Trabajo Práctico 1: Demostración De Capacidades

Técnicas. Kirby's Adventure

Nehemias Lugo

Universidad Nacional de Rafaela

Taller de Diseño de Juegos 3

Andrés Rossi, Maximiliano Pedro Cadenazzi y Diego Porello

28 de Abril del 2025

Kirby 's Adventure es un videojuego de plataformas desarrollado por HAL Laboratory y publicado por Nintendo para la consola Nintendo Entertainment System (NES) en 1993. Ambientado en el colorido mundo de Dream Land, este título marcó un hito en la franquicia al introducir por primera vez la icónica habilidad de copia de Kirby, permitiéndole absorber los poderes de sus enemigos.

Consigna:

Réplica de 2 niveles específicos de un videojuego 2d con sus mecánicas, dinámicas y arte, puede utilizarse cualquier tipo de motor de programación de dominio propio. Se propone como aporte personal del equipo rediseñar el arte del personaje principal a criterio propio.

Link al Repositorio:

https://github.com/GigaPikachu/TP1-TDJ3

Apartados:

A continuación se muestra una lista con cada apartado que se debe cubrir, desde el arte (gráfico y sonoro) hasta el tecnico (programacion de poderes, movimientos y habilidades):

Artistico:

- Grafico:
  - Tiles:
    - Nivel 1
    - Nivel del Jefe
  - Sprites y Animaciones:

- Protagonista (reemplazar el sprite de Kirby)
- Hud
- Enemigos:
- Efectos y Ataques:
  - Aspiradora
  - Beam
  - Fire
  - Spark
  - Star

Sonoro:

- Efectos de Sonido:
- Musica:
  - Nivel 1 (Crane Fever Theme)
  - Musica de Jefe (Boss Theme)
  - Musica de Victoria (Kirby Dance Theme)

Fuentes de Letra (Kirby's-Adventure font)

Tecnico:

Escenas:

- Hud:
  - Vidas
  - Poder Actual del Protagonista
  - Puntaje (nivel 1)
  - Vida del Jefe (nivel del jefe)

- Nivel 1
- Nivel del Jefe

Entidades:

- Protagonista:
  - Movimientos
  - Poderes / Habilidades
  - Estados
- Enemigos:
  - Waddle Doo
  - Waddle Dee
  - Bronto Burt
  - Sparky
  - Hot Head
  - Whispy Woods (Jefe)

Arte:

La consigna pide una réplica del juego, el apartado visual y sonoro debe ser igual al original con la excepción del protagonista. Por lo tanto para ahorrar tiempo se va a optar por sacar la mayor cantidad posible de Assets gráficos, Música y Efectos de Sonido del juego original. Por suerte hay páginas en internet que recopilan todo esto como una biblioteca destinada a conservar contenido digital y hacerlo público y gratuito, como es el caso de Archive.org.

Grafico:

Kirby's Adventure fue hecho en una consola con capacidades muy limitadas como lo fue la NES, y aun así, su apartado Artístico de 8 bits la explotó al máximo, tenía gran cantidad de Animaciones, gran variedad de enemigos y el estilo de arte de los escenarios se hizo de forma que pareciera muy detallado a pesar de ser caricaturesco.

Tiles:

Los tiles fueron fáciles de conseguir, todos estaban recopilados en una página llamada The Sprite Resource. El único problema fue que esta página traía el mapa ya hecho completamente en una imagen, así que se tuvo que cortar, separar y eliminar los repetidos para hacer un Tileset. cada bloque de Tile es de 16x16px.

Nivel 1. https://www.spriters-resource.com/nes/kirbyadv/sheet/2637/

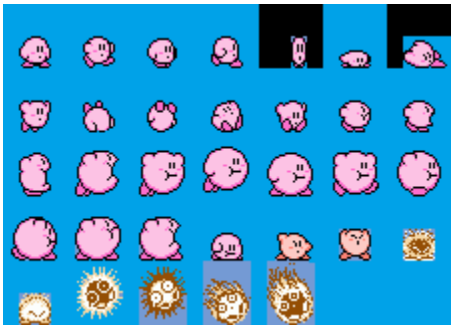
Nivel del Jefe. https://www.spriters-resource.com/nes/kirbyadv/sheet/49193/

Sprites y Animaciones:

Los Sprites y Animaciones también se pueden conseguir en la misma página que los Tiles. A diferencia de que todos los enemigos comunes están recopilados en una sola imagen, aunque son mucho más fácil de separar, ya que cada animación está agrupadas y no se repiten y todas están delimitadas por un color azul más claro de dónde comienza y termina el sprite.

