Hernández Monroy Luis Fernando

*Documento de diseño.*

Este es un proyecto que momentáneamente no cuenta con un nombre estable más que uno clave para referirse a él, este juego está clasificado de acuerdo al sistema ESRB (Entertainment Software Rating Board), E+10 para mayor comprensión de los temas a tratar dentro del mismo.  
Tiene la finalidad de poner en duda el estereotipo del héroe que es utilizado como la “leyenda” que contiene un destino a cumplir, por lo que se utiliza la sátira dentro del juego, con fin de dejar el mensaje que cualquiera puede ser el siguiente heroe, o aplicado a la vida, cualquiera puede ser el gran Diseñador que se propone, cocinero, profesor, dando un rango de interpretación abierto a cualquier nivel de comprensión a partir de 10 años, se pretende que la interpretación sea distinta entre rangos de edad.

*One Shot*

El juego te pone en el papel de un herrero promedio trabajador situada en un mundo fantasioso, donde eres encargado de hacer armas para caballeros de toda clase, se te es encargado de una espada para conmemorar la última victoria del reino donde su gran héroe de guerra “*Eroj*”, será reconocido como un caballero noble y se le entregará una con enchapado de oro, su nombre labrado y brillantes, como reconocimiento, al terminar la y entregar la, hay una invasión inesperada al momento de la ceremonia, desde las profundidades salen cinco entes que parecen ser demonios, y su líder *Geist* , el héroe intenta atacar lo de inmediato con su nueva espada, pero este muere al ser impactado por un hechizo que lo pulveriza y su espada se funde, por lo que queda inutilizable.

Al ver esto, el Rey del reino declara rendición del reino y la instauración y cumplimiento de las nuevas leyes establecidas por los Entes.   
Las semanas pasan y el pueblo deja de tener un flujo económico, ya que los entes han instaurado que los productos del reino se quedarían dentro del mismo para ser consumido por los mismos, así racionan el consumo de los habitantes y se comercia con poca gente extranjera, tu personaje deja de tener clientela ya que se ha prohibido el uso de armas por el hecho que ellos cuentan con magia negra, y son capaces de defender su nuevo territorio sin el uso de armas ambiguas, y empiezas a preguntarte tu como herrero que podrías hacer, si aun tienes materia prima sobrante.

Haces un par de herramientas, un arco y suficientes flechas, y decides acudir al rescate del reino, que nadie solicitó, pero sabes que es requerido, así que con su martillo y arco que pueden ser mejorados encontrando materia prima y crear otras herramientas con utilidad, acudir a los cinco puestos de batalla que rodean la ciudad a manera de muralla, derrotar los, donde tendrás la oportunidad de tener un diálogo con ellos para descubrir su motivación y cambiar el rumbo de la historia, afrontar las consecuencias de ser un héroe principiante y ayudar a los aldeanos que puedas en tu camino claro está.

*Especificaciones Técnicas e interactivas*

Single player.

Juego 2D, tercera persona, perspectivas isométricas, caballero y 3/4.

Diseño: Dibujo digital toon, mixtas proporciones.

Lenguaje de programación: C#

Hardware: Inicialmente ordenadores, portátiles y de escritorio.

Compatibilidad: Soporte de Control(Óptima jugabilidad) y teclado.

Software: Unity 2018, Krita, Garageband.

ESRB +10

Venta completa del juego, dependiendo su éxito puede haber actualizaciones gratuitas para avivar la comunidad.

*Planteamiento de problemas;*

1.- Sistema de vida.

2.- Sistema general de navegación.

3.- Mitigación daño (esquive) y Ataque.

4.- Recibir daño.

5.- Reconocimiento de ítems.

6-. Sistema de inventario.

7.- Creación de equipamiento.

8.- Mejoras en base al equipamiento.

9.- Interacción con NPC.

10.- Sistema de diálogos con NPC.

11.- Sistema de combate.

12.- Daño a los enemigos.

13.- Daño recibido.

14.- Eliminar objetos destructibles y enemigos.

15.- Sistema de recolección de recursos para construcción.

