas al proyecto final, y de realizar los diferentes comportamientos de los elementos del juego (pathing de enemigos, warps, etc)

**Ignacio Montaña:** Se encargó de los aspectos visuales del juego, esto incluye graficas del menú y de los diferentes personajes que se ven el juego. También realizó las colisiones de los mapas del juego, y aporta su punto de vista en la documentación.

**Conceptos Clave**

**RPG  
Kanban  
Scrum  
Backlot  
Balance  
Programación Funcional  
Programación Orientada a Objetos  
Pathing  
Warp  
Controlador de Versión**

**Estructura del Sistema**

Luego de investigar acerca de Python, y de otros juegos que habían sido realizados en este mismo lenguaje, optamos por utilizar una metodología funcional para la programación del juego, implementando al mismo tiempo programación orientada a objetos en los casos que encontraramos más óptimos.

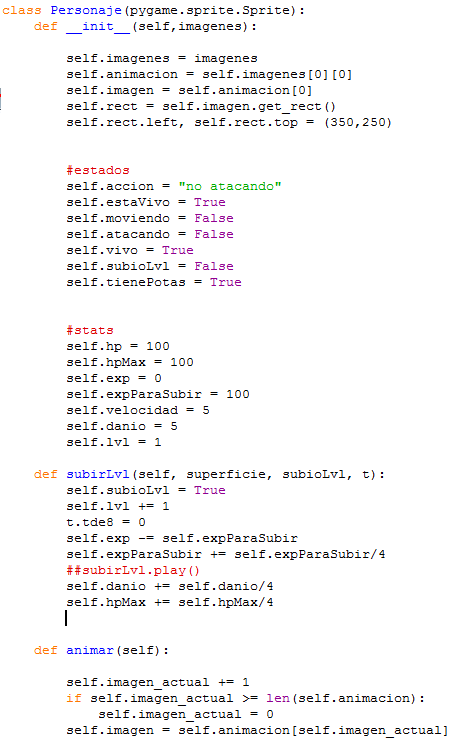
El juego consta de diferentes funciones que son llamadas constantemente cuando debe ocurrir algo en el juego (cuando empieza, cuando se ataca, cuando se quiere cambiar de mapa), esto es lo denominado programación funcional.

Al mismo tiempo, ciertos elementos del juego (como el personaje principal, los enemigos, ataques, botones) están encapsulados en clases. Las clases se dividen en 3 elementos: Un nombre, una serie de atributos, y un conjunto de métodos. Los atributos son las características que poseen todos los objetos de la clase, y los métodos, aquellas cosas que saben realizar esos objetos. Este tipo de programación es denominada “orientada a objetos”.

La sintaxis del lenguaje Python fue la que nos permitió trabajar de esta manera, ya que es el único lenguaje (al menos, en nuestro conocimiento) que permite trabajar simultáneamente con dos paradigmas diferentes. Esto llevó a que la programación del producto se facilite bastante teniendo en cuenta el tiempo para la misma.

A continuación, presentamos algunas extracciones del código de nuestro proyecto (Cabe aclarar que los códigos no están completos por razones de espacio, sino que con estos buscamos explicar lo mencionado previamente):

**Programación Orientada a objetos: Clase personaje**



En este caso tenemos un extracto de la clase Personaje, que representa a nuestro personaje principal, es decir, aquel que maneja el usuario. Podemos notar las características explicadas previamente de la programación orientada a objetos: precedidos de ***SELF***están algunos de los **atributos** que tiene la clase Personaje, y precedidos de **DEF**, los diferentes **métodos** pertenecientes a esta (en este caso observamos la clase que sube de nivel al personaje y la que ejecuta las animaciones de caminata del mismo).

**Programación Funcional: Funciones Main y Menú**



En este caso podemos ver un ejemplo de la aplicación de programación funcional, donde tenemos dos funciones. La lectura de código comienza desde la función **Menú**, que ejecuta el menú principal que tiene armado nuestro juego, con las diferentes opciones (Jugar, Ver los controles o Salir). Cuando el botón jugar sea pulsado, se pasará a la función **Main** que es la maneja la jugabilidad principal del software. Esta es la modalidad principal utilizada en el juego: llamar a diferentes funciones que realicen procesos cuando el jugador haga algo determinado, y luego se vuelva a la función Main.

**Herramientas de Producción**

A continuación, se listan las herramientas básicas que requerimos para llevar a cabo el proyecto:

**Python 2.7:** Versión del Lenguaje Python utilizada. Decidimos utilizar esta versión ya que era la recomendada por los tutoriales que utilizamos para aprender el código, porque era la más compatible con Pygame y de esta forma minimizábamos el riesgo de errores.

**Pygame 1.9.2:** Librería que contenía las herramientas para la realización del juego. Nuevamente optamos por utilizar esta versión por ser la utilizada durante el aprendizaje de la misma, además de poseer una extensa documentación (en la página de descarga) para esta versión que nos facilitó bastante el manejo de la librería.

**GitHub Desktop:** Cliente Git que utilizamos para realizar la actualización del repositorio hosteado en GitHub. Optamos por utilizar este ya que encontramos que era fácil de usar y nos permitía rápidamente actualizar el trabajo que realicemos.

**Eclipse Java Oxygen.1A:** IDE que utilizamos para producir el código del juego. Optamos por este ya que, además de ya haberlo utilizado como herramienta para otros trabajos, una de sus funcionalidades es la completa compatibilidad con Python y sus librerías. El único requisito fue configurar el compilador, lo que nos ahorró varios minutos a la hora de probar el código.

