VIVE

Optimization and Experience sharing

2BEARS STUDIO

VIVE

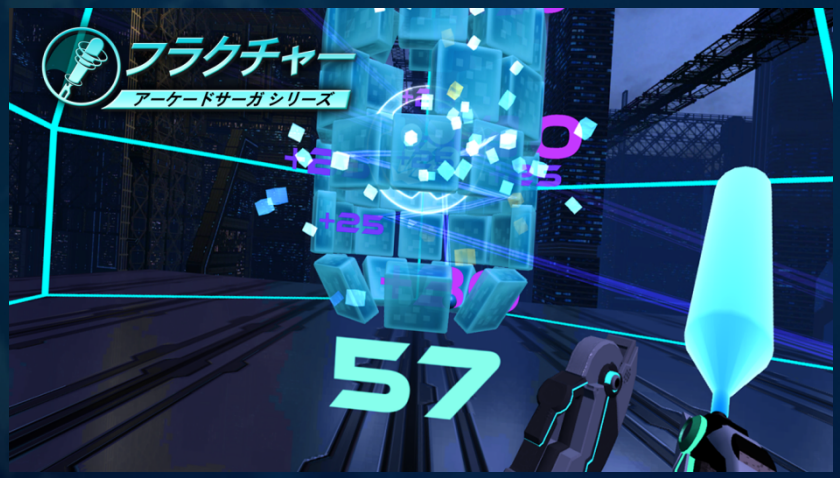
Guideline

- Why do we need to do optimization?
- Our optimization steps
- Porting experience sharing
- Conclusion

2BEARS STUDIO

Why do we need to do optimization?

- Vive game porting to Mobile platform
  - PC is much powerful than Mobile



Mobile
drawcalls about 50
Triangles about 20k

PC
drawcalls about 250
Triangles about 140k

Our optimization steps

- Analysis
  - Draw calls, Vertices, GPU
- Get the optimization factors
  - Visualize
    - Model (include Material)
    - Particles
    - Light
  - Others
    - I/O
    - Heavy calculation

Analysis

- Review and Prioritize
  - For each VFX and Models
    - Triangles
    - GPU Usage
    - Draw call
  - Lighting
- Use these data to analysis and adjust

Draw calls

- Suggest drawcalls under 80 per frame
  - Based on our experience porting existing game
    - HTC U11
    - Better trade off result between quality and frame rate
    - Here is our environment from Profiler

Fracture:

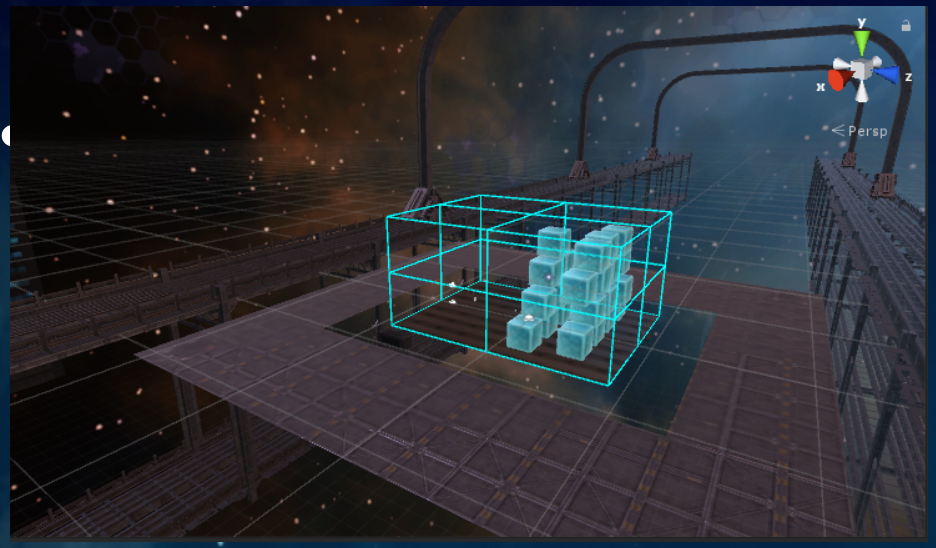
DrawsCalls:50

Triangles:20k

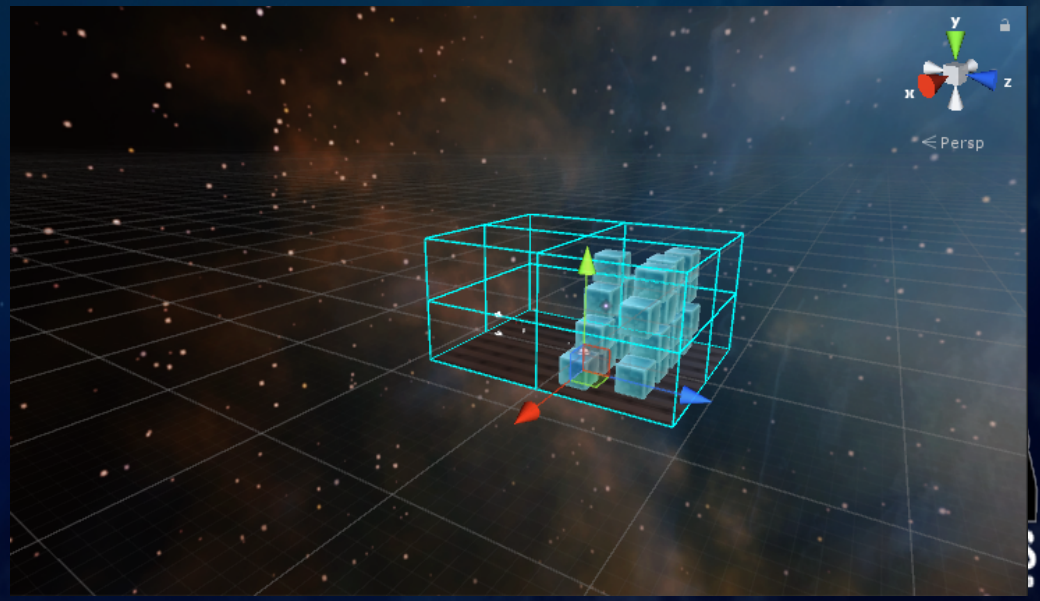
Optimization – Visual Factors

- Models
  - Mid-distance Scene, Controllers
    - Remove or simplify
  - Batching
- Particles
  - Review VFX weighting and prioritize importance
  - Decrease max particles or eliminate it
- Material
  - Use lightweight Shaders
- Light
  - Baked

Models



Description	Tris(%)
Arch bridge + Rail 1 + Rail 2	63.99%
Normal Brick Effect - 46 Bricks	17.01%
1st, 2nd, and 3rd row Bricks - 46 Bricks	4.25%
Ground + Extended Ground + Building under the ground	3.59%
Hexagonal Effect	2.84%