16.- Inventario desplegable.

17.- Mapa desplegable.

18.- Decisiones en el juego.

19.- Múltiples finales.

20.- Trampas dentro de dungeons.

21.- Boss multifacetas.

22.- Pausa.

23.- Pantalla de muerte.

24.- Menú principal.

24.- Menú con distintos slots de carga de partida.

25.- Detección de teclado o control.

*Sistema de vida*

El usuario necesita una barra de vida que represente el valor igual a su vida, su mayor vida posible es 100 como límite, sin embargo con las upgrades que creas en el futuro, este valor se puede multiplicar para que aumente o reduzca, sin desplegar una diferencia visual en la barra de vida.

Nota : Se empieza pruebas con valores enteros ya que el damage puede ser cuantificado en enteros o flotante, pero es cuestión de prueba y ver que acomoda más a valores ya integrados.

*Pseudocódigo*

**Declaración de variables**

Int maxHealth con valor de 100

Int curHealth sin valor

txt healthCount (Esto recibirá el valor de la curHealth, transformándolo en un string posteriormente).

Int daño con valor 0

**Algoritmo**

Se inicializa curHealth con valor a 100

La barra de vida estará desplegada siempre en el costado izquierdo superior, de manera horizontal, con el valor desplegado curHealth, que se actualizará constantemente, y variara el UI de la barra dependiendo el monto de vida que se tiene, su máximo valor posible desplegado es de ”100”, por lo que cualquier otra entrada mayor a ella, no es desplegada, si no hasta posteriormente que se consigue alguna mejora de equipamiento y aumenta la vida.

Si curHealth es mayor a maxHealth, entonces curHealth sera igual a maxHealth

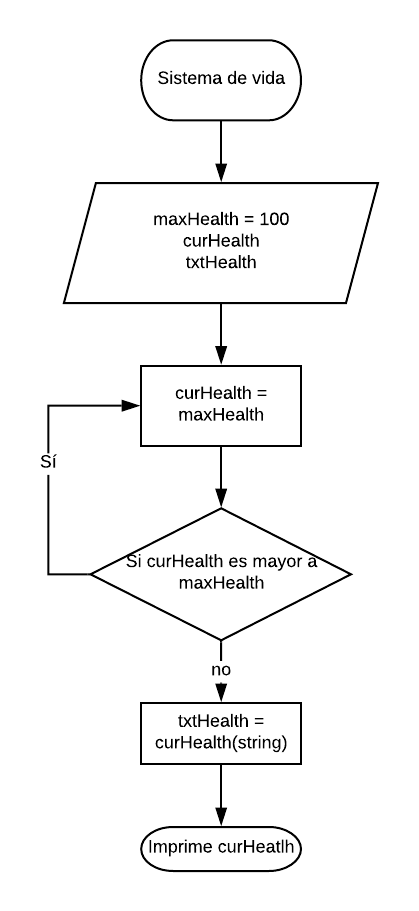
Si el box collider hace contacto con el target “enemigo” o “ataqueE”.

Obtener los componentes del “enemigo” o “ataqueE” , encontrar el valor de daño

Operación algorítmica curHealth= curHealth - daño

Imprimir curHealth

**Fin del Algoritmo**

*Diagrama de flujo*

*Movement System*

El sistema de movimiento será en 360 grados el los dos Axis que jugará (X, Y), para dar mayor fluidez al movimiento del personaje, si se usa el keyboard, se moverá en total en ocho direcciones cada una determinada por los puntos cardinales.

Gracias al rigidbody podemos construir físicas donde el jugador empiece desde un estado inicial y cuando llegue a su estado final, desacelerar que simula el “momento” adquirido con la velocidad.

*Pseudocódigo*

**Declaración de variables**

Se declara un Enum público que posteriormente se usará en el código para el ataque, esquive y otras variables

Declaración del estado público del player caminando

Variable flotante maxSpeed con valor a cero

Vector3 transform

Rigidbody

**Inicio del algoritmo**

Iniciar el animador obteniendo las propiedades de la animación correspondientes.

