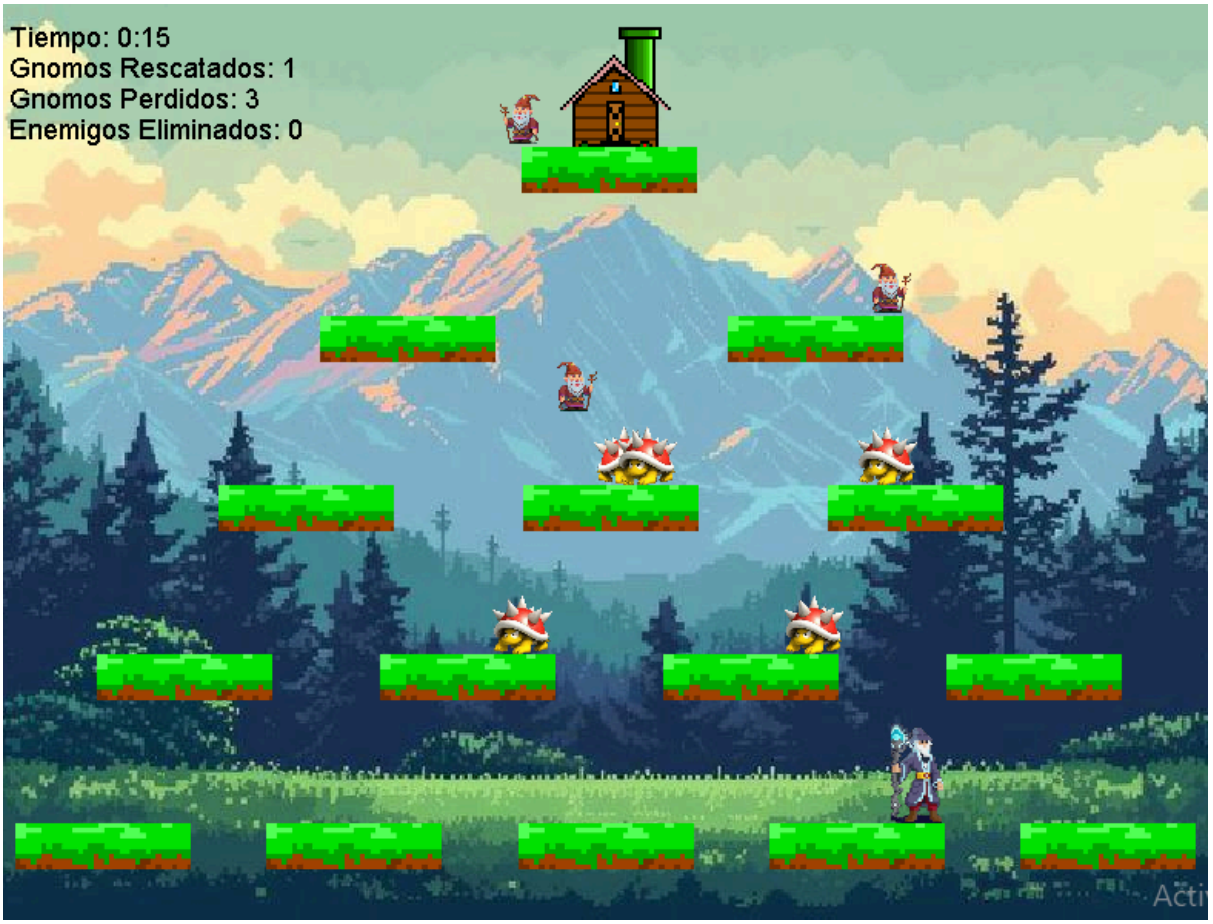
Trabajo practico de programacion I - Universidad nacional general sarmiento

Al rescate de los gnomos



Emanuel Sanchez - Ema.san2005@gmail.com

Franco Muñoz - franco4409@gmail.com

Martin Vidal - martinvidal04@gmail.com

Introducción

En este trabajo práctico se desarrolló un juego en Java denominado Pep al Rescate de Gnomos. En este juego, el jugador controla a Pep, un mago medieval, quien debe rescatar a gnomos atrapados en islas flotantes. Estas islas están amenazadas por tortugas venenosas de caparazón de acero. El objetivo es evitar que los gnomos caigan al vacío o sean destruidos por las tortugas. El proyecto plantea una serie de desafíos tanto en términos de lógica de juego como de control de entidades en un entorno bidimensional, que serán abordados y explicados en este informe.