**Herramientas de Comunicación/Auxiliares**

En este caso, listamos aquellos programas que utilizamos para facilitar el trabajo, y al mismo tiempo, resolver dudas rápidamente en caso de que no nos encontremos trabajando físicamente juntos:

**Skype:** Programa que utilizamos para comunicarnos en caso de que no podamos realizar reuniones, principalmente para aclarar alguna duda (ya que podemos ver la pantalla de otro que se encuentre en la llamada).

**Trello:** Aplicación Web que nos permitió simular una tabla Kanban con su backlot (lista de tareas) correspondiente, que utilizamos para aplicar esta metodología ágil a nuestro proyecto, lo que nos permitió trabajar de forma más organizada.

**Optimización de código e inconvenientes**

A medida que íbamos produciendo el juego, nos topábamos con pequeños inconvenientes, que aunque molestaban al principio, estos siempre lograban aportar para mejorar nuestra experiencia en la construcción del software.

Un problema bastante común fue el hecho de que no teníamos mucha experiencia programando en Python ni la librería Pygame, y dado que era la primera vez que nos enfrentábamos a un proyecto así, a veces pasábamos mucho tiempo para solucionar pequeños errores que luego entendíamos que era cuestión de poner un poco más de atención a lo que escribíamos. En todas estas situaciones logramos encontrar la solución y seguir avanzando con el proyecto.

El problema principal fue que, dado que asignamos diferentes partes del proyecto a cada miembro del grupo, a la hora de unir las diferentes clases y funciones, esto nos obligaba a tener que juntarnos para arreglar la estructura del código y que este compile. La utilización del repositorio en GitHub y de Scrum como metodología ágil nos facilitó bastante estos problemas ya que hizo que estos problemas puedan resolverle en lapsos de aproximadamente 1 hora, para que luego podamos enfocarnos en las siguientes tareas pendientes del proyecto.

Otro inconveniente bastante común fue que usábamos diferentes computadoras para el trabajo, todas ellas con recursos diferentes (tanto las de escritorio, como las notebooks) lo que produjo que en algunas de las computadoras que utilizábamos, el juego tenga problemas de FPS, que complicaban la jugabilidad y volvían tediosa la experiencia de juego, lo que nos obligó a buscar escribir siempre el código más óptimo y con menos consumo de recursos posible, usando como referencia la computadora con menos capacidad de todas las que teníamos, con el objetivo de que nuestro juego pueda correrse en cualquier computadora usada actualmente, sin generar problemas.

**Conclusión**

La construcción del producto no resultó complicada como nosotros esperábamos. Logramos construir un juego que consideramos es jugable y aporta un pequeño tiempo de diversión a quien lo juegue.

Aprender y aplicar las metodologías ágiles (principalmente Kanban y Scrum) nos ayudó bastante ya que hizo que en todo momento tengamos en cuenta que trabajo había que realizar, de esta forma, si terminábamos algo antes de lo esperado, podíamos seguir rápidamente con lo que seguía, sin tener que andar debatiendo en grupo con que parte del proyecto continuar.

Utilizar un repositorio y un controlador de versiones también facilitó mucho el trabajo del grupo, ya que en todo momento teníamos una versión actualizada del proyecto, y podíamos agregar los cambios que realizamos en cualquier momento y en cualquier lugar, y el sistema permitía mantener a los demás miembros al tanto de estos cambios.

Concluimos que este proyecto no solo nos permitió comprender y experimentar la utilización de las principales metodologías de producción de software (y sus complementos), sino que al mismo tiempo aprendimos un lenguaje y técnicas de programación que podremos reutilizar a futuro en caso de que no nos propongamos personalmente un proyecto de esta índole.

**Elementos adicionales**

En este apartado explicaremos algunas de las funcionalidades que planeamos para nuestro proyecto, pero que por razones (principalmente la falta de tiempo) no llegamos a implementar:

**Sistema de Tiendas:** Durante las primeras reuniones planeamos un sistema de tiendas donde el jugador pudiera intercambiar oro soltado por sus enemigos, y de esta forma comprar mejoras para sus habilidades y atributos.

**Sistema de Objetos:** Ideamos un sistema de objetos para que el jugador mejore sus atributos. Estos serían obtenibles de forma aleatoria por los enemigos, o intercambiables por el oro soltado por los enemigos del juego llegada una determinada parte del juego.

**Guardado de Partida:** Otra función que pensamos fue que el jugador pueda guardar todos los datos de su partida, y cuando desee volver a jugar pueda seguir la partida que ya estaba jugando o empezar una totalmente nueva. No pudimos concretar esta función debido a problemas de código a la hora de cargar la partida, por lo que decidimos descartarla.

**Referencias**

**Página de descarga de Python:** <https://www.python.org/>

**Página de descarga de librería Pygame:** <https://www.pygame.org/>

**Documentación librería Pygame:** https://www.pygame.org/docs/ref/pygame.html

**Mapas realizados con Tiled Map Editor:** <http://www.mapeditor.org/>

Música y efectos de sonido obtenidos de:

**VGMusic:** <https://www.vgmusic.com/>

**YouTube:** <https://www.youtube.com/>

**Sprites obtenidos de:** <http://spritedatabase.net/>

**Agradecimientos**

Queremos agradecer principalmente a ChelinTutorials (<https://www.youtube.com/channel/UCZJS-lpC1BhLSdsjAqj1i8A>) por enseñarnos el lenguaje Python y a utilizar la librería Pygame, a las páginas mencionadas en las referencias por los elementos que utilizamos en el juego (ya que son de libre uso) y a la página Stack Overflow (<https://stackoverflow.com/>) por resolver las dudas que nos surgían a la hora de codificar.