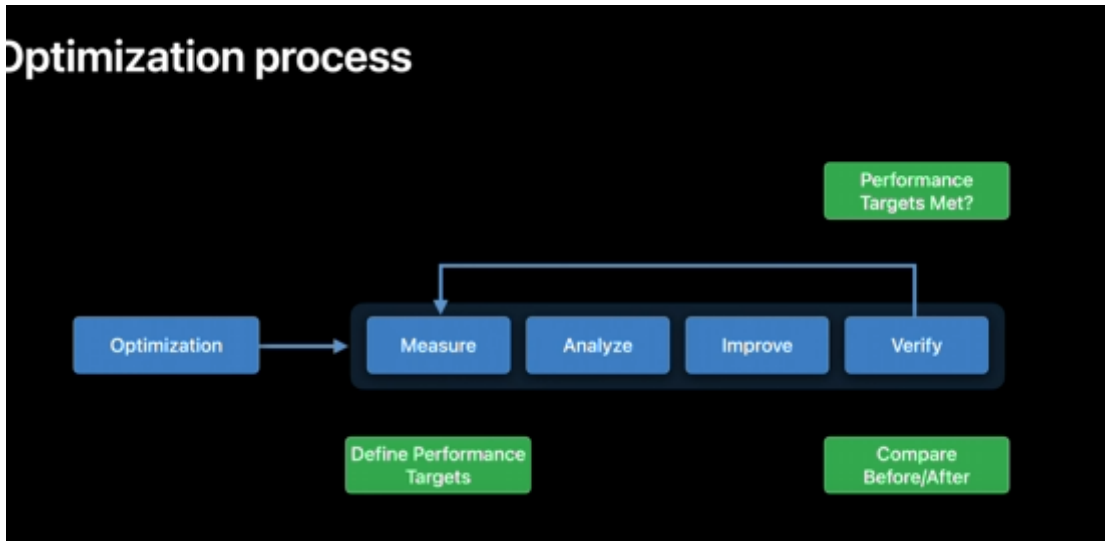


# Optimize high-end games for Apple GPUs

## Optimization Process



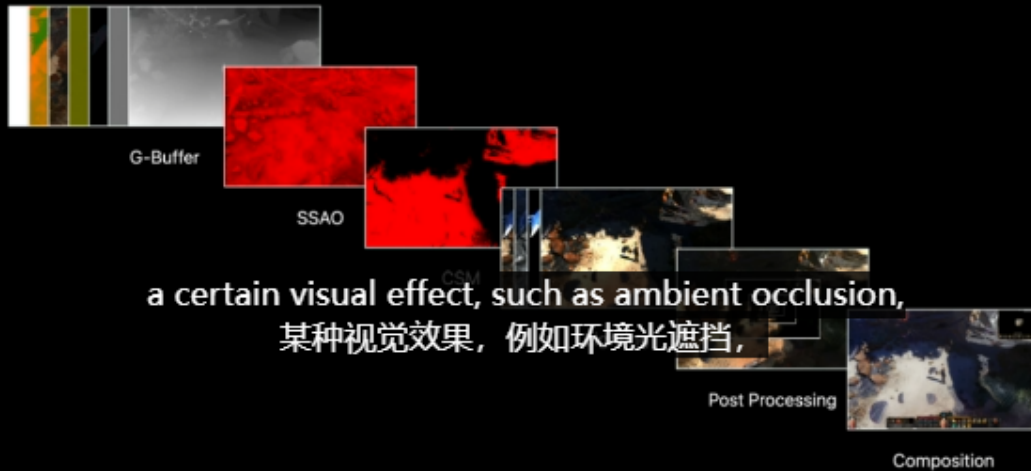
1. Measure
2. Analyze
3. Improve
4. Verify

## CheckList

### Optimization checklist

- ✓ Use Xcode and Instruments regularly
- ✓ Shader performance
- ✓ Memory bandwidth
- ✓ Workload overlap
- ✓ Resource dependencies
- ✓ Development workflows
- ✓ Redundant bindings

## Analyze frame graph



## Metal Debugger

### Summary页面

#### Show Dependencies

#### BandWidth

Loseless Compression Warnings for textures and Reasons

#### API

Redundant Bindings

### Memory页面

1. Filter by RenderTargets
2. Texture---Usage
- 3.

### Group by API Call

### Group By Pipeline State

#### Function Statistics

观察指令数

#### Pipeline Stage

观察计算过程中绑定的资源

## Compile Options

Fast math flag

### Fast math

Trades between speed and correctness

- No NaNs
- No Infs
- No signed zeros
- Allows algebraically equivalent transformations

Check if you need IEEE 754 conformance

## Instruments Tools

---

Game Performance

Metal System Trace(GPU)

## 优化

---

### Shader

#### Improving complex shaders performance

Break complex shaders into shader permutations

Simplify shaders to reduce register pressure

Prefer 16-bit types, if appropriate, over 32-bit types to reduce register spilling

High ALU + Low shader Occupancy 指示寄存器压力

## BandWidth

---

### Verify your texture usage flags

Flags which prevent lossless compression

MTLTextureUsageUnknown

MTLTextureUsageShaderWrite

MTLTextureUsagePixelFormatView

- Color Attachment不需要ShaderWrite标志

## Memory

---

## Redunt Bindings

---

## GPU Timeline

---

### GPU

#### Fragment

所有pass的时间分布

shader时间

load/store

#### Encode

一次pass的所有信息

固定ALU和Occupancy两条Timeline

High ALU + Low shader Occupancy 指示寄存器压力

Floating Point Utilization

Shader right click openg shader

Shader Profiler

## Counters