

LIBGDX Anexo Consellos de programacion

UNIDADE 2: Consellos de programación

Uso de new's

Sempre que se poida debemos evitar o uso de **new's** dentro de calquera método que poida dar lugar a un número de chamadas elevado, sobre todo no método render.

Para evitalo só temos que crear a propiedade no constructor da clase e cambiarlle o seu valor en vez de asinarlle un novo valor.

Un exemplo:

```
public class Exemplo{

    /** Definimos o vector temporal **/
    private Vector3 temporal;

    public Exemplo() {

        temporal = new Vector3();    // Instanciamos unha vez...
    }

    public void render(float delta){
        temporal.set(x+delta*2,10,0);    // Dámoslle valores sen instanciar no método render
    }
}
```

Uso de cadeas

Debemos evitar o uso de cadeas concatenadas dentro de procedementos que estean no método render, xa que ó facelo estamos a crear novas cadeas dándolle un novo espazo de memoria. Cada vez que facemos un + entre dúas cadeas se crea un novo obxecto String que despois ten que eliminar o **garbage collector**.

Para solucionalo podemos facer uso da **clase StringBuilder**.

```
private StringBuilder stringbuilder;
stringbuilder = new StringBuilder(20);    // Num. de caracteres

stringbuilder.setLength(0);                // Borra o texto
stringbuilder.append("Engade cadea");    // Engade unha nova cadea
stringbuilder.delete(0,2);    // Borra os 3 primeiros caracteres
```

Nota: Tamén poderíamos facer uso da **clase StringBuffer**. Esta clase é menos rápida que StringBuilder, xa que é sincronizada. Quere isto dicir que cando temos varios fíos de execución que acceden ós mesmos datos de cadea, deberíamos usar a clase StringBuffer. No resto de casos é mellor StringBuilder.

Podedes consultar [este enlace](#) para ver as diferenzas.

