1 LIBGDX Anexo tipos de datos

1.1 Sumario

1.1.1 Introducción

En LIBGDX imos ter varios tipos de datos específicos para manexar os gráficos.

Neste punto imos analizar algúns dos empregados, tanto propios de LIBGDX como de JAVA.

1.1.2 Vector2

- **Clase Vector2**: garda dúas variables de tipo float. Empregado sobre todo nos xogos 2D. O primeiro parámetro é a coordenada ou valor x, e o segundo parámetro é a coordenada ou valor y.

Cando o creamos podemos:

- Facer un new indicando os dous valores:

```
Vector2 vector = new Vector2(10,10);
```

- Asinarlle despois os valores:

```
Vector2 vector = new Vector2();
vector.set(10,10);
```

A clase Vector2 ten multitude de métodos (os podedes consultar no enlace indicado arriba, no nome da clase). É moi importante comprender que ditos método afectan ó valor do vector orixinal. Explícome.

```
Vector2 vector = new Vector2(10,10);
            Vector2 vectorDevuelto = vector.add(5, 6);
```

O método **add** aplicado ó vector engade o número indicado (5,5) a cada variable x,y. Así x pasará a valer 15 e y pasará a valer 16. O método add devolve un punteiro ó mesmo vector. Quere isto dicir que vectorDevuelto é vector.

1.1.3 Rectangle

Clase Rectangle: Permite gardar unha posición e un tamaño (define un rectángulo).

A posición inicial (a definida pola coordenada x,y) é a esquina inferior esquerda.

```
private rectangulo = new Rectangle(x,y,width,height);
```

```
private rectangulo = new Rectangle(10,21,100,100);
```

O utilizamos para definir o tamaño dos controis no noso xogo. Poderíamos usalo en vez de gardar a posición nun Vector2 e o tamaño noutro Vector2.

1.1.4 Circle

Clase Circle: Define un círculo. Entre os datos a enviar necesitamos indicarlle unha posición (coordenada x/y) e un tamaño (radio).

```
private circulo = new Circle(x,y,radio);
```

```
private circulo = new Circle(10,30,5);
```

O utilizamos na clase Intersector para saber se ó premer na pantalla (o representamos cun dedo) tocamos algún dos controis do xogo.

1.1.5 Array

Clase Array: utilizada para gardar un conxunto de obxectos.

Métodos máis importantes:

- **Método public void add(T value)**: engade un novo elemento ó array.
- **Método contains(T value, boolean identity)**: informa se o elemento está no array.
- **public T get(int index)**: devolve o elemento do array indicado por index.
- **Método removeValue(T value, boolean identity)**: elimina un elemento do array.
- **items**: accede ós elementos do array

Para usala:

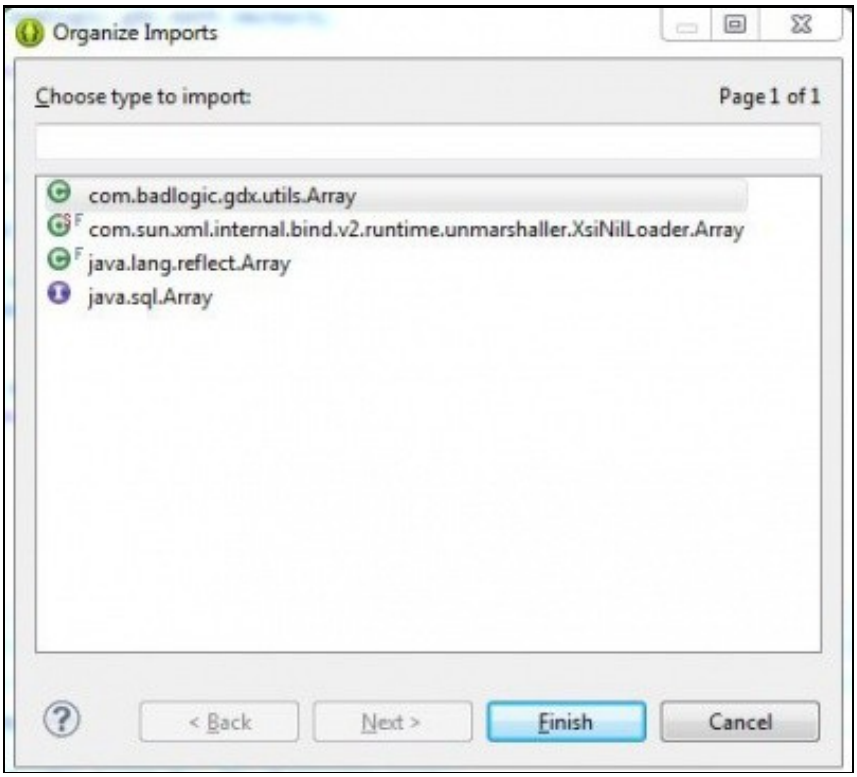
- Definir un obxecto desta clase indicando o tipo de clase que vai albergar (isto se indica cos caracteres < e >) da seguinte maneira:

```
private Array<Bolboreta> bolboretas;
```

