Sistema de coordenadas en LibGDX

LibGDX utiliza varios sistemas de coordenadas dependiendo del contexto (pantalla, cámara, mundo del juego, entrada táctil, física, etc.). Comprender cómo se relacionan es esencial para posicionar, renderizar y detectar colisiones correctamente.

1. Coordenadas de pantalla (Screen Coordinates)

Son las coordenadas proporcionadas por el sistema operativo o por LibGDX para la ventana o pantalla física del dispositivo.

Características

- El origen **(0,0)** está en la **esquina superior izquierda**.
- El eje **X aumenta hacia la derecha**.
- El eje **Y aumenta hacia abajo**.
- Se usan principalmente para:
    - Entrada táctil o de ratón (`Gdx.input.getX()`, `getY()`)
    - cálculos relacionados con el tamaño real de la ventana (`Gdx.graphics.getWidth()`)

Ejemplo

Para obtener la posición del cursor:

```
int x = Gdx.input.getX();

int y = Gdx.input.getY();
```

2. Coordenadas del mundo (World Coordinates)

Representan la posición de los objetos dentro del espacio lógico del juego. Son completamente independientes de la resolución real de la pantalla.

Características

- El origen depende del tipo de cámara o viewport.
- Las unidades son definidas por el desarrollador (píxeles lógicos, metros, tiles, etc.).
- Se usan para posicionar sprites, entidades, UI no escalada y cámaras.

Ejemplo

```
sprite.setPosition(10, 5);  // Coordenadas dentro del mundo del juego
```

3. Coordenadas de la cámara (Camera Coordinates)

La cámara transforma las coordenadas del mundo para que se adapten a la pantalla. LibGDX utiliza normalmente **OrthographicCamera** en 2D.

Características principales

- La cámara define qué parte del mundo es visible.
- Su posición determina el origen lógico que se ve en pantalla.
- Permite aplicar zoom, desplazamiento (pan), rotaciones.

Transformación típica

```
camera.update();

batch.setProjectionMatrix(camera.combined);
```

4. Coordenadas del Viewport

Un viewport adapta las coordenadas del mundo para ajustarlas al tamaño real de la pantalla física.

Función

Resolver problemas como:

- Resoluciones distintas,
- Pantallas panorámicas,
- Escalado uniforme,
- Bandas negras opcionales ("letterboxing").

Tipos principales de viewport

Viewport	Comportamiento	Uso típico
FitViewport	Mantiene la relación de aspecto, con bandas negras	Juegos 2D
FillViewport	Rellena toda la pantalla recortando contenido	Apps visuales
StretchViewport	Deforma para llenar pantalla	Prototipos
ExtendViewport	Similar a Fit pero agrega área extra	Juegos con cámaras amplias

5. Conversión entre sistemas de coordenadas

LibGDX permite convertir fácilmente entre coordenadas de entrada (pantalla) y coordenadas lógicas (mundo).

De pantalla a mundo

Usando cámara:

```
Vector3 coords = new Vector3(Gdx.input.getX(), Gdx.input.getY(), 0);

camera.unproject(coords);
```

Con viewport:

```
Vector2 worldCoords = viewport.unproject(new Vector2(screenX,
screenY));
```

De mundo a pantalla

```
Vector3 screen = camera.project(new Vector3(worldX, worldY, 0));
```

---

6. Coordenadas de entrada (Input Coordinates)

- LibGDX devuelve la posición del input en coordenadas de pantalla, y con el eje Y invertido respecto al mundo.
- Debe transformarse en coordenadas del mundo antes de usarlas en lógica de juego.

Ejemplo

```
int screenX = Gdx.input.getX();

int screenY = Gdx.input.getY();


Vector2 world = viewport.unproject(new Vector2(screenX, screenY));
```