Documentación de Sky Jump

Pablo Rodríguez Peña 2ºDAM

El juego "Sky Jump"trata de ayudar a Pou a saltar de plataforma en plataforma mientras cae del cielo. La premisa del juego es mantener a Pou en el aire el mayor tiempo posible y acumular la mayor cantidad de puntos posible.

Descripción del Juego:

La misión es guiar a Pou a través de un descenso desde el cielo, saltando de plataforma en plataforma. Utilizando los sensores del dispositivo móvil, los jugadores pueden controlar el movimiento lateral de Pou girando el dispositivo hacia la izquierda o hacia la derecha.

Mecánica del Juego:

Salto y Movimiento: Los jugadores controlan el movimiento lateral de Pou inclinando el dispositivo. Pou salta automáticamente hacia arriba y hacia abajo.

Aterrizaje en Plataformas: Pou gana puntos cada vez que aterriza en una plataforma. Cuanto más tiempo permanezca en una plataforma, más puntos acumulará.

Evitar el Vacío: Si Pou cae al vacío sin aterrizar en una plataforma, el juego termina.

Velocidad y Habilidad: Para tener éxito, los jugadores deben ser rápidos y hábiles, asegurándose de que Pou siempre tenga un lugar donde aterrizar. La velocidad y la precisión son clave para mantener a Pou en movimiento y evitar que caiga.

Puntos y Progresión:

Cada aterrizaje exitoso en una plataforma otorga puntos al jugador.

El objetivo es acumular la mayor cantidad de puntos posible antes de que Pou caiga al vacío.

Con cada nivel alcanzado, la dificultad del juego puede aumentar, presentando desafíos adicionales para los jugadores., como nubes que desaparecen y aviones que se mueven.

Lógica del juego

Clase MainActivity:

Esta clase representa la actividad principal de la aplicación "Sky Jump". Es la primera pantalla que se muestra al iniciar la aplicación y proporciona opciones para comenzar el juego o ver las reglas del mismo.

Atributos:

startButton: Representa el botón utilizado para iniciar el juego.
rulesButton: Representa el botón utilizado para ver las reglas del juego.
Métodos:

onCreate(Bundle savedInstanceState): Este método se llama cuando se crea la actividad. Aquí se inicializan los botones y se configuran los listeners para manejar los eventos de clic en los botones.
startGame(): Método que inicia la actividad del juego cuando se hace clic en el botón de inicio. Crea un Intent para iniciar la actividad GameActivity (anteriormente GameView).
showRules(): Método que muestra las reglas del juego cuando se hace clic en el botón de reglas. Crea un Intent para iniciar la actividad Reglas.

Clase Inicio:

La clase Inicio representa la pantalla de inicio de la aplicación "Sky Jump". Proporciona opciones para iniciar el juego o ver las reglas del mismo.

Métodos:

onCreate(Bundle savedInstanceState): Se llama cuando se crea la actividad. Aquí se configuran los botones para iniciar el juego y ver las reglas. Se establecen listeners para manejar los eventos de clic en los botones.
La clase Inicio es la primera pantalla que se muestra al abrir la aplicación "Sky Jump" y proporciona la navegación inicial para comenzar el juego o acceder a las reglas.

Clase Reglas:

Métodos:

onCreate(Bundle savedInstanceState): Configura la vista y el comportamiento de la actividad.

showRules(): Muestra las reglas del juego en una vista de texto.

Métodos para manejar los eventos de clic en botones.

Clase Game Activity:

La clase GameActivity representa la actividad principal del juego "Sky Jump". Esta actividad se encarga de iniciar la vista del juego (GameView) y de gestionar su ciclo de vida.

Atributos:

gameView: Representa la vista del juego, donde se muestra toda la lógica y los gráficos del juego.

Métodos:

onCreate(Bundle savedInstanceState): Se llama cuando se crea la actividad. Aquí se inicializa la vista del juego (GameView) y se establece como el contenido de la actividad.
onPause(): Se llama cuando la actividad está a punto de ser pausada. Pausa la vista del juego.
onResume(): Se llama cuando la actividad se reanuda desde un estado pausado. Reanuda la vista del juego.
Esta clase es esencial para iniciar y gestionar la vista del juego "Pou Saltarín" dentro de la aplicación.

Clase GameView:

La clase GameView es responsable de controlar la lógica del juego Sky Jump". Aquí se gestiona el ciclo de juego, se dibujan los elementos en la pantalla y se manejan los eventos del sensor para controlar el movimiento del personaje Pou. A continuación, se detalla toda la información relevante referente a esta clase:

Atributos y Constructor:

gameOver: Indica si el juego ha terminado.

isPlaying: Indica si el juego está en curso.

gameThread: Hilo del juego.

surfaceHolder: Holder para manipular la superficie de dibujo.

pou: Objeto que representa al personaje Pou.

sensorManager, accelerometer, gyroscope: Manejador y sensores del dispositivo.

screenWidth, screenHeight: Dimensiones de la pantalla.

platforms: Lista de plataformas en el juego.

random: Generador de números aleatorios.

jumpSound, gameOverSound, levelSound, backgroundMusic: Reproductores de sonido y música.

backgroundImage1, backgroundImage2, backgroundImage3, backgroundImage4: Imágenes de fondo del juego.

backgroundCounter, score: Contadores de pantalla y puntuación.

scoreTextView: Vista de texto para mostrar la puntuación.