Protagonista: El protagonista es el único Arte en todo el juego que se debe cambiar, aun así se decidió comenzar con la base del Sprite Sheet del protagonista original, Kirby.

Como solo se debía recrear 2 Niveles, y en solo 2 niveles Kirby no usa todas sus



animaciones, se tomó la decisión de crear un Sprite Sheet con los frames que son necesarios (quieto, caminar y saltar) y se van a agregar más a medida sean necesarias nuevas animaciones. La consola de la NES cargaba Sprites con tamaños múltiplos de 8 pixeles (siendo 8x8px el mínimo) y los sprites de Kirby llegaban a ser de hasta 24x24px los más grandes, se decidió hacerlo de 32x32px para tener un pequeño margen en caso de ser necesario pero no tan grande para mantener la armonía de escala del personaje.

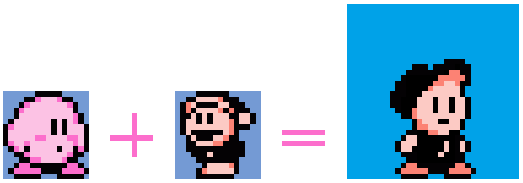
https://www.spriters-resource.com/nes/kirbyadv/sheet/49192/

El nuevo protagonista debe encajar con la estética del juego y con los poderes de Kirby. Desde un inicio se decidió que iba a ser un niño con una aspiradora, pero al tener una forma humana reconocible debe ser más grande que el tamaño normal del sprite de Kirby (16x16px). pero ya se tenía previsto eso como se mencionó anteriormente.

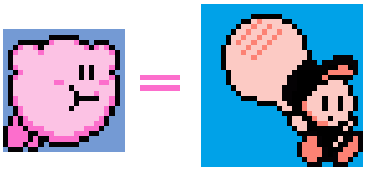
Para darle forma se mantienen los pies de Kirby, ya que al ser bastante grandes lo hacen fácil de expresar acciones al animar. Se le da una cabeza redonda con ojos grandes, el cuerpo y

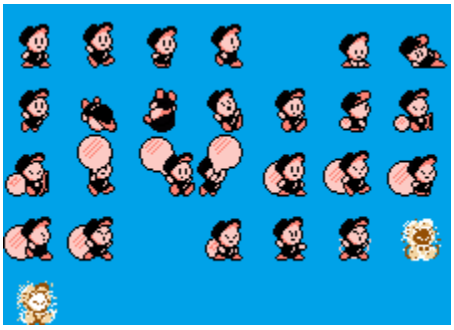
brazos pequeños y una gorra (que es lo que más lo identifica como un niño y para no tener que hacerle pelo). Todo estos rasgos exagerados ayudan a mantener el estilo caricaturesco.

Al diseño de Kirby se lo mezcló con el de un enemigo del juego original llamado Poppy Bros. Jr. ya que es el que más apariencia de humano tiene. Y así se llegó al diseño final.



Se tuvieron que cambiar el concepto de algunas animaciones para traducir lo que puede hacer Kirby a lo que podría hacer un niño con una aspiradora. Por Ejemplo, Kirby puede inflarse para volar, para hacer lo mismo el nuevo personaje infla la bolsa de su aspiradora. También al caminar cargando a un enemigo dentro fue cambiado por el personaje cargando la bola mostrando una cara de hacer fuerza.





Al final se descartaron algunos sprites ya que podían ser reemplazados por otros, pero de igual forma se mantuvieron los espacios vacíos para no tener que reconfigurar las animaciones en la programación.

Hud. https://www.spriters-resource.com/nes/kirbyadv/sheet/49204/

Enemigos. https://www.spriters-resource.com/nes/kirbyadv/sheet/49202/

Efectos y Ataques. La mayoría se encontraban repartidos en los Sprite Sheets del jugador y los enemigos, pero como muchos se repetían se los separó en un nuevo Sprite Sheet.

Sonoro:

Al igual que los gráficos, se puede los sonidos y las músicas se pueden sacar de páginas de internet, como Archive.Org. El problema es que no tienen un nombre y son difíciles de identificar, por lo que por ejemplo, si se quiere el sonido de aspiradora que hace Kirby al absorber a un enemigo se debe revisar todos los sonidos, y compararlos con el juego. Pero con la música es diferente, son fáciles de identificar su melodía y tienen un nombre.

