Manual de Usuario

DARK MAZE Ghostly Decisions

Bienvenido al manual del usuario del juego "Dark Maze: Ghostly Decisions ". A continuación, se proporciona información sobre el funcionamiento del juego y las acciones que puedes realizar para disfrutar de la experiencia de juego.

Ericka González Romero
Eduardo Rafael Pérez Martínez

Profesor Dr. Francisco Javier Torres Reyes

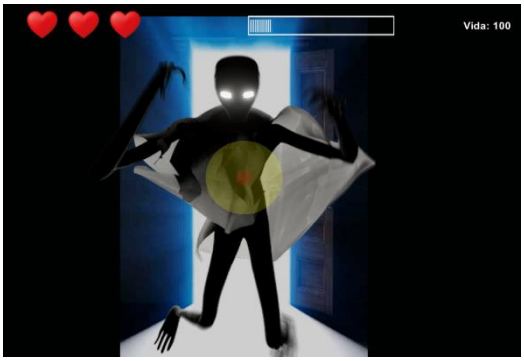
Descripción general

" *Dark Maze: Ghostly Decisions* " es un juego de aventuras en el que controlas a un personaje principal que debe enfrentarse a fantasmas en un entorno oscuro y desafiante. El objetivo del juego es llegar al final de cada nivel eliminando a los fantasmas antes de que el tiempo termine y nos reste nuestra vida.

Controles del juego

Movimiento del personaje principal: Utiliza la tecla "W" para avanzar hacia adelante.

Interacción con las puertas: Haz clic en las puertas para interactuar con ellas y avanzar al siguiente nivel.



Interacción con el fantasma: Mediante el cursor sigue al fantasma para disminuir su vida.

Temporizador: El temporizador muestra el tiempo restante para completar el nivel. Debes completar el nivel antes de que el temporizador termine.

Gameplay

Fantasma: Deberás tomar en cuenta el tiempo entre menos demores en disminuir la vida del fantasma hasta eliminarlo la probabilidad de seguir hacia el segundo nivel aumentara, de lo contrario perderás una vida y regresaras al nivel principal. Si pierdes todas tus vidas, el juego terminará.

Puertas: Interactúa con las puertas para avanzar al siguiente nivel. Cada puerta puede llevar por los diferentes niveles, por lo que debes elegir sabiamente.

Corazones: Cuida tus vidas al enfrentarte con el fantasma para poder mantenerte en el juego.

Tiempo extra: Podrá obtenerse únicamente el ultimo nivel, y tener la oportunidad de eliminar el fantasma más veloz, permitirá reiniciar el tiempo, es decir el tiempo tendrá valor doble.

Puntaje: El juego no tiene un sistema de puntaje, se centra en la exploración y vencimiento del enemigo.

Indicaciones adicionales

El juego se desarrolla en diferentes entornos oscuros, abandonados e inclusive misteriosos, al entrar al mundo oscuro elige correctamente y presta la suficiente atención a los movimientos de los fantasmas.
Utiliza el temporizador como una guía para administrar tu tiempo y completar los niveles.
¡Disfruta del juego " *Dark Maze: ¡Ghostly Decisions" y diviértete explorando los niveles mientras te enfrentas a los desafíos, que la suerte te acompañe!*

Manual del programador

DARK MAZE
Ghostly Decisions

Bienvenido al manual del programador del juego "Dark Maze: Ghostly Decisions ". A continuación, Se describirán las clases y métodos principales utilizados en el código, así como su funcionalidad y uso*, así como una sección que describe los problemas presentados y la resolución de estos.*

Clase BaseWorld

La clase BaseWorld es una clase abstracta que extiende de la clase World de Greenfoot. Sirve como base para la creación de mundos en el juego, proporcionando funcionalidades comunes como la gestión de fondos y la transición entre fondos.

Constructor

Recibe el número de imágenes de fondo, el prefijo del nombre de las imágenes, el índice de inicio y el incremento del índice para obtener los nombres de archivo de las imágenes de fondo.

Métodos

act: Método que se ejecuta en cada ciclo del juego. En este caso, solo llama al método checkKeyPressBck().

checkKeyPressBck: Método que gestiona la transición entre fondos. Cambia el fondo actual al siguiente fondo en la lista de fondos, y si se alcanza el último fondo, se crea una instancia de la clase Inicio y se establece como el mundo actual. Además, se borra el fondo anterior.

initBackgrounds:Método que inicializa los fondos del mundo. Recibe el prefijo del nombre de las imágenes, el índice de inicio y el incremento del índice. Genera los nombres de archivo de las imágenes de fondo y las carga en el arreglo backgrounds.

clearPreviousBackground: Método que borra el fondo anterior. Obtiene el índice del fondo anterior al actual y lo borra.