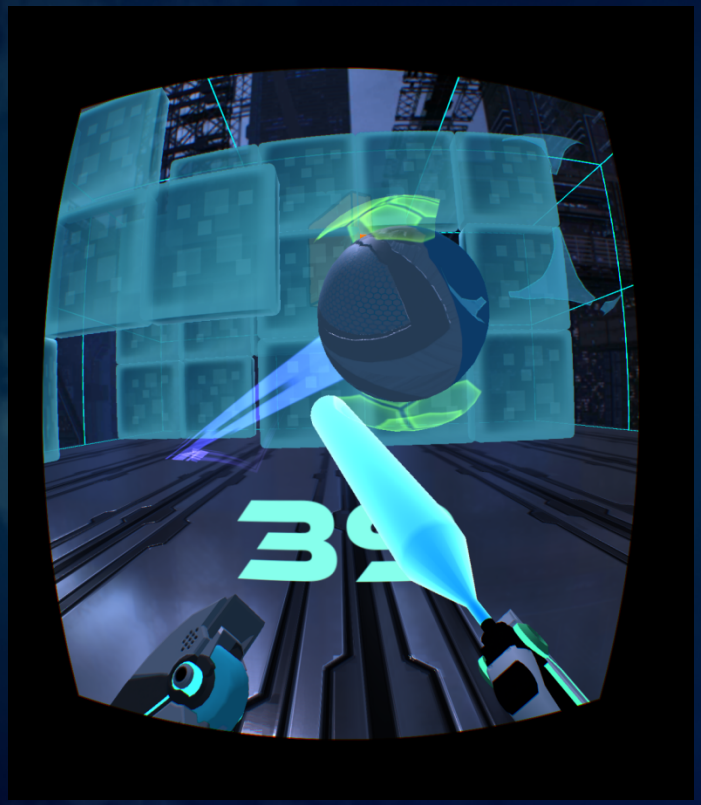


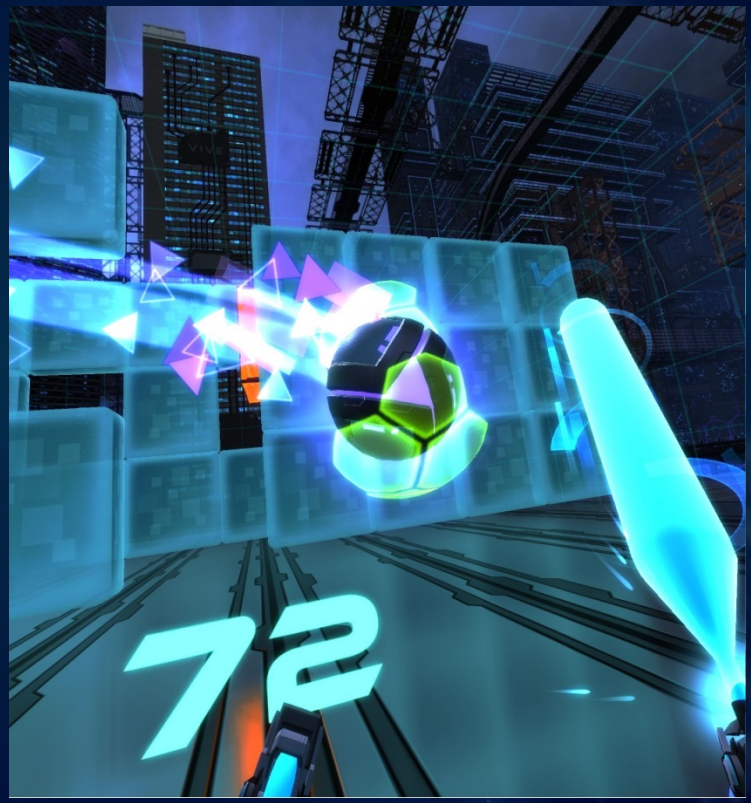
Models

- **Models**
  - **Do not leave many game objects, merge game objects to improve performance**
    - Dynamic batching
      - Group many similar vertices together
      - Things violate batching
        - » Objects are different scale
        - » Objects are different material instances
        - » Multiple lights
    - Combine mesh as much as possible
      - Material set by code
    - Remove unnecessary game objects

