Achieving High Quality Mobile VR Games

Roberto Lopez Mendez
Senior Software Engineer, ARM

Vision VR/AR Summit 2016 – Hollywood
11/02/2016

Agenda

- Ice Cave Demo
  - Porting Ice Cave Demo to Samsung Gear VR

- Improving VR quality & performance
  - Dynamic soft shadows, refractions and reflections based on local cubemaps
  - Stereo reflections

© ARM 2015

Demo Ice Cave

Porting Ice Cave Demo to Samsung Gear VR

© ARM 2015

Unity VR Support

unity VR

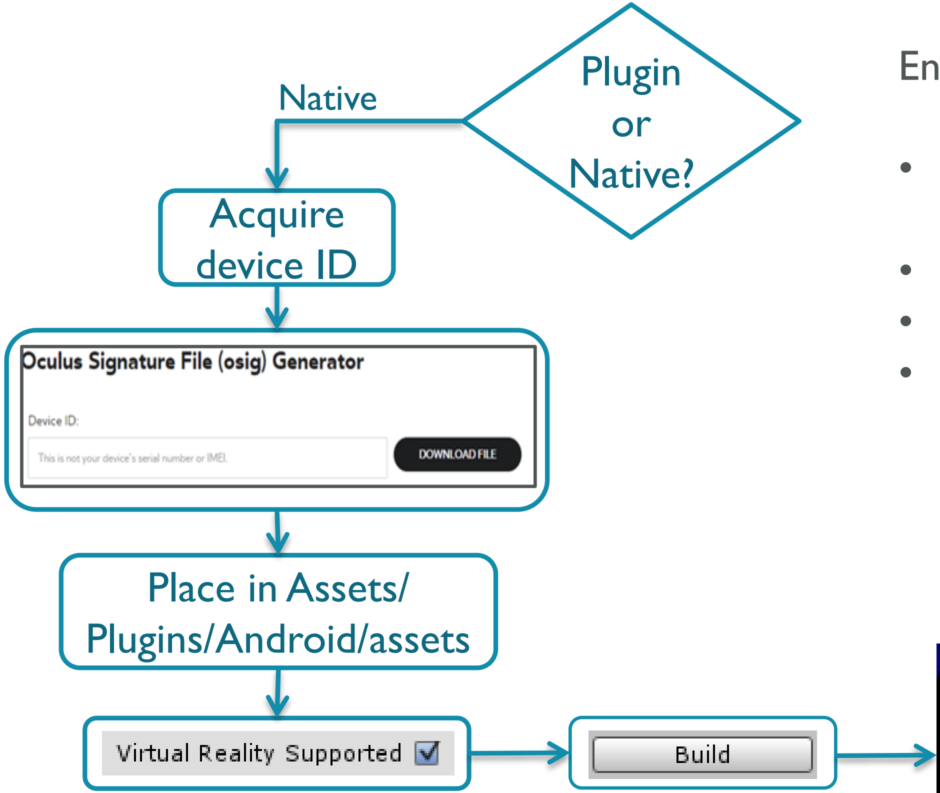
VR Plugin support limitations

- Each VR device has a different plugin
- Plugins may conflict with each other
- Each release of newer VR SDKs / Runtimes can break older games
- Lower level engine optimizations are not possible with plugin approach of two separate cameras

Unity 5.1 VR
Native Support

© ARM 2015

Porting Ice Cave for Samsung Gear VR



Plugin or Native?

Native

Acquire device ID

Oculus Signature File (osig) Generator

Device ID:

This is not your device's serial number or IMEI.

DOWNLOAD FILE

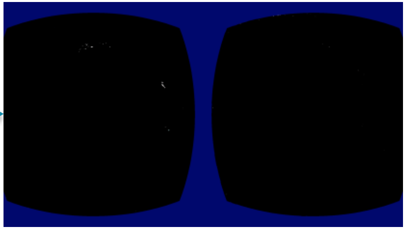
Place in Assets/ Plugins/Android/assets

Virtual Reality Supported ☑

Build

Enabling Gear VR developer mode:

- Go to your device → Settings Application Manager → Gear VR Service
- Tap on "Manage storage"
- Tap on the "VR Service Version" six times
- Wait for scan process to complete and you should now see the Developer Mode toggle