Declaramos la traslación inicial es cero.

Posterior obtener los componentes transform y cambiar de posición del player por medio de input getAxis, con las teclas asignadas al movimiento horizontal y vertical.

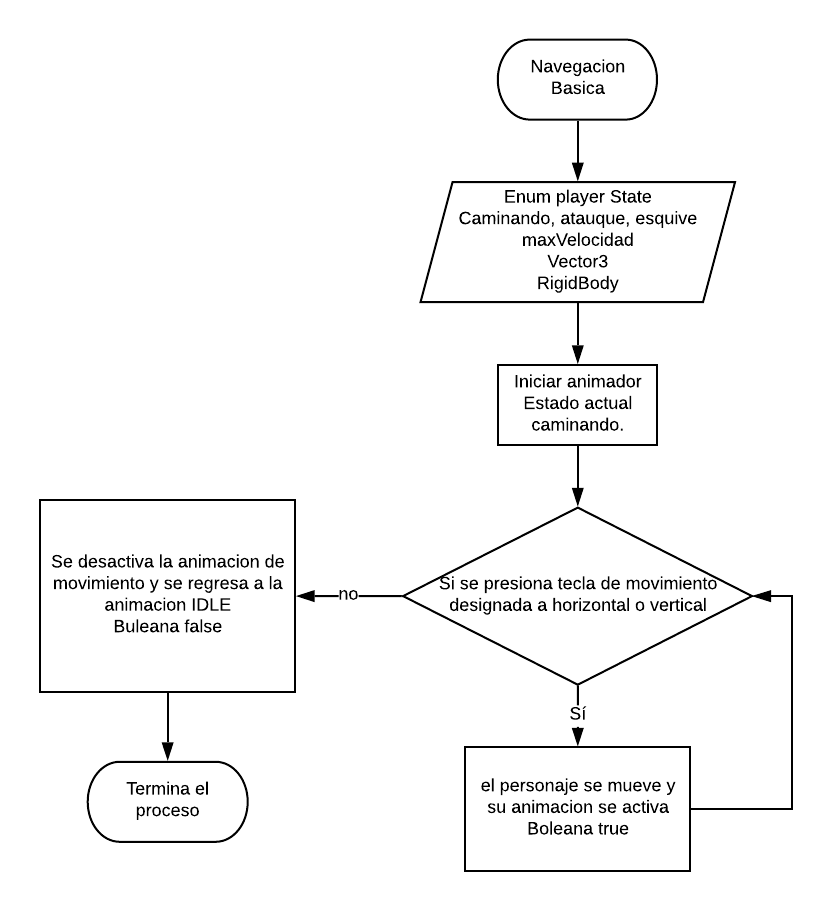
Transformar posición sumando traslación por máxima velocidad por delta time.

Por medio de If, si la traslación no es igual a su valor original, cero(Velocidad inicial)

set booleana para para animar caminata true  
Activa la animación correspondiente a dirección de traslación

else la velocidad vuelve a cero, setearla false (asi cambia de animación siempre estando en el estado “caminando”)

**Fin del algoritmo**

*Diagrama de flujo*

El diagrama está incompleto ya que cuenta con múltiples facetas, tal como la activación del ataque y del esquivo, que posteriormente se complementaran en el documento.

*Mitigación de daño (esquive) y Ataque*

Esto servirá para que el jugador tenga una manera de no recibir daño al moverse, con una penalización post realizado el esquive, de unos unos segundos.

Este código pertenece a la misma categoría de movimiento, por lo que solo se agregaran los cambios que se hacen dentro de la declaración y dentro del código.

Ambos serán comparados con el state “caminando” para evitar que se pueda cambiar de estado por frame entre ataque y esquive usando los a la vez.

*Pseudocódigo*

**Declaración de variables**

Agregar un enum al listado para la mitigación y ataque.