Descripción

Clase Casa

Descripción general

La clase Casa representa una casa en el juego "Pep al rescate de Gnomos". Se encarga de dibujar la imagen de la casa en el entorno del juego.

Variables de instancia

- double x: Posición horizontal de la casa.
- double y: Posición vertical de la casa.
- double tamanio: Escala de la imagen, fija en 0.25.
- double ancho: Ancho de la imagen escalada.
- double alto: Alto de la imagen escalada.
- Entorno e: Entorno donde se dibuja la casa.
- Image imgCasa: Imagen de la casa.

Métodos

- Constructor Casa(double x, double y, Entorno ent): Inicializa las variables, carga la imagen y calcula ancho y alto.
- void dibujarCasa(Entorno e): Dibuja la casa en la posición (x, y) con la escala tamanio.
- double getX(): Retorna la posición x.
- double getY(): Retorna la posición y.

Problemas encontrados y soluciones

- Carga de imagen: Es crucial asegurarse de que la imagen se cargue correctamente. Implementar manejo de excepciones puede ayudar a detectar errores durante la carga.
- Coordenadas fuera de pantalla: Al instanciar la casa, verifica que las coordenadas se mantengan dentro de los límites de la ventana del juego para evitar problemas de visualización.

Clase Fondo

Descripción general

La clase Fondo gestiona la imagen de fondo del juego, permitiendo cargar y dibujar el fondo en el entorno.

Variables de instancia

- Image fondo: Imagen que representa el fondo del juego.
- Entorno e: Entorno donde se dibuja el fondo.

Métodos

- Constructor Fondo(String ruta, Entorno ent): Inicializa la variable fondo cargando la imagen desde la ruta proporcionada y asigna el entorno e.
- void dibujar(): Dibuja la imagen de fondo centrada en la ventana del juego, aplicando un escalado proporcional para ajustarse al tamaño del entorno. Utiliza el máximo entre las escalas de ancho y alto para mantener la relación de aspecto.

Problemas encontrados y soluciones

- Carga de imagen: Asegurarse de que la ruta de la imagen sea correcta es esencial. Se recomienda manejar excepciones para gestionar fallos en la carga de la imagen.
- Escalado de la imagen: El escalado debe ser proporcional para evitar distorsiones. Se usa el máximo entre las escalas de ancho y alto para lograr un ajuste adecuado, lo que garantiza que la imagen se mantenga en proporción al cambiar el tamaño de la ventana.

Clase Gnomo

Descripción general

La clase Gnomo representa a los gnomos en el juego "Pep al rescate de Gnomos". Cada gnomo tiene una posición en el espacio (coordenadas x e y), una imagen que lo representa y lógica de movimiento y gravedad. Los gnomos se mueven horizontalmente y caen debido a la gravedad cuando no están apoyados.

Variables de instancia

- double x: posición horizontal del gnomo en la ventana del juego.
- double y: posición vertical del gnomo en la ventana del juego.
- double tamanio: escala de la imagen del gnomo, que se establece en 0.10.
- double ancho: ancho de la imagen del gnomo escalada.
- double alto: alto de la imagen del gnomo escalada.
- boolean direccion: indica la dirección en la que el gnomo está mirando (verdadero = izquierda, falso = derecha).
- boolean estaApoyado: indica si el gnomo está apoyado sobre una superficie (verdadero = apoyado, falso = en el aire).
- Image imagenDer: imagen del gnomo mirando hacia la derecha.
- Image imagenIzq: imagen del gnomo mirando hacia la izquierda.
- Entorno e: objeto que representa el entorno de juego donde se dibuja el gnomo.