- Instanciamos o obxecto no constructor da clase:

```
bolboretas = new Array<Bolboreta>();
```

Nota: Ó utilizar a clase Array vai dar un erro xa que non está importada. Cando a importedes (control +shift+O) vos dará varias opcións:



Deberedes de escoller **com.badlogic.gdx.utils.Array**.

- **Engadir novos obxectos:**

Debemos de usar o método add.

```
bolboretas.add(new Bolboreta());
```

- **Recorrendo o array:**

Para percorrer un array podemos usar:

◊ A través do método Iterator

Exemplo da wiki: https://github.com/libgdx/libgdx/wiki/A-simple-game (buscar por Iterator)

Outro exemplo:

```
Iterator <Bolboreta> iter = bolboretas.iterator();
while(iter.hasNext()){
  Bolboreta bolboreta = iter.next();
}
```

◊ A través dun for da forma:

```
for (Bolboreta bolboreta : bolboretas){
  // bolboreta é cada un dos obxectos que están no array bolboretas.
}
```

- **Borrando un elemento do array:**

◊ Método removeValue(T value, boolean identity)

◊ Parámetro T value: é o obxecto a borrar.
◊ Parámetro boolean identity: se é true só compara os valores das propiedades do obxecto cos do array. Se é false compara que sexa o mesmo obxecto. Normalmente poñeremos true.

Exemplo de uso:

```
bolboretas.removeValue(candela, true);
```

Este método está sobrecargado polo que tamén podemos borrar por un índice,....

Por exemplo:

- Método public T removeIndex(int index)

Devolve o obxecto eliminado do array polo índice indicado.

- **Obtendo o número de elementos do array:**

  - **items**: accede ós elementos do array.

A través de dito array podemos obter o número de obxectos que temos no array da seguinte forma:

```
int num = bolboretas.items.length;
```

1.1.6 Enum

Documentación oficial: http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

Serve para definir unha variable que vai ter un valor dunha serie de valores predefinidos.

Por exemplo:

```
public static enum TIPOS_ELEMENTOS {COCHE, AUTOBUS, TRONCO, ROCA};
```

1.1.7 HashMap

Documentación oficial: http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

Permite definir un mapa que ven a ser un obxecto que 'mapea' pares de clave-valor.

Cando definimos un HashMap indicamos o tipo de dato que vai gardar e o valor asociado a ese dito tipo de dato.

Por exemplo:

```
public enum Keys {
ESQUERDA,DEREITA,ARRIBA,ABAIXO
}
HashMap<Keys, Boolean> keys = new HashMap<Keys, Boolean>();
```

Neste exemplo estamos a definir un obxecto mapa (keys) que ten como claves o tipo de datos enum Keys, quero isto dicir que imos gardar as claves ESQUERDA, DEREITA,.... e cada unha deles vai poder ter como valor true/false. Por exemplo:

```
ESQUERDA => false
DEREITA  => true
......
```