**Algoritmo**

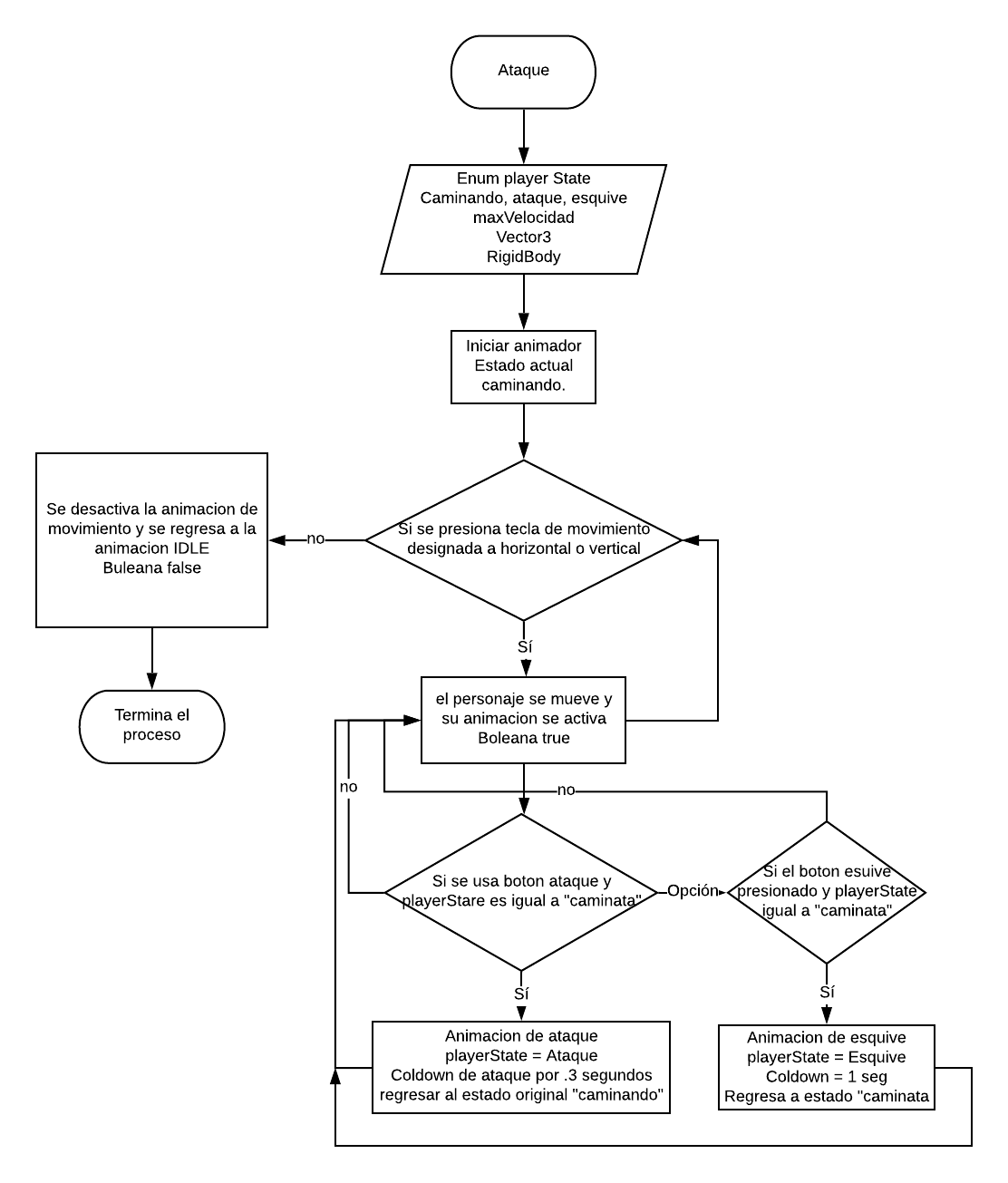
Utilizar un transform para modificar la velocidad normal del movimiento y multiplicar la por 1.5, para esquivar más rápido.

Usar una variable If para decir, “si se está caminando y es presionado el botón correspondiente a esquivar” entonces activar la animación de esquivar.

Definir que cuando se esquiva, se obtienen los componentes del sistema de vida, después si el target tiene el tag de “enemigo” , “ataqueE”, y el estado actual del player es esquive, que el daño sea cero.