Métodos

- Constructor Gnomo(double x, double y, Entorno ent):
  Inicializa las variables de instancia.
  Se establecen las posiciones iniciales x e y, se carga las imágenes del gnomo y se calcula su ancho y alto en base al tamanio.
- void cambiarDireccion():
  Cambia aleatoriamente la dirección en la que el gnomo mira (izquierda o derecha).
- void movimientoGnomo():
  Mueve al gnomo horizontalmente según su dirección actual.
  Si direccion es verdadera, se decrementa x, moviendo al gnomo hacia la izquierda.
  Si es falsa, se incrementa x, moviéndolo hacia la derecha.
- void gravedad():
  Aplica un efecto de gravedad al gnomo.

Si el gnomo no está apoyado (estaApoyado es falso), se incrementa y en 2 unidades, simulando la caída.

- void dibujarGnomo(Entorno e):
  Dibuja el gnomo en su posición actual.
  Si el gnomo está mirando a la izquierda, se dibuja la imagen correspondiente, de lo contrario se dibuja la imagen mirando a la derecha.
- double getBordeDer():
  Devuelve la coordenada x del borde derecho del gnomo.
- double getBordeIzq():
  Devuelve la coordenada x del borde izquierdo del gnomo.
- double getBordeSup():
  Devuelve la coordenada y del borde superior del gnomo.
- double getBordeInf():
  Devuelve la coordenada y del borde inferior del gnomo.

Problemas encontrados y soluciones

- Problema de dirección:
  Al inicializar un gnomo, existe la posibilidad de que la dirección inicial sea siempre la misma si no se implementa una lógica aleatoria adecuada. Esto se soluciona en el constructor al usar Math.random() < 0.5 para determinar la dirección inicial.
- Caída en el aire:
  Si el gnomo no tiene un sistema adecuado para detectar si está en una superficie, puede caer indefinidamente. La variable estaApoyado se utiliza para verificar esto y aplicar gravedad solo cuando no está en una superficie.
- Cálculo de dimensiones:
  Al calcular el ancho y alto, se usa incorrectamente el orden al multiplicar, lo que podría dar dimensiones inesperadas. Se corrige asegurándose de multiplicar getWidth() por tamanio para alto y getHeight() para ancho, siguiendo la lógica de escalado.

Clase Isla

Descripción general

La clase Isla representa una isla en el juego, gestionando su posición, tamaño y visualización en el entorno.

Variables de instancia

- double x: Coordenada horizontal de la isla.
- double y: Coordenada vertical de la isla.
- Image img: Imagen que representa la isla.
- double tamanio: Factor de escala de la imagen de la isla.
- Entorno e: Entorno en el que se dibuja la isla.
- double ancho: Ancho calculado de la isla según la imagen y el tamaño.
- double alto: Alto calculado de la isla según la imagen y el tamaño.
- boolean direccion: Indica la dirección en la que puede estar la isla (derecha o izquierda).
- boolean estaApoyado: Indica si la isla está apoyada o no.

Métodos

- Constructor Isla(double x, double y, Entorno ent): Inicializa la posición de la isla con las coordenadas proporcionadas, carga la imagen de la isla, calcula su tamaño y asigna el entorno e. Establece direccion como false (derecha) y estaApoyado como false.
- double getBordeDer(): Calcula y devuelve la coordenada del borde derecho de la isla.
- double getBordeIzq(): Calcula y devuelve la coordenada del borde izquierdo de la isla.
- double getBordeSup(): Calcula y devuelve la coordenada del borde superior de la isla.
- double getBordeInf(): Calcula y devuelve la coordenada del borde inferior de la isla.
- void mostrar(): Dibuja la imagen de la isla en el entorno utilizando sus coordenadas y tamaño.
- double getX(): Devuelve la coordenada x de la isla.
- double getY(): Devuelve la coordenada y de la isla.

Problemas encontrados y soluciones

- Carga de imagen: Asegurar que la ruta de la imagen sea correcta es fundamental. Se sugiere implementar un manejo de excepciones para detectar errores en la carga de imágenes.
- Cálculo de dimensiones: Los cálculos de ancho y alto se basan en el tamaño de la imagen y deben ser precisos para una correcta representación visual. Se verifica

que el tamaño del tamanio sea adecuado para que la isla se visualice correctamente en el entorno.

Clase Personaje

Descripción general

La clase Personaje representa al jugador en el juego, gestionando su posición, apariencia y comportamiento, incluyendo el movimiento y el salto.

