Quiros

Ejemplo de juego con Android Studio y sin Framework

Software de entretenimiento y videojuegos

Universidad de Oviedo

Adrián Bueno

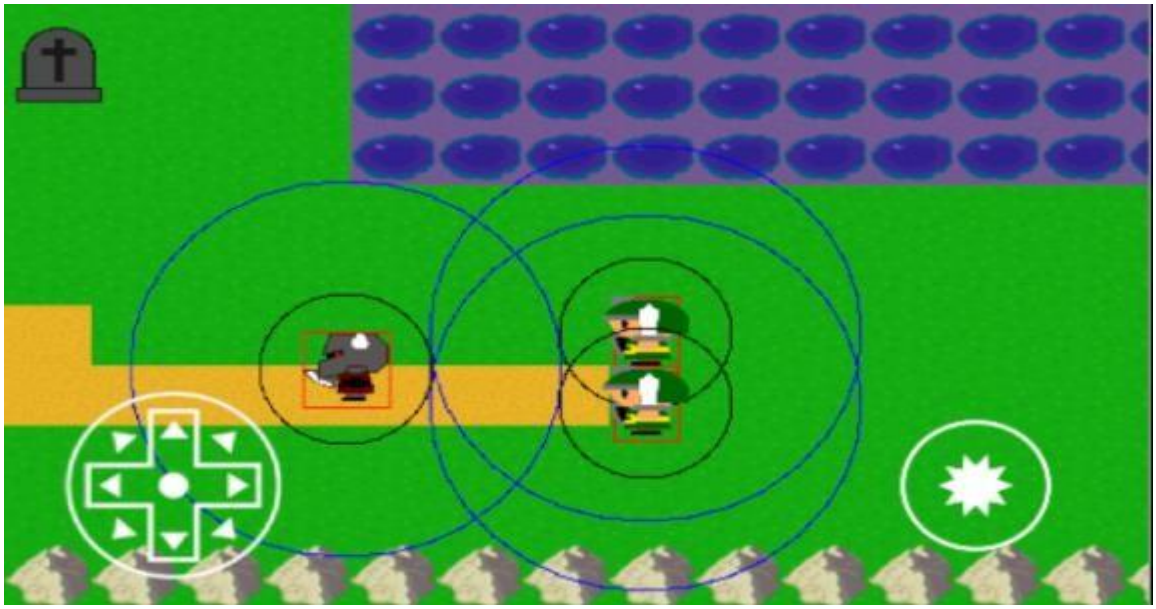
Uo232346@uniovi.es

Introducción

He realizado un juego con perspectiva superior en el que se maneja a un único personaje en 4 direcciones diferentes (arriba, abajo, izquierda y derecha), en el juego hay varios niveles los cuales se **completan cuando se elimina a todos los enemigos** del nivel.

Pantalla de inicio

Pantalla de juego



PARA EMPEZAR.

(Al empezar tienes 2 enemigos a la derecha, la catapulta aún más allá) y un solo arquero a la izquierda, activa el modo debug (botón 1) y acércate, puedes comprobar lo que hacen cuando entras y sales de su círculo externo, que es el área de detección)

Como Jugar

Se pueden usar las **teclas (w, A, S y D)** para moverse y la **barra espaciadora** para atacar, también hay controles en la pantalla para moverse y atacar. Además, el **botón 1** del teclado activa la vista en **modo DEBUG (HAZLO AL EMPEZAR)**, dónde se podrá ver aspectos como el área de colisión, área de ataque y de detección.
Usado los controles el jugador puede moverse hacia un enemigo y atacarle, también puede recoger varios tipos de coleccionables para recuperar su vida, aumentar su velocidad o incrementar su área de ataque.
Para dañar un enemigo hay que acercarse bastante a él.

Elementos del juego.

Los enemigos, cuando detecten al jugador irán a atacarle o le atacarán a distancia si pueden. Hay diferentes tipos de enemigos, los más débiles son los verdes, los azules y los rojos tienen diferentes características, unos son más rápidos, tienen más vida, tienen más área de detección o de ataque… En general el rojo es más fuerte. También hay arqueros y catapultas, que disparan proyectiles que hacen diferente cantidad de daño y tienen diferente velocidad.
Hay 2 tipos de coleccionables, las monedas que aumentan la velocidad y el área de ataque y las gemas que recuperan 2 puntos de vida, el jugador puede recogerlos acercándose a ellos.

Funcionalidad.

Menú principal.

Hay una pantalla de inicio con 2 botones, uno para salir y otro para jugar, esta pantalla está implementada en 'MainActivity', al pulsar sobre el botón 'Jugar' se pasa otra pantalla 'GameActivity'.

Controles del juego.

Los 2 controles están en el paquete controles El **botón de ataque**, está a la derecha, cuando se pulsa cambia un valor booleano en la clase "Nivel", que es comprobado al procesar las órdenes del jugador.

El **botón de control** es el de la izquierda, según en la parte en la que se pulse variará la dirección del jugador.

Detalles de implementación.
En la clase "pad" puede verse que divido el **control en 4 zonas** (4 **ángulos de 90º**), como cada zona tiene un ángulo de 90º puede haber **2 direcciones** por zona, la correspondiente a la **inferior de los 45º y la superior**. **Al hacer click tengo un punto** y puedo hacer una **recta desde el punto central del control al punto del click**, como en este punto **tengo las coordenadas x e y** (relativas al centro del control) **se los lados del triángulo** que acabo de formar y a no ser que x e y sean iguales este triángulo nuevo tendrá un ángulo diferente. Comparando las 2 coordenadas y teniendo muy en cuenta la zona en la que estoy, dependiendo de cuál es mayor será una orientación u otra.