Regresar el estado a caminar y definir un tiempo para evitar que se pueda esquivar siempre.

**Fin del algoritmo**

*Diagrama de flujo*

*Recibir daño del enemigo y hacer daño*

El daño será variable dependiendo al arma que se use en ese instante, la mejor manera de utilizarlo ahora mismo en con un valor entero equivalente a 10, para objeto de prueba y cinco para otro ataque de prueba.

*Pseudocódigo*

**Algoritmo (Ataque previamente establecido en el diagrama de flujo anterior)**

Se activa la animación de ataque y la boxCollider de la animación, correspondiente a la dirección que te encuentres observando.

Al acabar la animación y regresar al estado caminando

Enfriamiento de la animación por .3 segundos(valor de prueba)

**Fin del algoritmo 1**

**Algoritmo 2 (Enemigo vida)**

Declaración de vida sin valor

valor entero vidaActual = 0

valor entero dañoContacto;

valor entero dañoAtaque;

Box Collider dentro del enemigo prueba

Inicializar la vida con valor igual a 20.

Obtener los valores del boxCollider.

vidaActual = Vida

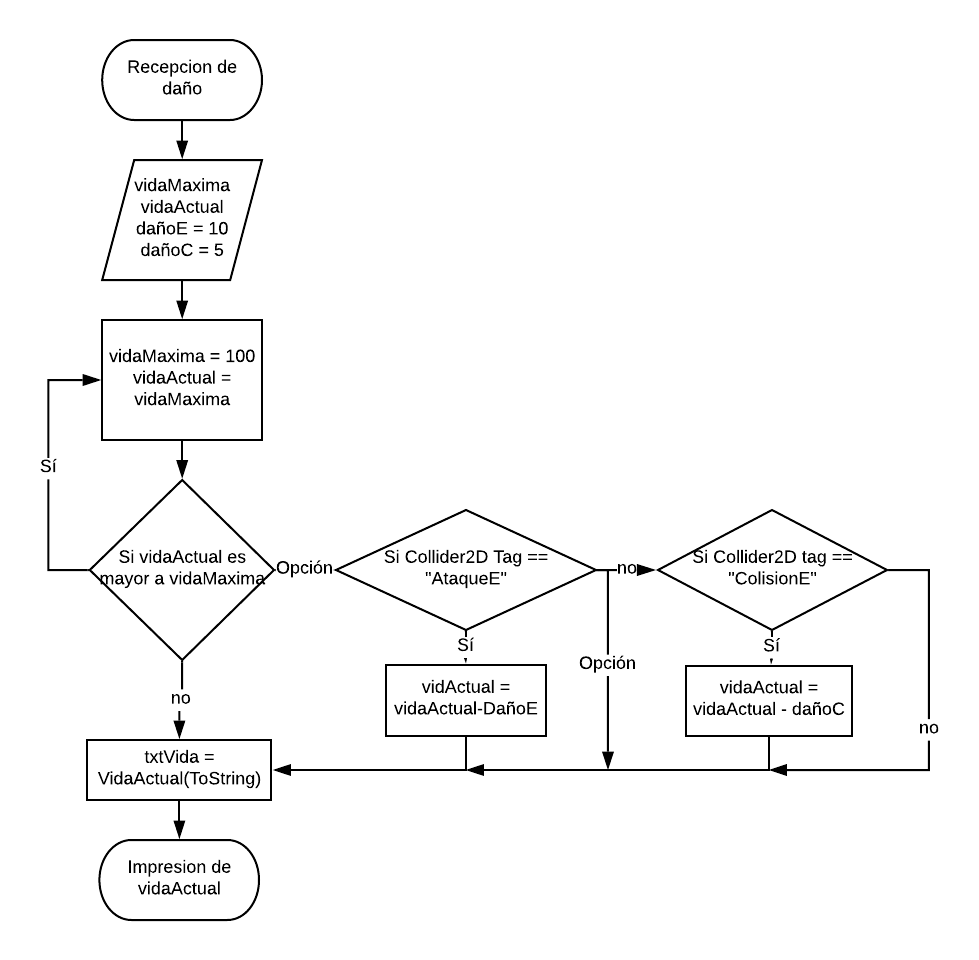
Si BoxCollider se encuentra con target “ataque”