Considering VR Specifics

- Removed existing UI based on virtual joysticks
- Added very simple UI through Gear VR touchpad
- Removed camera animation to avoid motion sickness
- Carefully set the camera speed

- Ice Cave was designed big so users don't feel claustrophobic
- Removed dirty lens effect as it doesn't translate well to VR
- Added camera collision and sliding as going through geometry leads to bad VR experience

© ARM 2015

Ice Cave VR Extras

- Added streaming to another device to show what the user is actually experiencing (camera position and orientation)

- Added an alternative UI by means of a mini Bluetooth controller

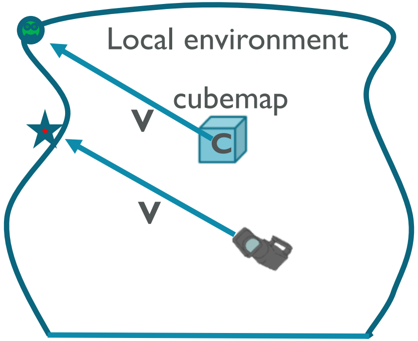
© ARM 2015

Quality and Performance for VR

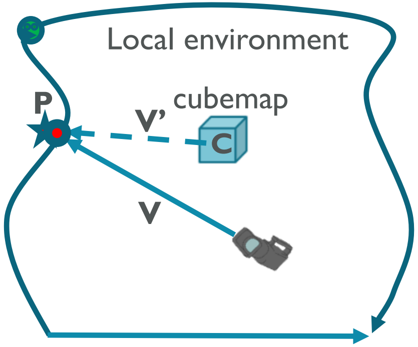
Optimized Rendering Techniques Based on Local Cubemaps

© ARM 2015

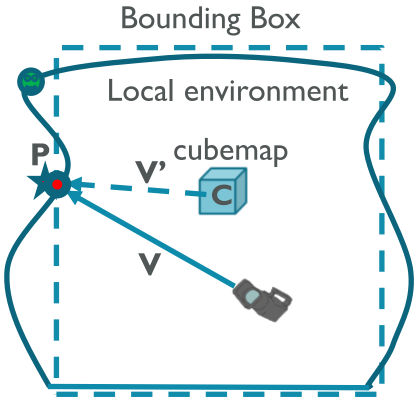
The Concept of Local Cubemaps



If we use the view vector **V** defined in WCS to fetch the texel from the cubemap we will get the smiley face instead of the star.



We need to use a new vector **V'** = **CP** to fetch the correct texel. We need to find the intersection point **P** of the view vector **V** with the boundaries of the local environment.



We introduce a proxy geometry to simplify the problem of finding the intersection point **P**.The simplest proxy geometry is the bounding box.

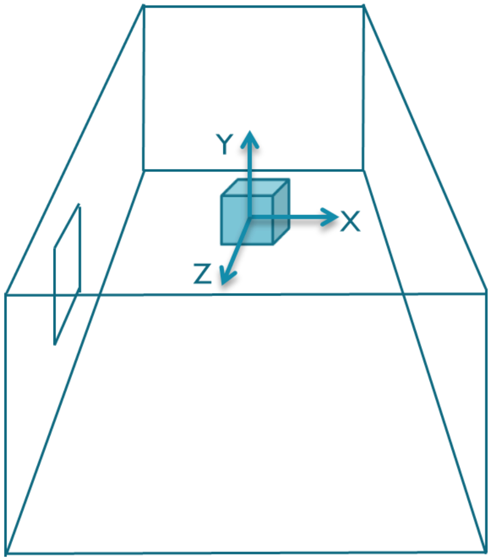
Local Cubemap = Cubemap + Cubemap Position + Scene Bounding Box + Local Correction

© ARM 2015

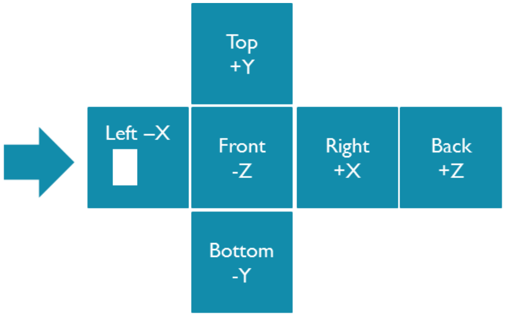
Dynamic Soft Shadows Based on Local Cubemaps

Dynamic Soft Shadows Based on Local Cubemaps

Generation Stage



Render the transparency of the scene in the alpha channel



Camera background alpha colour = 0.

