

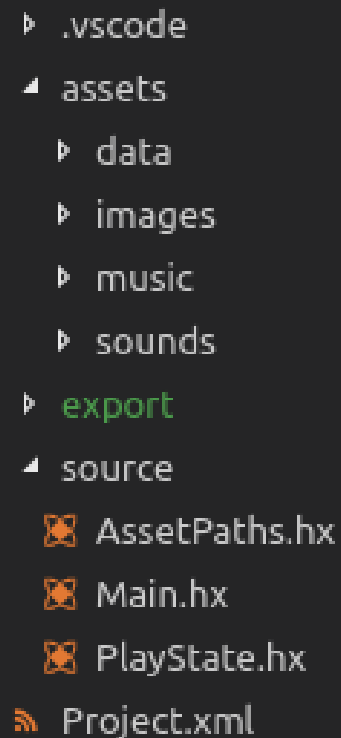


ESTRUCTURA DE UN PROYECTO EN HAXEFLIXEL

APRENDE A PROGRAMAR TUS PROPIOS VIDEOJUEGOS

BOOTCAMP UTN 2019

ESTRUCTURA DE UN PROYECTO EN HAXEFLIXEL



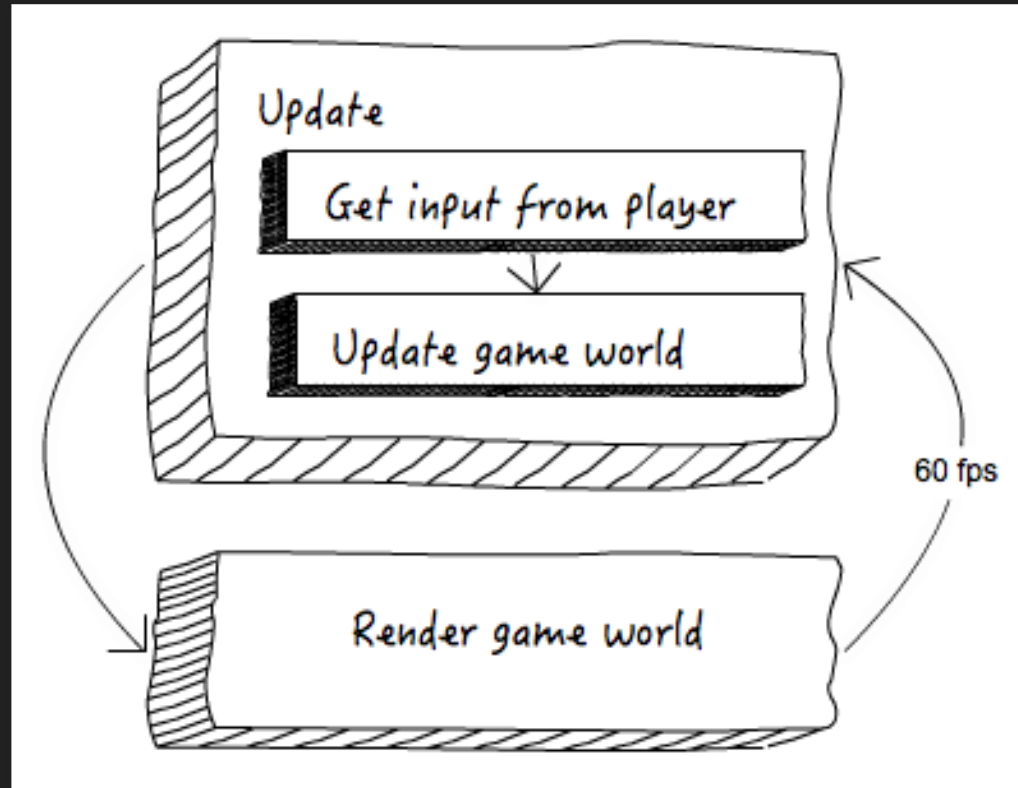
```
└─ .vscode
└─ assets
    └─ data
    └─ images
    └─ music
    └─ sounds
└─ export
└─ source
    └─ AssetPaths.hx
    └─ Main.hx
    └─ PlayState.hx
└─ Project.xml
```

The image shows a file explorer window with a dark background. It displays the directory structure of a HaxeFlixel project. The root directory contains several folders and files. The 'assets' folder is expanded, showing subfolders for 'data', 'images', 'music', and 'sounds'. The 'export' folder is highlighted in green. The 'source' folder is also expanded, showing three Haxe source files: 'AssetPaths.hx', 'Main.hx', and 'PlayState.hx'. At the bottom, the 'Project.xml' file is listed with an XML icon.

ESTRUCTURA DE UN PROYECTO EN HAXEFLIXEL

- **assets:** recursos del juego (imagenes, sonidos, etc.)
- **export:** ejecutables generados
- **source:** fuentes
- **Project.xml:** descriptor del proyecto (configuración gral.)

BUCLE DE JUEGO



ESCENAS

Nuestro juego, por ahora, va a tener **una única escena**
(archivo PlayState.hx)

```
import flixel.FlxState;

class PlayState extends FlxState
{
    override public function create():Void
    {
        super.create();
    }

    override public function update(elapsed:Float):Void
    {
        super.update(elapsed);
    }
}
```

DEFINIR UNA ESCENA EN HAXEFLIXEL

- Crear una clase que herede de **FlxState**
- Redefinir (por lo menos) los métodos **create()** y **update()**

CREATE()

- Aquí hay que escribir el código para inicializar la escena
 - Crear y posicionar actores
 - Cargar el nivel
 - Crear items y enemigos
 - etc...
- La función se ejecuta **una sola vez** al crear la escena

UPDATE()

- Es el **corazón** del juego, acá hay que escribir el código para actualizar la escena en cada paso
 - Mover personajes
 - Responder ante la entrada del usuario
 - Ver qué cosas chocan con qué otras
 - Aplicar reglas de juego
- La función se llama "**60 veces/segundo**"
- Recibe como parámetro la cantidad de tiempo que pasó desde la última actualización

REFERENCIA DEL LENGUAJE HAXE

- **import**: sirve para importar módulos/clases
- **class**: para declarar una clase
- **extends**: indica que una clase hereda de otra (PlayState hereda de FlxState)
- **override**: indica que la función ya estaba en la clase padre y la estamos sobrescribiendo
- **public/private**: indica si la variable/función se puede acceder desde afuera de la clase
- Algunos de los tipos primitivos: **Int, Float, Void, String**
- **super**: es para invocar al método redefinido en la clase padre