Clase Fin

La clase Fin es una subclase de BaseWorld. Representa el mundo final del juego.

Constructor

El constructor Fin(), llama al constructor de la clase BaseWorld con los parámetros específicos para este nivel.

Clase GameOver

La clase GameOver es una subclase de BaseWorld. Representa el mundo de "Game Over" del juego.

Constructor

Constructor de la clase GameOver, llama al constructor de la clase BaseWorld con los parámetros específicos para este nivel.

Clase Inicio

La clase Inicio es una subclase de BaseWorld. Representa el mundo de inicio del juego.

Constructor

Constructor de la clase Inicio, llama al constructor de la clase BaseWorld con los parámetros específicos para este nivel. También crea una instancia de GreenfootSound para reproducir música de fondo.

Métodos

checkKeyPressBck: Método que gestiona la transición entre fondos en el mundo de inicio. Además, reproduce la música de fondo y realiza la transición al siguiente nivel después de un cierto tiempo.

Clase BlackWorld

La clase BlackWorld es una subclase de World de Greenfoot. Representa el mundo principal del juego, donde el jugador se enfrenta a los fantasmas.

Constructor

Constructor de la clase BlackWorld, recibe el número de corazones y una bandera de nivel para determinar el nivel actual del juego.

Métodos

populateWorld: Método que crea instancias de la clase Corazon y las coloca en posiciones aleatorias dentro del mundo.
checkKeyPressBck: Método que gestiona la transición entre fondos. Cambia el fondo actual al siguiente fondo en la lista de fondos y borra el fondo anterior.
checkGameOver: Método que verifica si el juego ha terminado. Si el jugador se queda sin corazones, crea una instancia de la clase GameOver y la establece como el mundo actual.

Clase Corazon

La clase Corazon es una subclase de Actor de Greenfoot. Representa un corazón en el mundo del juego.

Constructor

Constructor de la clase Corazón, crea una instancia de la imagen del corazón y la establece como imagen de este actor.

Métodos

act: Método que se ejecuta en cada ciclo del juego. Verifica si el jugador ha tocado este corazón y, si es así, lo elimina del mundo y aumenta el contador de corazones del jugador.

Clase Phantom

La clase Phantom es una subclase de Actor que representa un fantasma en el juego. El fantasma se mueve de manera autónoma por el escenario y verifica las colisiones con otros objetos y las condiciones de juego.

Constructor

Constructor de la clase Fantasma, crea una instancia de la imagen del fantasma y la establece como imagen de este actor.

Métodos

act: Método que se ejecuta en cada ciclo del juego. Mueve al fantasma de manera aleatoria y verifica si ha tocado al jugador. Si es así, reduce el contador de corazones del jugador y, si el contador llega a cero, crea una instancia de la clase GameOver y la establece como el mundo actual.
mover: realiza el movimiento del fantasma según la velocidad y dirección actuales.
idle: realiza la animación del fantasma en estado de reposo.
initAnimationSpritesIdle: inicializa las imágenes utilizadas en la animación del fantasma en estado de reposo.
animateIdleWithDelay: realiza la animación del fantasma en estado de reposo con un retraso entre los fotogramas.
initAnimationSpritesSpawn: inicializa las imágenes utilizadas en la animación de aparición del fantasma.
spawn: realiza la animación de aparición del fantasma.
animateSpawn: realiza un fotograma de la animación de aparición del fantasma.
checkCollision: verifica las colisiones con el colisionador circular.
increaseSpeed: aumenta la velocidad máxima y la aceleración del fantasma al colisionar con el jugador.
checkLabelCount: verifica si el contador de vidas ha llegado a cero y realiza las acciones correspondientes.
checkTimerCount: verifica si el temporizador ha llegado a cero y realiza las acciones correspondientes.
updateLabelPosition: actualiza la posición de la etiqueta en relación con la posición del fantasma.

Clase Corredor

La clase Corredor es una clase abstracta que extiende la clase World. Representa a un corredor en el juego y proporciona funcionalidad común para los diferentes tipos de corredores. Esta clase contiene atributos y métodos que son compartidos por las subclases.

Constructor

El constructor de la clase recibe como parámetros n y banderaNivel para inicializar los corazones y el nivel del corredor.