VIVE

Particles, Materials

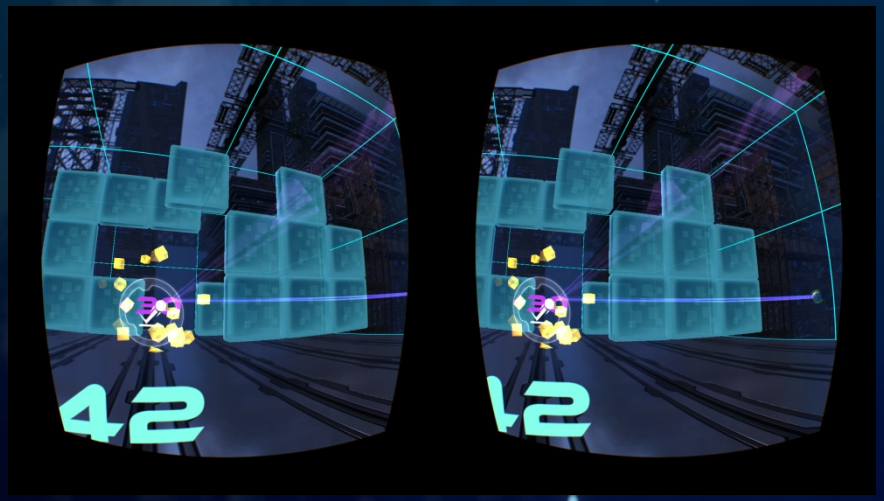


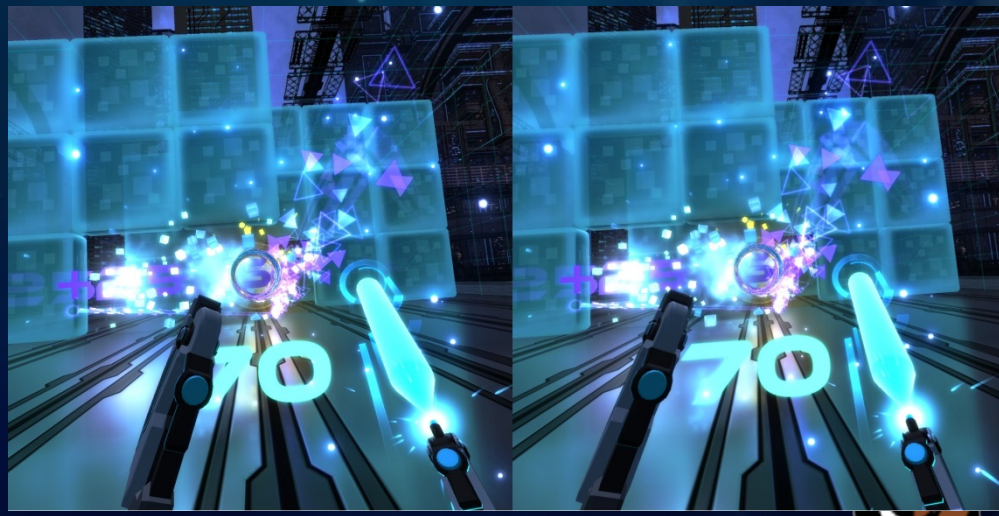


2BEARS STUDIO

Particles

- **Particles**
  - **Particle System decrease 6% performance per trigger**
    - Reserved main VFX
    - Decrease max particles





Material

- **Material**
  - **Use Mobile shaders first, and avoid using “standard” shader**
    - Unity provide several simplified shaders for mobile platform under “Mobile” category
      - Have significant performance advantage

Light

- **Using only one light source whenever possible**
  - Additional light may introduce extra lighting pass
    - Generates more drawcalls to impact performance
- **Usually baked is better than real-time from performance standpoint, but need to make trade-off in between visual and performance effect**

Optimization – Other Factors

- I/O
- Heavy Calculation
- Texture format
- Object Instantiation

VIVE

I/O

- Saving and loading files
  - Audio
    - Unity have 3 types for loading
      - Decompress on load
      - Compressed in memory
      - Streaming
  - Local data
    - Avoid using open/close frequently

2BEARS STUDIO

Heavy Calculation

- Use Unity Profiler to find the killer
  - Find CPU impulse
    - Function cost
      - Total usage
      - CPU Time
      - Calls
  - Find unused scripts/functions
- Garbage Collection
  - Avoid using a lot of temp variable

Texture format

- Use compressed textures to decrease the size of your textures
  - Recommend to use ASTC or ETC2
    - Quality
      - ASTC > ETC2
    - Compression rate
      - ETC2 > ATSC
  - Benefits
    - Fast loading time
    - Reducing build size

Object Instantiation

- Common objects be instantiated frequently on scenes
  - **How**
    - don’t destroy objects after used
  - **Benefit**
    - Save Instantiate cost
    - Increased FPS 15% (our game: Fracture)
  - **Objects Status in pool**
    - Deactivates

      Can be used
    - Activates

      Using by someone

Object Instantiation – Fracture Case

- Common Objects in Object Pool
  - Ball Summoner Indicator
  - Brick Shatter VFX
  - Rebound Collision VFX
  - Score Text
  - Ball

