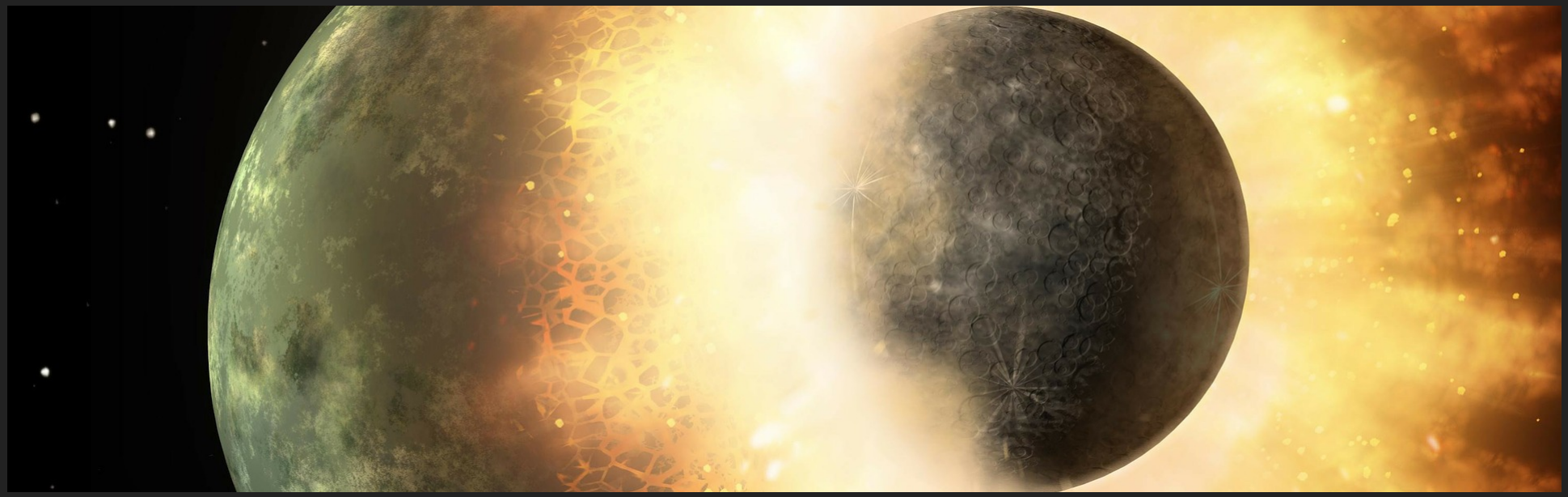

DETECCIÓN Y MANEJO DE COLISIONES

APRENDE A PROGRAMAR TUS PROPIOS VIDEOJUEGOS

BOOTCAMP UTN 2019

FUNCIONES PARA DETECCIÓN DE COLISIONES

- FlxG.overlap()
- FlxG.collide()

FUNCIONES PARA DETECCIÓN DE COLISIONES

Ambas hacen lo mismo, *collide()* cuando detecta colisión, además invoca a *FlxObject.separate()* para separar los objetos

UTILIZACIÓN

```
if(FlxG.overlap(hero, enemy)){
  // escribir aquí el código que
  // se ejecutará cuando haya colision
}
```

Las funciones devuelven true o false en caso de que haya colisión

UTILIZACIÓN

```
FlxG.overlap(hero, enemy, onCollision);

function onCollision(obj1: FlxObject, obj2: FlxObject){
  // escribir aquí el código que
  // se ejecutará cuando haya colision
}
```

Las función callback *onCollision()* se dispara en caso de que haya una colisión

Recibe como parámetro los objetos que colisionaron

GRUPOS

¿Qué hacemos cuando queremos detectar colisiones entre muchos objetos?

Tanto *overlap()* como *collide()* permiten pasar grupos como argumentos

GRUPOS

```
var walls = new FlxGroup(); // crear grupo
walls.add(wall1);           // agregar entidades al grupo
walls.add(wall2);
walls.add(wall3);
add(walls);                 // agregar grupo a la escena
```

GRUPOS

```
FlxG.overlap(hero, walls, onHeroWalls);

function onHeroWalls(obj1: FlxObject, obj2: FlxObject){
  // escribir aquí el código que
  // se ejecutará cuando haya colision
}
```

En el parámetro *obj2* la función **NO RECIBIRÁ AL GRUPO** sino al objeto del grupo con el que hubo colisión.

Si hay colisión simulánea con más de un objeto, la función se llama por cada objeto

GRUPOS

- Se debe armar grupos para buscar minimizar la cantidad de llamadas a *overlap()*/*collide()*
- Los grupos pueden contener otros grupos.
- Los callbacks siempre recibirán los objetos que colisionaron (no se puede colisionar contra un grupo, si con un objeto del mismo)

GRUPOS

- Un objeto puede estar en más de un grupo a la vez
- No todos los grupos tienen que ser agregados a la escena (algunos solo se arman para simplificar la detección de colisiones)
- Cuidado de no agregar un objeto a la escena más de una vez

OPERAR SOBRE UN GRUPO

```
var group = new FlxGroup();

group.forEach(function(obj: FlxBasic){
  var enemy = cast(obj, Enemy);
  enemy.getDamagePoints();
})
```

El método *forEach()* recibe como argumento una función, ejecuta dicha función pasándole cada elemento del grupo

OPERAR SOBRE UN GRUPO

La función recibe un objeto de tipo **FlxBasic** (así es como almacena **FlxGroup** los objetos)

Por eso es necesario hacer una conversión (cast) al tipo real del objeto que almacenamos en el grupo (en éste caso la función *getDamagePoints()* está en **Enemy** no en **FlxBasic**)

GRUPOS TIPADOS

```
var enemyGroup = new FlxTypedGroup<Enemy>();

enemyGroup.forEach(function(e: Enemy){
  e.getDamagePoints();
})
```

Los grupos tipados **sólo almacenan elementos del tipo especificado**

No pueden almacenar otros grupos u otra cosa (menor posibilidad de errores)

No hace falta hacer cast() (ya sabemos qué es lo que guardan)

MANEJO DE COLISIONES CON GRUPOS

Cuando en un grupo existen objetos de clases diferentes se puede preguntar por su tipo (con **Type**.*getClass()*) para decidir qué hacer

MANEJO DE COLISIONES CON GRUPOS

```
FlxG.overlap(hero, heroCollideables, onHeroCollide);

function onHeroCollide(hero: Hero, object: FlxObject){
  if(Type.getClass(object) == Wall){
    FlxObject.separate(hero, object);
  }else if(Type.getClass(object) == Item){
    var item: Item = cast(object, Item);
    points = points + item.getPoints();
    item.kill();
  }
}
```

MANEJO DE COLISIONES CON GRUPOS

Si sabemos el tipo del objeto que vamos a recibir podemos asignar dicho tipo al parámetro (por ejemplo, hero es de la clase **Hero**)

Si no sabemos (porque pueden ser varias cosas), utilizamos algún tipo básico (**FlxBasic** o **FlxObject**) y hacemos la conversión al tipo verdadero con *cast()*

PROPIEDADES DE LOS OBJETOS

- **width**: ancho del hitbox
- **width**: alto del hitbox
- **offset**: desplazamiento del imagen respecto del hitbox
- **allowCollisions**: desde cuáles lados permitir la colisión
- **immovable**: si el objeto es o no estático (no se mueve)

DEPURADOR VISUAL INCORPORADO

- Compilar en modo *debug*
- Activar/Desactivar con F2