Opaque geometry is rendered with alpha = 1.

Semi-transparent geometry is rendered with alpha different from 1.

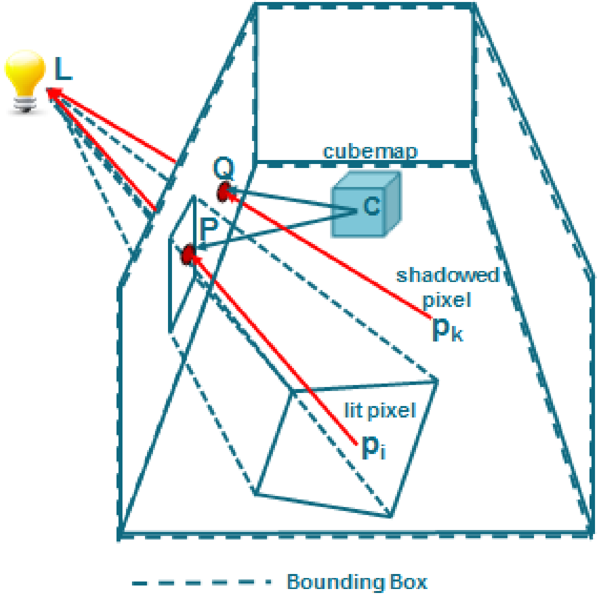
Fully transparent geometry is rendered with alpha 0.

We have a map of the zones where light rays can potentially come from and reach the geometry.

No light information is processed at this stage.

Dynamic Soft Shadows Based on Local Cubemaps

Runtime stage



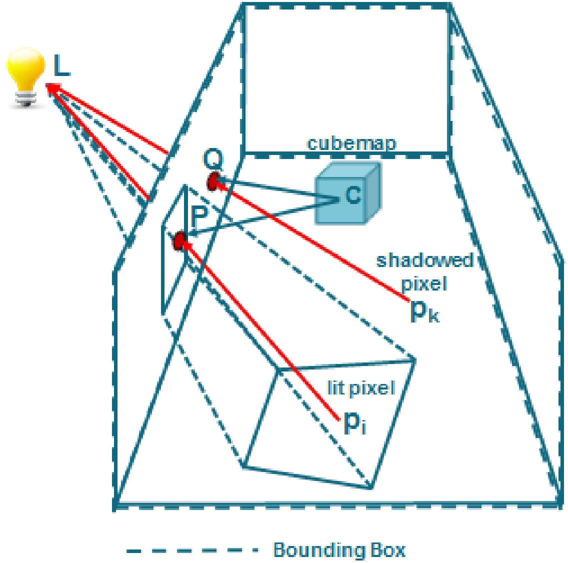
- Create a vertex to light source L vector in the vertex shader.
- Pass this vector to the fragment shader to obtain the vector from the pixel to the light position **p_iL**.
- Find the intersection of the vector **p_iL** with the bounding box.
- Build the vector **CP** from the cubemap position C to the intersection point P.
- Use the new vector **CP** to fetch the texture from the cubemap.

```
float texShadow = texCUBE(_CubeShadows,  CP).a;
```

Source code in the ARM Guide for Unity Developers at MaliDeveloper.arm.com

Dynamic Soft Shadows Based on Local Cubemaps

Why soft?