Efectos de Sonido:

https://archive.org/details/kirbysadventuresoundeffects/

Musica:

https://archive.org/details/Kirby-Adventure-OST

Fuentes de Letra:

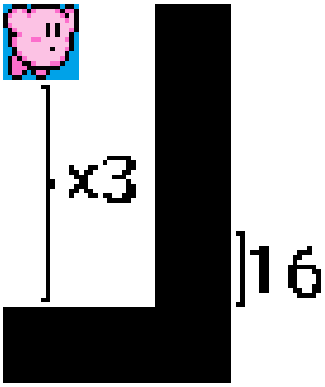
Es una fuente llamada Kirby 's Adventure y es usada únicamente en el Hud, fue sacada de la página dafont.com. https://www.dafont.com/es/kirbys-adventure.font

Tecnico:

Protagonista:

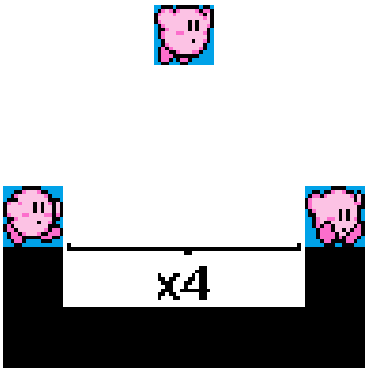
***Movimientos*:**

Comenzó con una demo técnica, en el que simplemente se iba a hacer una mapa de prueba con 2 colores para delimitar por donde moverse y donde colisionar. Agregamos al personaje con un sprite de kirby y sus habilidades más básicas, caminar y saltar.



Saltar: Fue más fácil saber hasta qué altura puede saltar el personaje en el videojuego, ya que se tiene un punto de referencia al que siempre va a volver (el suelo) con lo que se llegó a la conclusión de que Kirby puede saltar hasta superar un muro de 3 bloques de 16px, significando que un muro de 4 bloques no lo podría superar solo saltando, después fue cuestión de configurar y probar la gravedad y la fuerza que se le aplica al personaje.

En el juego original se salta con el boon A (no el UP), lo traduje a la tecla Z en teclado ya que es el que generalmente está asignado en un emulador y los jugadores que vengan de ahí van a estar familiarizados.



Desplazarse: Medir la velocidad por segundo a la que se puede mover es más complicado, sería muy impreciso y tardado usar un cronómetro y medir la distancia en píxeles que recorre en el juego original, el método que se usó para medirlo fue el propio salto, cuando Kirby salta mientras se mueve puede llegar hasta una plataforma a más o menos 4 bloques de distancia, siendo al que llega el 5º bloque, con esto ya pudimos ir midiendo la fuerza que hay que aplicarle para moverlo.

Volar: Kirby puede volar para llegar a lugares a los que no podría llegar saltando, pero su velocidad disminuye sintiéndose como mover un globo con un poco de viento (más pesado irónicamente). Estando así su velocidad vertical y horizontal es la misma, incluyendo la de caída por lo que se debe desactivar la gravedad y asignarle una caída de velocidad constante. Para volar tanto en Joystick como con teclado se usa la tecla UP y para volver a estar normal soltamos el aire con la tecla B (X en teclado).

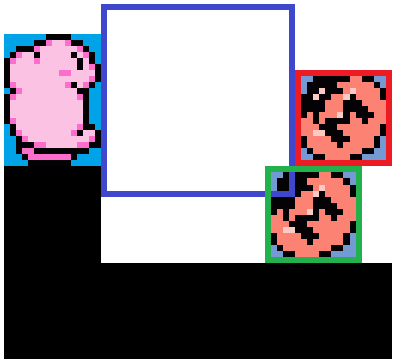
Se decidió que la animación se va a ejecutar si hay un movimiento, no solo por presionar un botón. Esto evitará que si el jugador está chocando con una pared pero se sigue presionando el botón para desplazarse se siga ejecutando la animación de caminar, en cambio el personaje solo estará en su sprite de quieto hasta que se le aplique una velocidad. Al igual que cambiar la dirección a la que el jugador mira, solo si tiene una velocidad hacia la izquierda el personaje cambiará su sprite.

Poderes / Habilidades:

La habilidad característica de Kirby es poder absorber a sus enemigos y con ello sus poderes. Al realizar cualquiera de estas acciones el jugador no se va a poder mover, y si estaba en el aire solo cara en línea recta vertical por gravedad.

