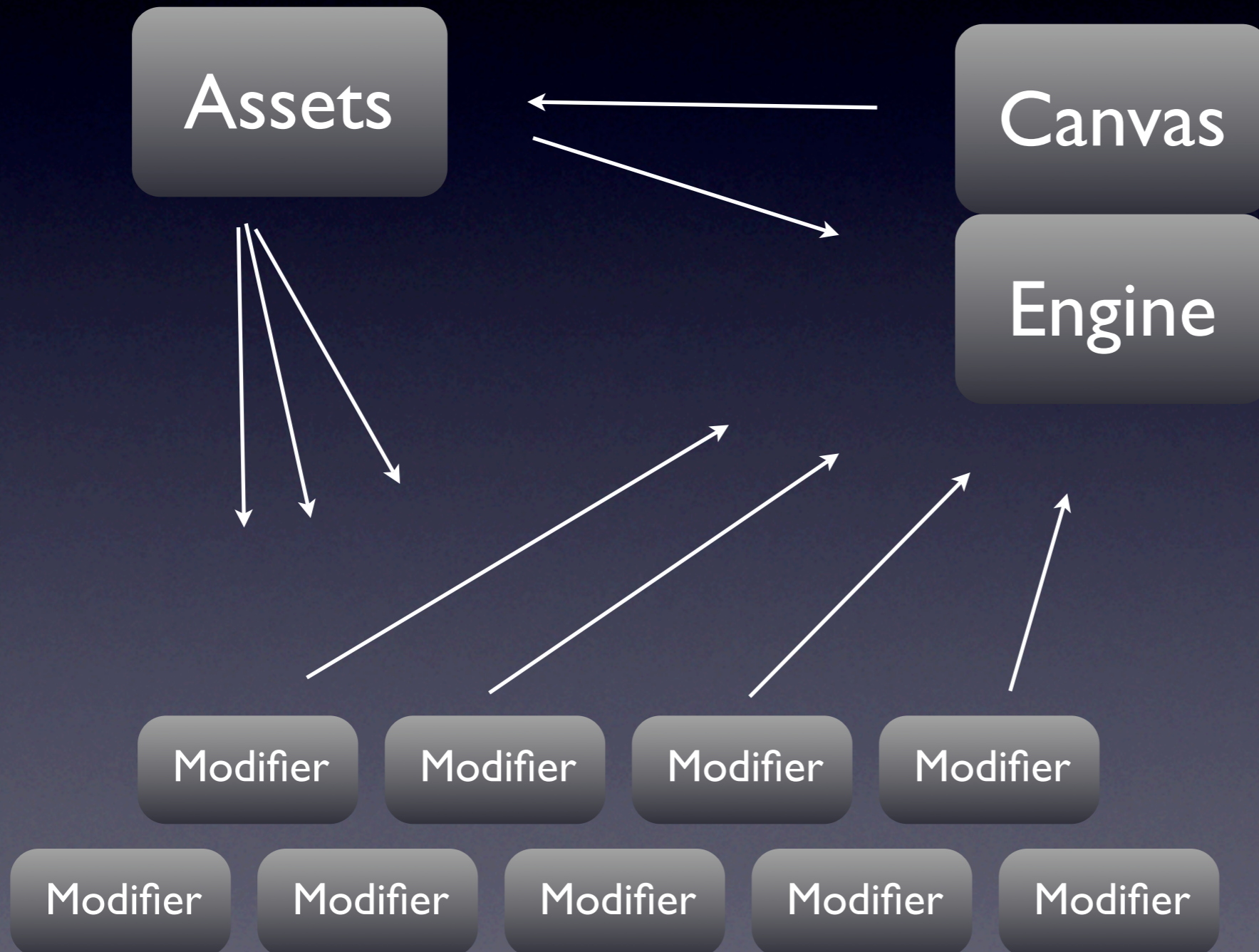


# EvoTinyEngine (4k)

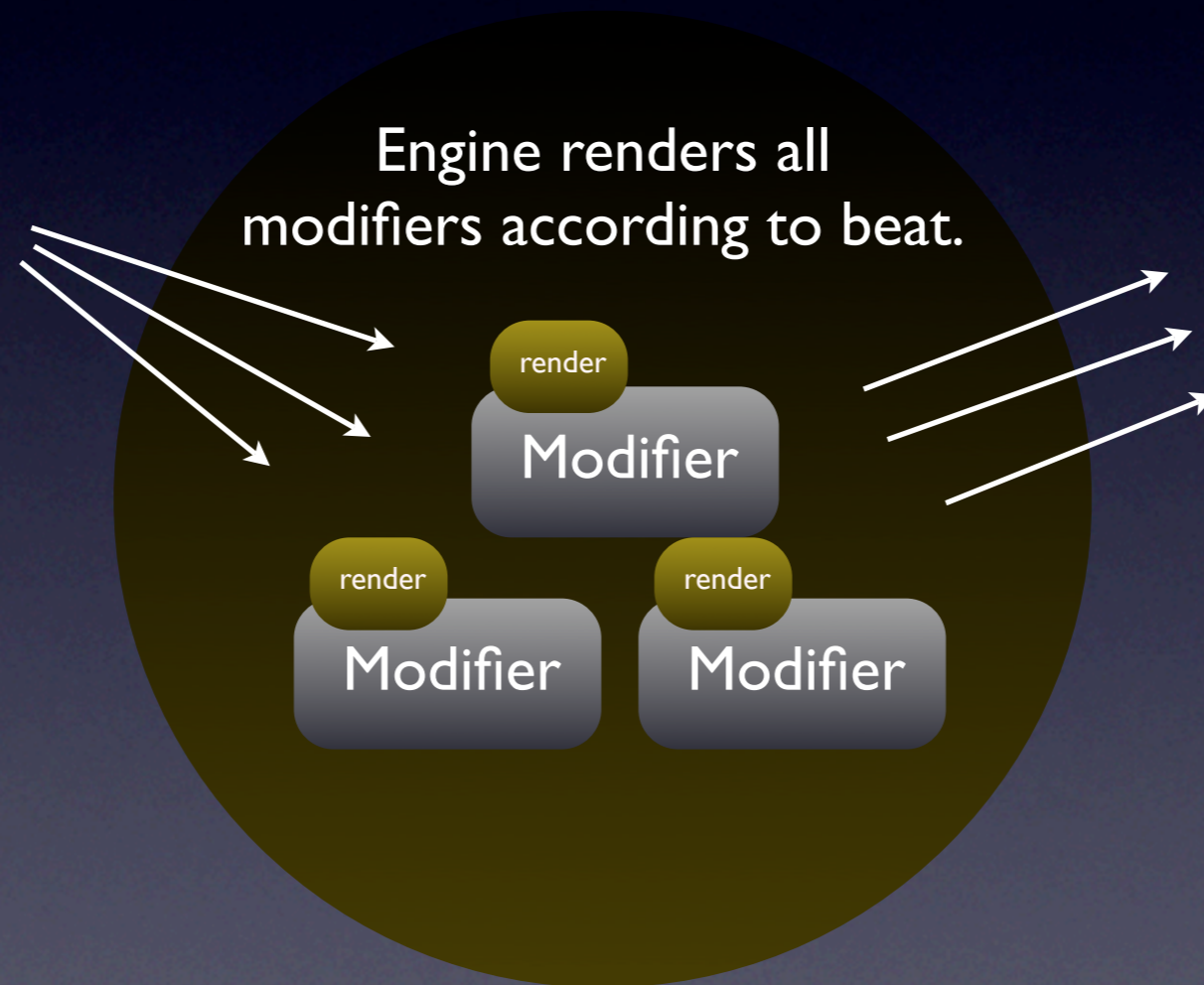
EvoFlash 2009 / jac

# Initializing



# Rendering

Engine



canvas @ engine

Assets

# Assets

Assets holds by default the reference to  
BitmapData of Engine.

Place here all textures, 3D engines.. etc

## Assets

```
* bitmapdata:BitmapData // main bitmapData ("Canvas")  
* width:int // width of "Canvas"  
* height:int // height of "Canvas"
```

# Modifiers

Modifiers edit the “Canvas”.  
All modifiers have access to Assets class.

initialize

dispose

render(RenderData)

## Modifier

```
* assets:AbstractAssets // reference to demo's Assets class
* bit:BitmapData // main bitmapData ("Canvas")
* w:int // width of "Canvas"
* h:int // height of "Canvas"
* ws:int // devided width of "Canvas"
* hs:int // devided height of "Canvas"
* start:int // start time of this Modifier
* end:int // end time of this Modifier
* duration:int // duration of this Modiefier
```

## Modifier Types (ModifierType -class)

PREPROCESS - render  
first

EFFECT - render second

POSTPROCESS - render  
third

# RenderData

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

## RenderData

```
* time:int          // Passed time from beginning
* deltaTime:int    // deltaTime, time from last render
* tickCount:int    // current tick (beat)
* sndValue:int     // sound hit value
```

# TinyEngine

Is the DisplayObject of a demo and give rendering commands.

addModifier

volume

play(beat:int)

pause

soundHit(event:SoundHitEvent)

## TinyEngine

Manages position of the demo and send rendering commands to Modifiers.

### SOUND IMPLEMENTATION:

Add SoundHit listener to soundHit-method.  
`addEventListener(SoundHitEvent.EVENT_HIT, engine.soundHit);`

## SoundHitEvent

Event to be sent from SoundSynth. Place the sound hit type value.

```
dispatchEvent(new  
SoundHitEvent(value,  
SoundHitEvent.EVENT_HIT))
```

# TinyEngine

TinyEngine controls the durations and rendering order of Modifiers.  
Initializes and disposes Modifiers.

```
addModifier(modifier:IModifier, start16thNote:int, end16thNote:int)
```

```
* start16thNote:int           // start of Modifier  
* end16thNote:int             // end of Modifier
```



# Keep it simple stupid

- Modifiers edit the canvas 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 or what ever.

# Hello World

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