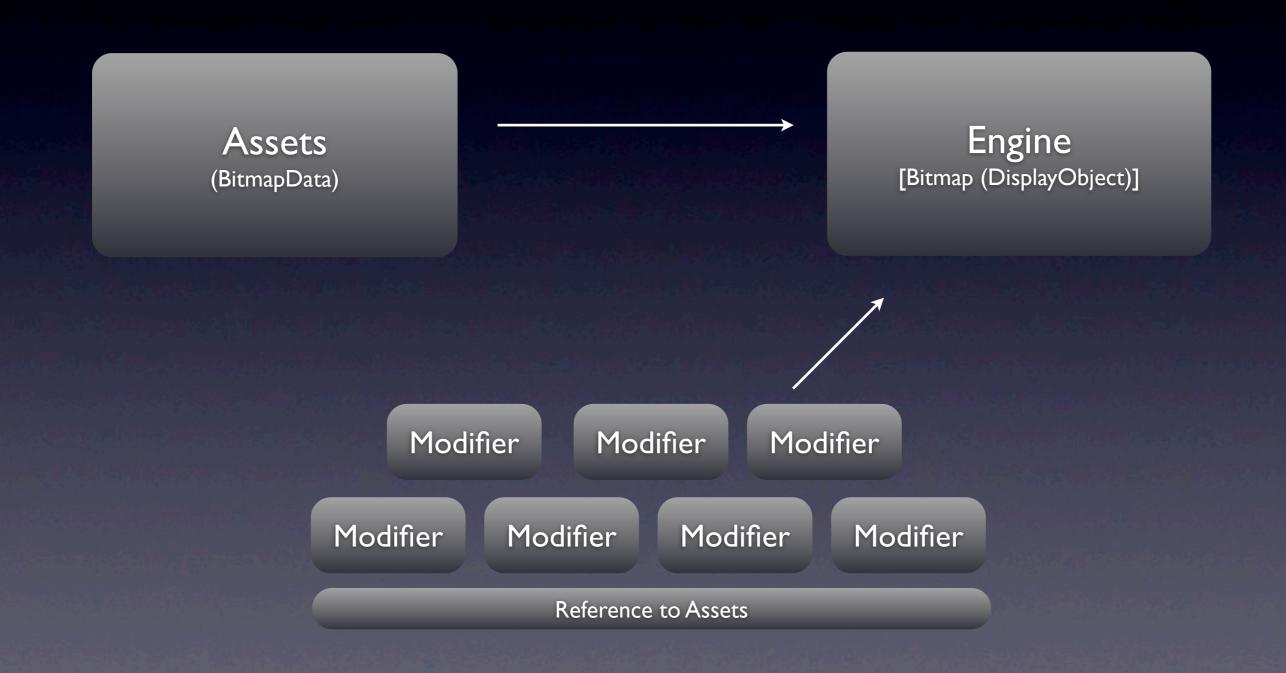
EvoTinyEngine

EvoFlash 2010 / jac

Initialize

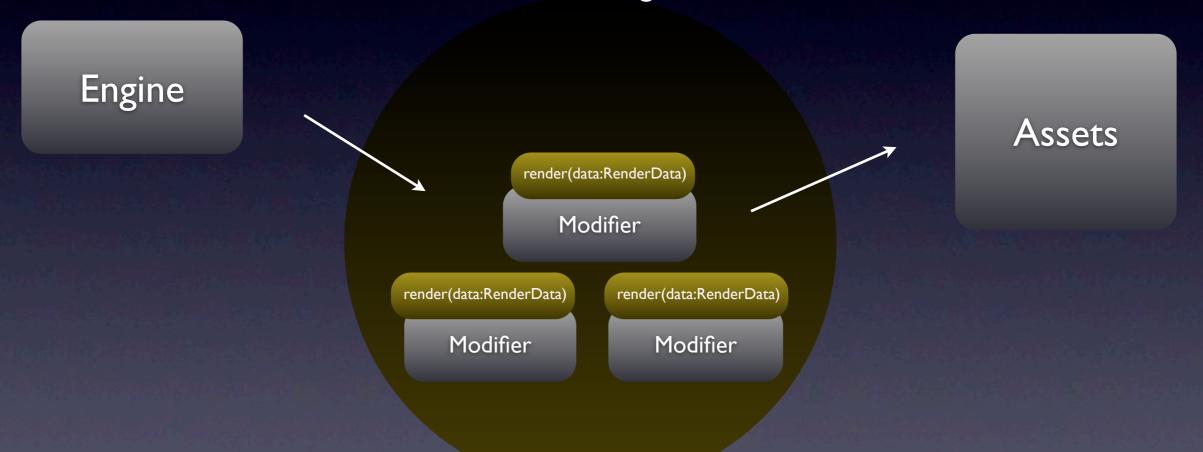
- Create Assets
- Create Engine and give Assets to it
- Add Modifiers to Engine
- Play.