Aspiradora. Comenzando con lo básico, el poder por defecto es el de absorber o aspirar objetos y enemigos. para eso usamos un sprite en blanco al que podamos darle físicas (debe ser una imagen para que en un principio podamos ver y luego podamos animar el efecto de aspirar aire), dependiendo de el lugar al que el personaje esté mirando debemos hacer que aparezca en un lugar o en otro respecto al personaje. Ahora cuando se presione el botón B (X en el teclado) aparecerá un rectángulo blanco que indica el área de la aspiradora y desaparecerá cuando se suelte el botón. Para descubrir el área de absorción se midió en el juego original y se descubrió

que los objetos a 3 cuadrados de distancia del personaje eran absorbidos, pero no se debe hacer un rectángulo de 3 bloques de distancia, sino de 2, ya que si se hace de 3 y absorbe un objeto, aunque este esté mayormente en el cuadrado 4 igual será absorbido por el mínimo contacto con



el área afectada. Pero verticalmente es diferente, el área de absorción también afecta a un bloque de arriba y uno de abajo del personaje, pero como los enemigos se mueven mayormente horizontal y no verticalmente nunca entraron en el contacto mínimo con el bloque arriba de ellos. Al final se decidió por un área de 32x32px.

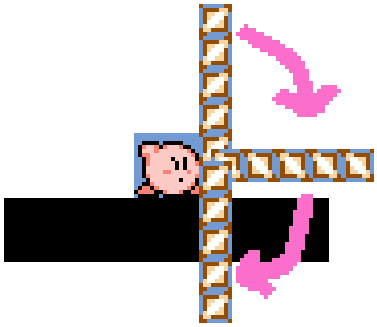
Ahora se le debe asignar propiedades para interactuar con objetos que el personaje pueda absorber. Se hizo un grupo de físicas llamada ObjetosAbsorbibles en el que se guardará tanto a enemigos como a objetos que el jugador se pueda comer. Dentro de la clase aspiradora se coloca una superposición en el que cualquiera que esté sobre el área de la aspiradora se moverá hacia el jugador y al tocarlo desaparecerá. Un error fue no desactivar las configuraciones de los colisionadores con el jugador, ya que si es un enemigo el que es absorbido cuando toque al jugador sufrirá daño, también se debe parar y eliminar la acción que esté haciendo el objeto absorbible, ya que si un enemigo es absorbido mientras realiza un ataque volverá a dañar al jugador.

Disparar. Al absorber un objeto el jugador pasa a un estado lleno en el que puede o tragar o disparar un proyectil con forma de estrella. parece simple, pero un error es que cuando el jugador absorbe a un enemigo, este puede seguir manteniendo presionada la tecla de acción lo que haría que el personaje lo dispare inmediatamente cuando el objeto lo toque aunque el jugador decidiera tragarlo. Para solucionarlo

hay que hacer una variable que pase a **TRUE** cuando se realice una acción y pase a **FALSE** cuando se suelte el boton de accion, entonces no solo hay que comprobar que el botón de acción esté presionado, sino que además la variable sea false, como con un gatillo de pistola (por eso les digos variables gatillo).

Este ataque se detiene cuando choca con un objeto o una pared.

Pero si el jugador decide tragar el objeto hay 2 posibilidades, si el objeto no da poderes no pasara nada, solo hará la animación de tragar y volverá a un estado normal, pero si da poderes, aparte de la animación de tragar, se hará una muestra del poder obtenido.



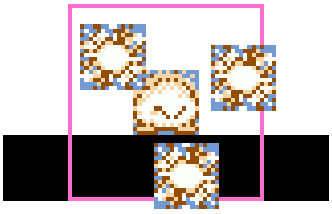
Beam. Es un rayo de media distancia (48px de distancia) que inicia en la mano del jugador y hace un movimiento parabólico de arriba a abajo. Para hacer este movimiento se creó 5 objetos de 8x8px, les asigne una variable llamada **radio** que es a la distancia a la que cada uno estará del jugador (el primero y más cercano o 8px, el siguiente a 8px más y así sucesivamente), después se creó una variable llamada **Ángulo** que iniciara en 90º (ya que el rayo debe comenzar arriba) y para actualizar la posición de cada parte del rayo se hace las siguientes ecuaciones.

BeamPart.X = BeamParts.radio x Cos(BeamParts.angulo + Velocidad);

BeamPart.Y = BeamParts.radio x Sin(BeamParts.angulo + Velocidad);

Luego solo hay que detener la actualización y eliminar cada parte cuando el rayo llegue hasta abajo.