Object Instantiation – Fracture Case

	Original	Used Object Pool
FPS	47~51	60up

VIVE

Porting experience sharing

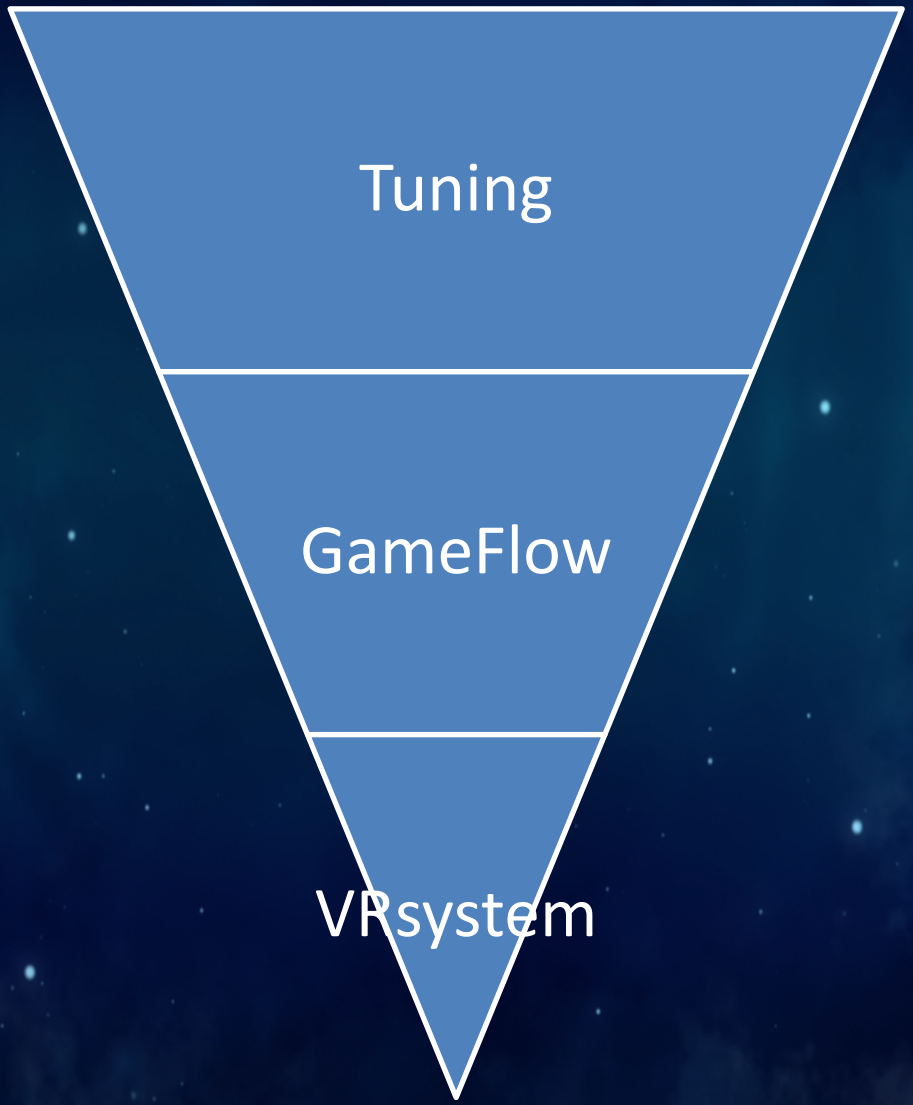
2BEARS STUDIO

Vive to Mobile -- Fracture

- Steps
  - Replace SDK
    - Functions mapping
      - Write interface to integrate each platform (VRSystem)
  - Modify Game flow
    - Revise online data access to local
    - Game design
  - Performance Tuning
    - Environment, Particles, Material, etc.

Playable

Complete



script mapping

SteamVR	WaveVR
StreamVR_Camera.cs	WaveVR_Render.cs
StreamVR_Controller.cs	WaveVR_Controller.cs
StreamVR_TrackedObject.cs	WaveVR_PoseTracker.cs
StreamVR_ControllerManager.cs	WaveVR_ControllerManager.cs
StreamVR_RenderModel.cs	WaveVR_RenderModel.cs
StreamVR_Utils.cs	WaveVR_Utils.cs

Function and API mapping

SteamVR class and enums	WaveVR class and enums
StreamVR_Controller.Device	WaveVR_Controller.Device
StreamVR_Controller.Device.GetPress	WaveVR_Controller.Device.GetPress
SteamVR_Controller.ButtonMash.Touchpad	wvr.WVR_InputId.WVR_InputId_Alias1_Touchpad
SteamVR_TrackedObject	WaveVR_PoseTracker
SteamVR_ControllerManager	WaveVR_ControllerManager

Conclusion

- Profiling early and often
- Combine or reduce meshes as much as you can
- Optimize function and remove unused scripts
- Reduce lights computation