Initializing



Rendering

Engine calls Modifiers render method according to beat.



Modifiers draw BitmapData. (located at Assets, displayed at Engine)
Modifiers have access to Assets (3d engine, sound, put everything there)

Assets

Assets holds by default the reference to BitmapData of the Engine.

Place here all textures, 3D engines.. etc

Public's of Assets

public var tinyengine:ITinyEngine; public var width:int; public var height:int; public var bitmapdata:BitmapData; public var tickTime:int; public var sound:Sound; public var channel:SoundChannel;

Modifiers

Modifiers edit the BitmapData of Engine. All modifiers have access to Assets class.

Modifier Types (evoTinyEngine.modifier.ModifierType)

PREPROCESS - render first

EFFECT - render second

POSTPROCESS - render third

OVERLAY - render last

Has variables

end16thNote:int; start16thNote:int; duration:int; // duration in 16thNotes durationTime:Number; // in ms startTime:Number; // in ms endTime:Number; // in ms id:String; type:String; initialize()

dispose()

render()

soundHit(event:SoundHitEvent)

remove()

toString()

RenderData

Class sent from Engine. Hold's information for rendering.

RenderData

TinyEngine

Is the DisplayObject of a demo.

Manages position of the demo and send rendering calls render methods of Modifiers.

addModifier(modifier:IModifier, start I 6thNote:int, end I 6thNote:int):IModifier

play(start | 6th Note:int = 0, loops:int = 0):void

pause():void

reset():void

remove():void

Public varibles

tick:int // Current 16thNote volume:Number // volume 0-1 soundSynchroniser:SoundSynchroniser

SOUND HIT IMPLEMENTATION:

Use the engine.soundSynchroniser.'s methods to add hits at right beat.

addHit(hit:SoundSyncHit):void

addHitList(list:Vector.<SoundSyncHit>):void

addHitPairPosValue(list:Array):void

addHitPairPosSndHit(list:Array):void

SoundHitEvent

Event to be sent from SoundSynchroniser.

Contains parameters a:int, b:Number, c:Number or d:Number.

TinyEngine

TinyEngine controls the durations and rendering order of Modifiers.
Initializes and disposes Modifiers.
Based on either time or sound duration.

addModifier(modifier: IModifier, start I 6thNote: int, end I 6thNote: int)

```
* start | 6tNote:int // start of Modifier
```

* end16thNote:int // end of Modifier

Keep it simple stupid

- Modifiers edit the BitmapData during their lifetime. After that they are disposed.
- Every modifier have access to Assets so the memory usage is optimized. For example need only one 3D instance.
- Modifier can be an effect, part, preprocess, postprocess, overlay or what ever.

Hello World

(simple as it can be)

```
assets = new Assets(720, 405);
engine = new TinyEngine(assets);
engine.addModifier(new ModifierPixels(assets), 0, 64);
engine.addModifier(new ModifierMandel(assets), 64, 256);
engine.addModifier(new ModifierBloom(assets), 256, 512);
addChild(engine);
engine.play();
```