Este ataque puede atravesar paredes y no se elimina si golpea a un enemigo, solo se elimina cuando termina su recorrido, por lo que no hay problemas con actualizar su posición sin usar físicas.



Spark. Es un ataque de área que rodea al jugador a 1 bloque de distancia a cada lado (horizontal y verticalmente) en el que se lanzan chispas en todas las direcciones.

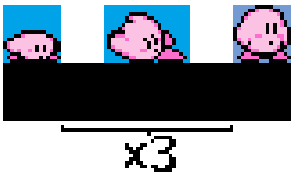
Para hacer este poder se hizo algo parecido a la aspiradora, en el que se creaba un sprite que marca el área afectada y elimina a los objetos que se superpongan (excepto al jugador), pero en vez de usarlo para una animación lo que se hace es crear un emisor de partículas, ya que en el juego original las chispas que lanza son aleatorias pero el área afectada siempre es la misma.

Fire. Lanza bolas de fuego de forma repetitiva a una corta distancia (32px), solo afecta al lado al que el jugador esté mirando de forma horizontal, no se mueve de ninguna manera en vertical.

Lanza muchos objetos de forma repetitiva como proyectiles que se destruyen rápidamente después de ser creados.

Este ataque no puede atravesar paredes como los otros, se destruyen al impactar con una pared dejando una animación de una pequeña explosion antes de desaparecer, motivo por el que no se podían usar efectos de partículas sin fisicas.

Dash: No depende del poder que tiene el jugador por lo que puede hacerlo teniendo cualquiera. El jugador debe estar en un estado normal, apretar el botón DOWN para agacharse y mientras lo mantiene apretado presionar el botón de acción (B / X).



El jugador ejecuta un dash en el que elimina a cualquier enemigo que atraviese. Recorre un total de 4 bloques, deteniéndose al terminar el último bloque.

Estados:

Para organizar cada acción que puede realizar el jugador sin que unas se estorben con otras se usó un método en el que se comprobaba el estado en el que esté el jugador (si estaba volando, si estaba tocando el suelo, o si estaba usando un poder, etc), fue muy complicado organizar de manera óptima a qué estados puede pasar el jugador, cual bloquea a cual o que acciones siempre se deben poder realizar, y se tuvo que reescribir el código un par de veces hasta llegar a la organización que se tiene ahora (las marcadas con un " * " no afectan a las demás y pueden realizarse aunque se ejecute otra acción de su misma jerarquía):

NoDañado (cuando el jugador es dañado entra en una pequeña animación en la que no puede hacer nada por unos milisegundos)

SoltarBotón (pasa a falso las variables que sean gatillos)

CambiarDirección (cambia la dirección y voltea al jugador dependiendo de su velocidad

AccionFalse (si no se está ejecutando ninguna habilidad o poder)

Desplazarse: Izquierda (le aplica una velocidad H al jugador negativa si se presiona la tecla)

Desplazarse: Derecha (le aplica una velocidad H al jugador positiva si se presiona la tecla)

NoMoverse (se deja la velocidad H en 0)

Volando

Acción: desinflarse (pasa a estado normal)

Subir: volar (sube con una velocidad constante mientras se presione UP o Salto)

Caer: bajar (con una velocidad baja y constante)

Lleno

Acción: disparar (pasa a estado normal)

EnElSuelo (se debe estar en el suelo para realizar)

Subir: saltar (el jugador solo puede saltar, no puede volar cargando algo)

Tragar (cambia el estado a normal pero puede cambiar la variable Poder)

Normal:

AcciónTrue (si se está ejecutando un ataque lo actualiza o detiene pasándolo a False)

AcciónButom (si se empieza a realizar una acción la inicia y la vuelve True cuando se presione)

Subir: Volar (pasa estado volando cuando se presione el botón)

EnElSuelo: (se ejecuta si está en el suelo)

Subir: Saltar (salta y pasa EnELSuelo a False hasta caer)

Abajo: Agachado (Pasa a estado agachado)

Agachado

Acción: Dash (pasa a estado normal después de la acción)

Así por ejemplo si el jugador está lleno no puede pasar a estar volando, y si esta lleno o normal pero no está en el suelo no puede saltar, en cambio en normal puede pasar a estar volando aunque no esté tocando el piso. pero en cualquier estado (a menos que se ejecute una acción) el jugador se va a poder moverse horizontalmente y cambiar de dirección incluso sin estar lleno/normal en el suelo.