Métodos

act: un método abstracto que debe ser implementado en las subclases. Define el comportamiento de actuar del corredor.
checkKeyPressBck: un método que verifica si se ha presionado la tecla de movimiento hacia adelante y cambia la imagen de fondo del corredor según el índice actual. También actualiza la variable z y verifica si se han agotado los corazones o si se ha alcanzado un límite de fondo.
clearPreviousBackground: un método que limpia la imagen de fondo anterior del corredor.
initBackgrounds: un método abstracto que debe ser implementado en las subclases. Inicializa las imágenes de fondo del corredor.
Otros métodos relacionados con la gestión de los corazones y la animación del corredor.

Clase PrimCorredor

La clase PrimCorredor es una subclase de la clase Corredor y representa un tipo específico de corredor en el juego. Extiende la funcionalidad de la clase Corredor y define los métodos abstractos initBackgrounds y act para implementar su propio comportamiento.

Constructor

El constructor de la clase recibe como parámetros n y banderaNivel y llama al constructor de la clase padre Corredor con los mismos parámetros.

Métodos

initBackgrounds: un método que inicializa las imágenes de fondo específicas del corredor PrimCorredor.
act: un método que define el comportamiento de actuar del corredor PrimCorredor. En este caso, implementa la lógica específica del corredor para cambiar la imagen de fondo según un patrón establecido.

Clase SegunCorredor:

La clase SegunCorredor es otra subclase de la clase Corredor y representa otro tipo de corredor en el juego. También extiende la funcionalidad de la clase Corredor y define los métodos abstractos initBackgrounds y act para su propio comportamiento.

Constructor

SegunCorredor: el constructor de la clase, recibe como parámetros n y banderaNivel y llama al constructor de la clase padre Corredor con los mismos parámetros.

Métodos

initBackgrounds: un método que inicializa las imágenes de fondo específicas del corredor SegunCorredor.
act: un método que define el comportamiento de actuar del corredor SegunCorredor. En este caso, implementa la lógica específica del corredor para cambiar la imagen de fondo según otro patrón establecido.

Clase TerCorredor

La clase TerCorredor es una subclase de la clase Corredor y representa un tercer tipo de corredor en el juego. También extiende la funcionalidad de la clase Corredor y define los

métodos abstractos initBackgrounds y act para su propio comportamiento.

Constructor

El constructor de la clase recibe como parámetros n y banderaNivel y llama al constructor de la clase padre Corredor con los mismos parámetros.

Métodos

initBackgrounds: un método que inicializa las imágenes de fondo específicas del corredor TerCorredor.
act: un método que define el comportamiento de actuar del corredor TerCorredor. En este caso, implementa la lógica específica del corredor para cambiar la imagen de fondo según otro patrón establecido.

Clase CircleCollider

La clase CircleCollider es una subclase de Actor que representa un colisionador circular. El colisionador se visualiza como un círculo rojo y se puede mover siguiendo la posición del mouse.

Constructor

Crea un nuevo colisionador circular con el radio especificado. Se inicializa una imagen con el círculo rojo y se establece la transparencia en 45.

Métodos

act: se ejecuta en cada ciclo de acto y actualiza la posición del colisionador según la posición del mouse.
setTransparency(int alpha): establece la transparencia del colisionador según el valor alpha especificado.

Clase Door

La clase Door es una subclase de Actor que representa una puerta en el juego. Al hacer clic en la puerta, realiza diferentes acciones según el valor del atributo destino.

Constructor

Crea una nueva puerta con el mundo especificado, el destino, el valor numérico y el nivel actual. Inicializa la imagen de la puerta cerrada y carga las imágenes de animación.

Métodos

act: se ejecuta en cada ciclo de acto y verifica si se ha hecho clic en la puerta. Realiza diferentes acciones según el valor de destino.

Clase Fed

La clase Fed es una subclase de Actor que representa un objeto utilizado en una animación. No realiza ninguna acción en el juego.

Constructor

Crea un nuevo objeto Fed y carga las imágenes de animación.

Métodos

act: se ejecuta en cada ciclo de act, pero no realiza ninguna acción.
fedInAnim: realiza una animación de entrada.
fedOutAnim: realiza una animación de salida.
clearPreviousBackground: borra el fondo de la imagen anterior en la animación.
initFedIn: inicializa las imágenes utilizadas en la animación.

Clase Label

La clase Label es una subclase de Actor que representa una etiqueta utilizada para mostrar información en el juego. La etiqueta muestra la cantidad de vidas restantes.

Constructor

Crea una nueva etiqueta con un contador inicializado en 100.