Métodos de Inicialización:

init(Context context): Inicializa los atributos y carga los recursos del juego.

updateScoreTextView(): Actualiza la vista de texto de la puntuación.

startSensor(): Inicia la lectura de los sensores.

stopSensor(): Detiene la lectura de los sensores.

createPlatforms(): Crea las plataformas iniciales del juego.

generateNewPlatforms(): Genera nuevas plataformas cuando Pou alcanza el borde superior de la pantalla.

incrementBackgroundCounter(): Incrementa el contador de pantalla.

playJumpSound(), playBackgroundMusic(), stopBackgroundMusic(), playGameOverSound(), playLevelSound(): Métodos para controlar la reproducción de sonidos y música.

Métodos Principales del Juego:

run(), update(), draw(), control(): Controlan el ciclo de juego.

pause(), resume(): Pausan y reanudan el juego.

onSensorChanged(SensorEvent event), onAccuracyChanged(Sensor sensor, int accuracy): Manejan los eventos del sensor.

Métodos de Dibujo y Control de Plataformas:

vectorToBitmap(Drawable vectorDrawable), rotateVectorDrawable(Drawable vectorDrawable, float degrees), getRotatedAirplaneBitmap(): Transforman y rotan imágenes.

drawPlatforms(Canvas canvas): Dibuja las plataformas en el lienzo.

drawScoreTextView(Canvas canvas): Dibuja la vista de texto de la puntuación.

drawBackground(Canvas canvas): Dibuja el fondo del juego.

La clase GameView también maneja la detección de eventos de los sensores del dispositivo, lo que permite controlar el movimiento del personaje Pou utilizando el acelerómetro y el giroscopio. A continuación, se detalla la parte del código relacionada con los sensores:

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float xAcceleration = event.values[0];
        float yAcceleration = event.values[1];
        double inclinacion = Math.atan2(-xAcceleration, yAcceleration) * 180 / Math.PI;
```

```
        float velocidadHorizontal = 10;

        pou.setSpeedX(velocidadHorizontal *
    (float)Math.sin(Math.toRadians(inclinacion)));

      }

    }
```

En el método onSensorChanged(SensorEvent event), se verifica si el evento corresponde al sensor de acelerómetro. Luego, se obtienen los valores de aceleración en los ejes x e y del dispositivo. Utilizando estos valores, se calcula la inclinación del dispositivo con respecto a la vertical. La velocidad horizontal de Pou se establece en función de esta inclinación, lo que permite controlar el movimiento lateral del personaje. Por último, se actualiza la velocidad horizontal de Pou utilizando el método setSpeedX().

Clase Plataforma:

Atributos y Constructor:

rect: Representa el área rectangular de la plataforma.

passable: Indica si la plataforma es atravesable.

moving: Indica si la plataforma se mueve horizontalmente.

speedX: Velocidad horizontal de la plataforma.

screenWidth, screenCounter: Dimensiones de la pantalla y contador de pantalla.

Métodos:

draw(Canvas canvas): Dibuja la plataforma en el lienzo.

updateHorizontalPosition(): Actualiza la posición horizontal de la plataforma.

Métodos getters y setters para acceder y modificar los atributos de la plataforma.

Clase Pou:

Atributos y Constructor:

bitmap: Representa la imagen de Pou.

x, y: Posición de Pou en la pantalla.

screenWidth, screenHeight: Dimensiones de la pantalla.

width, height: Dimensiones de la imagen de Pou.

speedX, speedY: Velocidades horizontal y vertical de Pou.

isJumping: Indica si Pou está saltando.

Métodos:

update(), moverDesdeElCentro(): Actualizan y manejan la posición y movimiento de Pou.

draw(Canvas canvas): Dibuja la imagen de Pou en el lienzo.

Métodos getters y setters para acceder y modificar los atributos de Pou.

Clase GameOver:

Esta clase se encarga de mostrar la pantalla de fin de juego cuando la partida termina. Proporciona opciones para que el jugador pueda reiniciar el juego o salir de la aplicación.

Atributos y Constructor:

No tiene atributos específicos.

El constructor puede inicializar algunos elementos necesarios para mostrar la pantalla de fin de juego, como botones para reiniciar o salir del juego.

Métodos:

showGameOverScreen(): Este método muestra la pantalla de fin de juego en la interfaz de usuario. Puede incluir elementos como mensajes de puntuación final, botones para reiniciar el juego o salir, y cualquier otra información relevante para el jugador.

restartGame(): Método para reiniciar el juego cuando el jugador elige esta opción desde la pantalla de fin de juego. Restaura todas las variables del juego a sus valores iniciales y vuelve a mostrar la pantalla de juego principal.

exitGame(): Método para salir del juego cuando el jugador decide abandonarlo desde la pantalla de fin de juego. Cierra la aplicación y devuelve al jugador al menú principal del dispositivo.