Si el jugador es dañado entra en un estado en el que es empujado unos bloques en la dirección contraria a la del objeto que lo daño mostrando una pequeña animación que hasta que no pare el jugador no puede hacer nada. Pero también entra por unos segundos en Invulnerabilidad donde cuando los enemigos se chocan con el jugador estos mueren sin herirlo.

Enemigos:

Waddle Dee:

Es el enemigo más básico, su comportamiento se basa únicamente en cambiar de dirección si choca con una pared.

No da ningun poder

Waddle Doo:

Aparte de desplazarse horizontalmente y cambiar de dirección al golpear una pared tiene 2 compartimientos más, cuando se encuentra a 48px del jugador se detiene, hace una animación de carga y al terminar realiza un ataque **Beam**, y, cada 3 segundos (si no está haciendo un ataque) da un salto con el que podría saltar un muro de 2 bloques.

Al tragarlo da el poder **Beam**

Bronto Burt:

Este enemigo volador tiene 2 tipos de movimiento que no intercambian entre sí como con los estados, si se le asigna el movimiento 1 siempre va a hacer ese movimiento. En el primero este enemigo empieza a volar en una línea recta diagonal hasta salirse de la pantalla, solo hay que aplicarle una velocidad en el eje X y en el Y igual. El segundo es un zigzag en el aire, se hace aplicando una ecuación de movimiento en forma de Ondas en el que la velocidad horizontal es constante

BrontoBurt.Y = BaseY + amplitud * Sin(Frecuencia * BrontoBurt.x)

actualizamos la **posición Y** del enemigo multiplicando la **amplitud** de la onda por el Seno de la **frecuencia** (que tan ancha va a ser cada onda) por la posición del **enemigo en X** y le sumamos la **BaseY** (posición inicial Y)

No da ningun poder

Sparky:

Este enemigo tiene 4 movimientos bastante erráticos y random, el primero es un salto corto en el que no se mueve horizontalmente, los siguientes dos son saltos en los que persigue al jugador, pero uno es un salto corto y otro un salto largo, en los saltos cortos la fuerza aplicada es menor a la necesaria para poder saltar un solo bloque de 16px, pero en el salto alto puede saltar hasta un muro de 2 bloques. Por último está su ataque, es un **Spark** que dura 1 segundo y medio y solo ocurre si el jugador está a menos de 32px (2 bloques).

Todos (incluyendo el ataque) funcionan de manera errática, cada 700 milisegundos este enemigo va a tomar una decisión tirando un dado entre 1 y 3, si sale 1 y el jugador está cerca hace el ataque, si esta lejos hace un salto en el lugar, si sale 2 el enemigo hace un salto corto y si sale 3 hace un salto largo, no va a lanzar el dado si está realizando el ataque.

Al tragarlo da el Poder **Spark**

Hot Head:

Se desplaza horizontalmente y cambia de dirección al golpear una pared igual que el Wadlle Dee, pero además tiene 2 acciones de ataque que dependen de la distancia a la que esté el jugador. en el primer ataque, si el jugador se encuentra a 40px de distancia este enemigo puede cambiar su dirección para mirar al jugador y hacer un **Fire** igual a como lo haría el jugador con

ese poder durante 1 segundo, luego seguirá caminando en la dirección en la que hizo el ataque. Pero, si el jugador está a 80px de distancia, lanzará una bola de fuego como proyectil en línea recta en la dirección en la que estaba el jugador al momento de dispararla.

Para que si el jugador no se mueve este enemigo no esté constantemente realizando ataques y comience a caminar después de realizar cualquiera de los dos, tiene un tiempo de recarga de 3000 segundos antes de realizar el siguiente ataque.

Whispy Woods:

Es el primer jefe del juego original, tiene 2 ataques entre los que parece cambiar aleatoriamente.

El primero consiste en dejar caer manzanas en una posición X aleatoria, en total suelta 3 manzanas pero no al mismo tiempo, cuando aparece una manzana parpadea y después de un segundo comienza a caer y al tocar el piso aparece la siguiente manzana, la primera en vez de desaparecer hace un rebote y activa su velocidad en X en dirección al oponente desapareciendo cuando sale de la pantalla. El segundo ataque consiste en soplidos con forma de nube pequeñas, las suelta de forma consecutiva, la cantidad parece ser aleatoria entre 2 y 4.

mientras está siendo herido la pantalla tiembla y no puede hacer ningún ataque, pero si el primero (manzanas) ya está activo va a continuar hasta que caigan todas las manzanas.