Métodos

act: se ejecuta en cada ciclo de acto pero no realiza ninguna acción.
act2: actualiza la etiqueta y el contador utilizando un valor de retraso aleatorio.
updateImage: actualiza la imagen de la etiqueta con el contador actual.
getCount: devuelve el valor actual del contador.

Clase Principal

La clase Principal es una subclase de Actor que representa al personaje principal en el juego. El personaje puede moverse hacia adelante al presionar la tecla "w".

Constructor

Principal: crea un nuevo personaje principal.

Métodos

act: se ejecuta en cada ciclo de acto y verifica las teclas presionadas para mover al personaje.

Clase Timer

La clase Timer es una subclase de Actor que representa un temporizador en el juego. El temporizador cuenta hacia atrás y muestra el tiempo restante en forma de contador.

Constructor

Crea un nuevo temporizador y carga las imágenes utilizadas en la animación.

Métodos

act: se ejecuta en cada ciclo de acto y actualiza el valor del temporizador, muestra el tiempo restante y realiza la animación del temporizador.
getTime: devuelve el valor actual del temporizador.
updateImage: actualiza la imagen del temporizador con el tiempo restante.
initAnimationSprites: inicializa las imágenes utilizadas en la animación del temporizador.

animateForward: realiza un fotograma de la animación del temporizador.

Clase YellowCircle

La clase YellowCircle es una subclase de Actor que representa un círculo amarillo en el juego. El círculo se puede mover siguiendo la posición del mouse.

Constructor

Crea un nuevo círculo amarillo con el radio especificado. Se inicializa una imagen con el círculo amarillo y se establece la transparencia en 50.

Métodos

act: se ejecuta en cada ciclo de acto y actualiza la posición del círculo según la posición del mouse.
setTransparency: establece la transparencia del círculo según el valor alpha especificado.
Esta es la documentación de las subclases de Actor proporcionadas en el código.

DESAFÍOS PRESENTADOS CON SU RESOLUCIÓN

1. No había documentación del efecto 3d fake. Solución: por medio de un video de youtube donde se explicaba las animaciones del Sprite, se pudo aplicar a las animaciones del fondo dando efecto de profundidad y al mismo tiempo el leve movimiento del arma del principal.
2. Debido a la grande cantidad de imágenes utilizadas en un arreglo, demorábamos mucho no era practico ya que se debían agregar todas de forma manual aumentado la probabilidad de error. Solución: Se pudo automatizar la inicialización y asignación de imágenes en el arreglo.
3. Búsqueda de videos con el efecto infity zoom: invertir mucho tiempo de búsqueda. Solución: aplicando palabras claves para filtrar mejor la búsqueda.
4. Tomar capturas a los videos para obtener los frames, era tardado, las imágenes eran irregulares y era difícil calcular los frames que se necesitarían. Solución: por medio del programa blender se automatizo la cantidad frames deseada, indicando el inicio del video y el intervalo de frames que se saltaría entre cada frame.
5. Error en memoria debido a la cantidad de arreglos de imágenes en los diversos mundos (considerando movimiento de los sprites). Solución: disminuir el tamaño en megas de las imágenes, una vez que el índice avanzaba, se removían completamente las imágenes que no se requerían.
6. Problemas con el movimiento del fantasma especialmente con las esquinas ya que era aleatorio. Solución: se realizo cuatro bloques de if, uno para cada esquina donde si se

cumple la condición el fantasma deberá cambiar de dirección generando dos números aleatorios.

7. Los contadores, y tiempos iban relativamente rápido por ende como solución se decidió utilizar una función en greenfoot, timer delay la cual permite parar los ciclos de la computadora.
8. Delay paraba todos los ciclos, alentando otros procesos. Solución: se aplicaron dos formas de delay, la principal función obtenida desde greenfoot y otra creada por nosotros por medio de condicionales.
9. Comunicación de variables: entre mundos y actores
10. Objetos superpuestos, cada que añadíamos nuevos objetos a algún mundo, este error se genera especialmente por trabajar mayormente con imgenes y/o sprites. Solución: Capas en greenfoot, cada objeto dependía del orden de como se agregaba, la solución fue tener cuidado en el orden al añadir un objeto en el mundo.
11. Función que genera la aleatoriedad en las puertas, ya que podrían ser siempre la misma. Solución: se creó una función que generara un arreglo de tres números aleatorios tomando en cuenta que ninguno de estos se repitiera, para acceder a cada número solo fue necesario asignar en tres variables diferentes cada índice del arreglo.
12. Música sonaba todo el tiempo. Solución: se decidió parar la música en cada cambio de mundo y volver a empezarla en el nuevo mundo.