Elementos del juego móviles.

Los jugadores, enemigos y disparos comparten la interfaz **Movible**, que se usa en el método aplicar reglas de movimiento, de este modo el **código que se usa es el mismo** para los 3 tipos de elementos. La implementación puede verse en la interfaz **Movible** y en el método de **Nivel aplicarReglasDeMovimiento().**

Comportamiento

TE HE PUESTO VALORES BAJOS EN LAS AREAS DE DETECCION EN LOS ENEMIGOS PARA QUE NO TE DE PROBLEMAS AL MOVERTE, LA CATAPULTA LO SIGUE TENIENDO ALTO.

Áreas de ataque y detección

El jugador y los enemigos comporten estas variables.

- **aRadio**: Radio para calcular el área de **Ataque Físico.** Solo se puede atacar físicamente a quién esté dentro de esta área. (modelo.puedeAtacar(Imodelo))
- **pRadio**: Radio de detección, sobretodo usado en los enemigos, cuando el jugador entra dentro del área de detección los enemigos le disparan o le persiguen hasta que el jugador entra en su área de ataque. (modelo.detecta(Imodelo))
- Las 4 variables de colisión, vida, velocidad base, ca (clase de armadura) ect.

Las diferentes áreas se controla su colisión de una manera similar a la colisión mostrada en las prácticas.

Clase de armadura

Cuando se es atacado se **hace una tirada aleatoria** que se compara con la **clase de armadura del jugador** (variable **ca**) si se supera el ataque hace el daño definido en la clase Enemigo, (si el jugador se mueve es más difícil darle aún).
Los enemigos también tienen Clase de armadura, pero no está implementada la funcionalidad de fallo por parte del jugador.

Enemigos, tipos y habilidades.

Existen **2 tipos de enemigos**, los que atacan **físicamente** y los que atacan a **distancia**.
Los enemigos de ataque físico, son los soldados verdes, azules y rojos. Cuándo **se entra en su área de detección** (diferente para cada uno) estos intentan **perseguir** al jugador hasta que este sale de esa área. Si el jugador entra en su **área de ataque** podrán **atacar al jugador**, se hará la tirada de ataque.

Los arqueros y catapultas lanzan proyectiles al jugador, los arqueros tienen mayor cadencia de tiro, pero sus disparos son más lentos.

- Arqueros: Dispararán al jugador cuando entren en su área de detección. **Si el jugador se acerca** y el **arquero entra en el área de ataque del jugador**, el arquero dejará de atacar e intentará alejarse en la dirección opuesta y después volverá a disparar.
- Catapultas, tienen un **área de detección muy grande** y dispararán al jugador antes de ver la catapulta en pantalla, las rocas son rápidas. **Si el jugador se acerca** lo suficiente como para que la catapulta entre en el área de detección del **jugador la catapulta no podrá disparar** y estará indefensa, (aunque tienen mucha vida).

Disparos

Los disparos van siempre en la dirección **en la que estaba** el jugador cuándo se creó el disparo. Cuando impactan con el jugador quitan una cantidad de vida diferente dependiendo del tipo de disparo. Si colisionan con un tile no pasable o con el jugador son destruidos

Detalles de implementación
Para ello considero **los puntos** formados por las **coordenadas del enemigo**, que **será el origen** y las **coordenadas del jugador** que es **el destino**. Entonces tengo **un vector definido por 2 puntos**. Con este vector puede obtener la **proporción de velocidad que le corresponde a cada eje**. Para esto tengo que despejar x e y en la siguiente ecuación.

$$velocidadBase = \sqrt{x^2 + y^2}$$

Para resolver esta ecuación **necesito otra que me relacione X e Y**. Las distancias X e Y **guardan esta proporción** asique dividiéndolas (grande entre pequeña) y multiplicando el resultado a la variable pequeña ya tengo la segunda ecuación, despejo y ya se una velocidad, la otra será la velocidadBase menos la que obtenemos.

Esto **puede hacerse de forma más sencilla**, dividiendo la distancia pequeña entre la grande (al revés que antes), multiplicando el resultado a la velocidad base ya tendríamos una velocidad, calculando la diferencia entre la velocidad base y la obtenida tendremos la otra. (*Pero lo he hecho paso a paso, porque para una práctica en todo el grado que puede hacer algo así… :D*)
Esta implementación está en DisparoFlecha y DisparoRoca, método inicializar.

Scroll y movimiento.

Controlo el **scroll** en los 2 ejes de la pantalla y pinto solo lo que hay en pantalla. En el método **dibujarTiles** de la clase nivel puede verse la implementación.
Las reglas de movimiento están simplificadas respecto a la práctica de plataformas.

Audio y dibujado.

Uso un **gestor de audio** para reproducir la música de fondo y los efectos de sonido que ocurren al perder vida, atacar o hacer un disparo.
Uso **animaciones diferentes** para todos los enemigos y coleccionables, el arquero solo tenía una animación asique está siempre mirando para el mismo sitio.
Al implementar **las flechas** quería orientar el sprite según la dirección, pero no se me ocurría como así a bote pronto.
Si se pulsa **el botón 1** se activará la vista de debug, que permitirá ver el área de ataque y percepción.

Ampliaciones

El juego lo he hecho para que sea fácil ampliarlo, desde añadir enemigos, coleccionables, música, sprites o añadir funcionalidades al jugador, como que dispare él también cosas.