Variables de instancia

- double x: Coordenada horizontal del personaje.
- double y: Coordenada vertical del personaje.
- int contadorSalto: Contador que controla la duración del salto.
- Image imagenDerecha: Imagen del personaje mirando hacia la derecha.
- Image imagenIzquierda: Imagen del personaje mirando hacia la izquierda.
- double tamanio: Factor de escala de la imagen del personaje.
- Entorno e: Entorno en el que se dibuja el personaje.
- double ancho: Ancho calculado del personaje según la imagen y el tamaño.
- double alto: Alto calculado del personaje según la imagen y el tamaño.
- boolean direccion: Indica la dirección en la que mira el personaje (derecha o izquierda).
- boolean estaApoyado: Indica si el personaje está apoyado en el suelo.
- boolean estaSaltando: Indica si el personaje está en estado de salto.

Métodos

- Constructor Personaje(double x, double y, Entorno ent): Inicializa la posición del personaje y carga las imágenes necesarias. Establece los valores iniciales de contadorSalto, tamanio, direccion, estaApoyado y estaSaltando.
- void dibujarPersonaje(Entorno e): Dibuja la imagen del personaje en el entorno, eligiendo la imagen correspondiente según la dirección en la que está mirando.
- void movHorizontal(double m): Mueve al personaje horizontalmente según el valor m, actualizando su posición. También limita su movimiento para que no salga del entorno.
- void gravedad(): Aplica la gravedad al personaje si no está apoyado ni saltando, y gestiona el comportamiento del salto. Controla el movimiento vertical durante el salto y reinicia el contador al completar el salto.

- double getBordeDer(): Calcula y devuelve la coordenada del borde derecho del personaje.
- double getBordeIzq(): Calcula y devuelve la coordenada del borde izquierdo del personaje.
- double getBordeSup(): Calcula y devuelve la coordenada del borde superior del personaje.
- double getBordeInf(): Calcula y devuelve la coordenada del borde inferior del personaje.
- void saltar(): Inicia el salto del personaje, estableciendo estaSaltando y estaApoyado.
- void cancelarSalto(): Cancela el salto, reiniciando el contador de salto.

Problemas encontrados y soluciones

- Manejo de saltos: El control del salto se implementó mediante un contador que limita la duración del mismo. Se deben realizar ajustes en la gravedad y el movimiento para que el salto sea fluido.
- Límites del entorno: Se implementaron verificaciones en el método movHorizontal para asegurar que el personaje no pueda salir de los límites del entorno, evitando errores en la posición.

Clase Poder

Descripción general

La clase Poder representa una bola de poder que puede ser lanzada en el juego. Gestiona su posición, apariencia y comportamiento al ser lanzada.

Variables de instancia

- double x: Coordenada horizontal de la bola.
- double y: Coordenada vertical de la bola.
- double tamanio: Factor de escala para la imagen de la bola.
- double velocidad: Velocidad a la que se mueve la bola.
- double ancho: Ancho calculado de la bola basado en la imagen y el tamaño.
- double alto: Alto calculado de la bola basado en la imagen y el tamaño.
- boolean direccion: Indica hacia dónde se lanza la bola (izquierda o derecha).
- Image bolaIzq: Imagen de la bola cuando se lanza hacia la izquierda.
- Image bolaDer: Imagen de la bola cuando se lanza hacia la derecha.

- Entorno e: Entorno en el que se dibuja la bola.

Métodos

- Constructor Poder(double x, double y, Entorno ent, boolean direccion): Inicializa la posición de la bola, carga las imágenes necesarias y establece la dirección en la que se lanzará. También define el tamaño y la velocidad de la bola.
- void dibujarBola(Entorno e): Dibuja la imagen de la bola en el entorno, seleccionando la imagen adecuada según la dirección en la que se está lanzando.
- void lanzarBola(): Actualiza la posición de la bola según su dirección, moviéndola a la izquierda o derecha a una velocidad definida.
- double getX(): Devuelve la coordenada horizontal x de la bola.
- double getY(): Devuelve la coordenada vertical y de la bola.