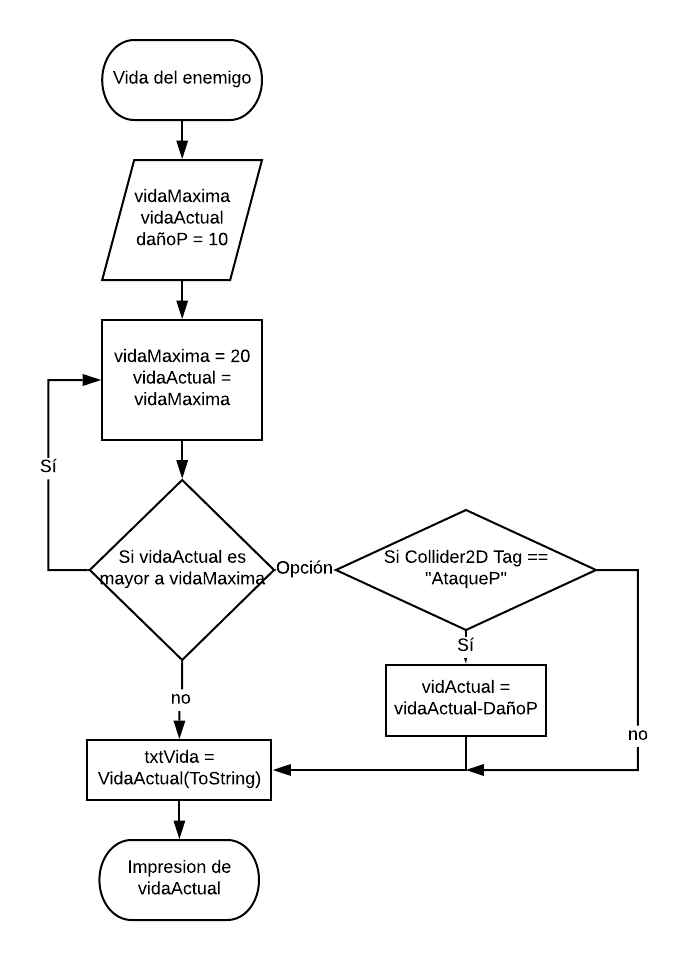
Obtener los valores del objeto con target ataque(player)

vidaActual = vida - daño (del objeto con target ataque)

Impresion de vidaActual

**Fin del algoritmo 2**

Diagrama de flujo



Seguimiento de cámara.

La cámara seguirá al jugador y tendrá un límite para no poder observar fuera de los bordes, así no tendremos que usar tantas texturas para tapar los bordes.

Pseudocódigo.

**Algoritmo**

Declaración de las variables con valor flotante, para un máximo y un mínimo, también una variable que seguirá al personaje.

Creamos un target que siga al personaje a donde vaya en el eje Z

Calculamos la esquina en el eje X y posteriormente en el eje Y (Máximo y mínimo)

Aplicamos el movimiento de la cámara suavizado.

**Fin del algoritmo.**

Destrucción de objetos (enemigos y objetos).

Este código ayudará a destruir los objetos que porten el mismo, por medio de comparación de tags.

Pseudocódigo

**Algoritmo**

Creamos una función que pueda ser llamada por cualquier consigo

Esta función contendrá destruir el objeto

También lo desactiva de la escena para mejorar el procesamiento de la memoria.

**Fin del algoritmo.**

Movimiento de enemigo 1

**Algoritmo.**

Declaramos variables de velocidad y otra que obtenga los datos del player.

Activamos la variable de los componentes del player y decimos que compare la distancia entre nuestro objeto y el player

Si el player está a cierta distancia mayor a un número y menor a otro, perseguirlo, si esta fuera de ese rango, dejarlo en su posición IDLE

**Fin del algoritmo.**