Problemas encontrados y soluciones

- Control de dirección: Se implementó una lógica que determina qué imagen utilizar y cómo actualizar la posición de la bola en función de la dirección. Esto asegura que el jugador pueda ver correctamente hacia dónde se está lanzando.
- Velocidad constante: La velocidad de movimiento se establece como una constante, lo que permite que la bola mantenga un desplazamiento uniforme.

Clase Tortuga

Descripción general

La clase Tortuga representa una tortuga en el juego, gestionando su posición, apariencia y comportamiento, incluyendo movimiento y gravedad.

Variables de instancia

- double x: Coordenada horizontal de la tortuga.
- double y: Coordenada vertical de la tortuga.
- double tamanio: Factor de escala para la imagen de la tortuga.
- double alto: Alto calculado de la tortuga basado en la imagen y el tamaño.
- double ancho: Ancho calculado de la tortuga basado en la imagen y el tamaño.
- double velocidad: Velocidad a la que se mueve la tortuga.
- boolean direccion: Indica hacia dónde se mueve la tortuga (izquierda o derecha).
- boolean estaApoyado: Indica si la tortuga está en contacto con una superficie.
- Image imagenDer: Imagen de la tortuga mirando hacia la derecha.

- Image imagenIzq: Imagen de la tortuga mirando hacia la izquierda.
- Entorno e: Entorno en el que se dibuja la tortuga.

Métodos

- Constructor Tortuga(double x, double y, Entorno ent): Inicializa la posición de la tortuga, asegurándose de que esté dentro de posiciones válidas en el eje x. Carga las imágenes necesarias, define el tamaño, la velocidad y establece su posición inicial en y.
- void movimientoTortuga(Isla i): Controla el movimiento de la tortuga. Cambia la dirección cuando la tortuga alcanza los bordes de la isla pasada como parámetro.
- void dibujarTortuga(Entorno e): Dibuja la imagen de la tortuga en el entorno, seleccionando la imagen adecuada según la dirección en la que se está moviendo.
- void gravedad(): Aplica la gravedad a la tortuga, incrementando su coordenada y si no está apoyada.
- double getBordeDer(): Devuelve la posición del borde derecho de la tortuga.
- double getBordeIzq(): Devuelve la posición del borde izquierdo de la tortuga.
- double getBordeSup(): Devuelve la posición del borde superior de la tortuga.
- double getBordeInf(): Devuelve la posición del borde inferior de la tortuga.

Problemas encontrados y soluciones

- Posición válida: Se implementa un ciclo para asegurar que la tortuga aparezca en posiciones válidas, evitando las áreas no deseadas del entorno. Esto se logra mediante una verificación de coordenadas en el constructor.
- Movimiento controlado: La tortuga cambia de dirección al llegar a los límites de la isla, asegurando un movimiento fluido dentro de su espacio de juego.

Conclusiones

El desarrollo del juego "Pep al rescate de Gnomos" ha sido un viaje enriquecedor para nuestro equipo. A través de este proyecto, hemos podido aplicar y consolidar conceptos de programación orientada a objetos, gestión de eventos y diseño de interfaces, lo que nos permitió crear un simulador atractivo y funcional.

Una de las principales lecciones aprendidas fue la importancia de la comunicación y la colaboración en equipo. Cada miembro aportó diferentes habilidades y perspectivas, lo que enriqueció el proceso de desarrollo y nos ayudó a superar los desafíos que se presentaron.

La gestión de tiempos y tareas fue clave para cumplir con los plazos establecidos y asegurar que cada aspecto del juego se desarrollara de manera ordenada.

Además, enfrentamos y resolvimos problemas técnicos relacionados con la física del juego y la detección de colisiones, lo que nos enseñó a investigar y encontrar soluciones creativas a los obstáculos. La experiencia adquirida en la implementación de mecánicas de juego nos dará una base sólida para futuros proyectos en el desarrollo de videojuegos.

En resumen, estamos orgullosos del resultado obtenido y de las habilidades adquiridas a lo largo de este trabajo práctico. La creación de "Pep al rescate de Gnomos" no solo ha sido una oportunidad para aprender sobre programación, sino también para disfrutar del proceso creativo y de la diversión que los videojuegos pueden ofrecer.