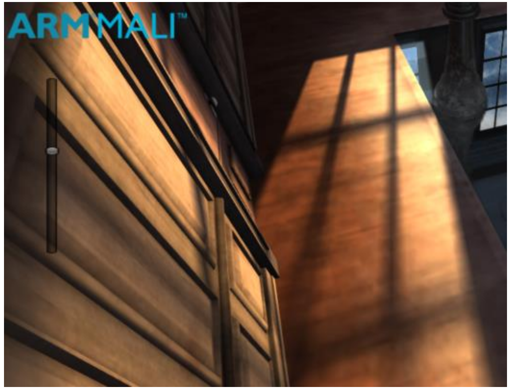


float texShadow = texCUBE(~~_CubeShadows, CP).a;~~

float4 newVec = float4(CP, factor * length(p_iP))

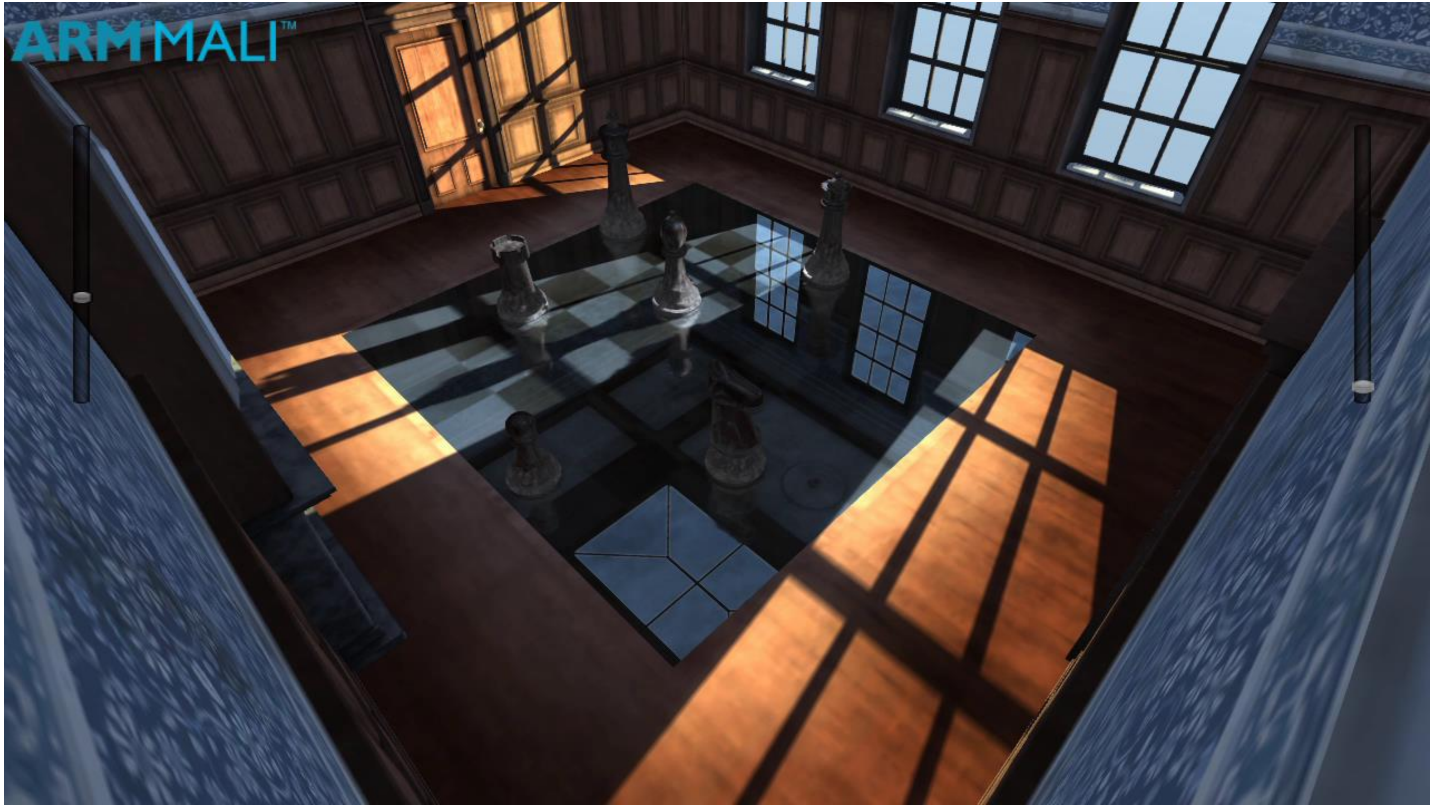
float texShadow = texCUBElod(_CubeShadows, newVec).a;

The further from the object the softer the shadows.



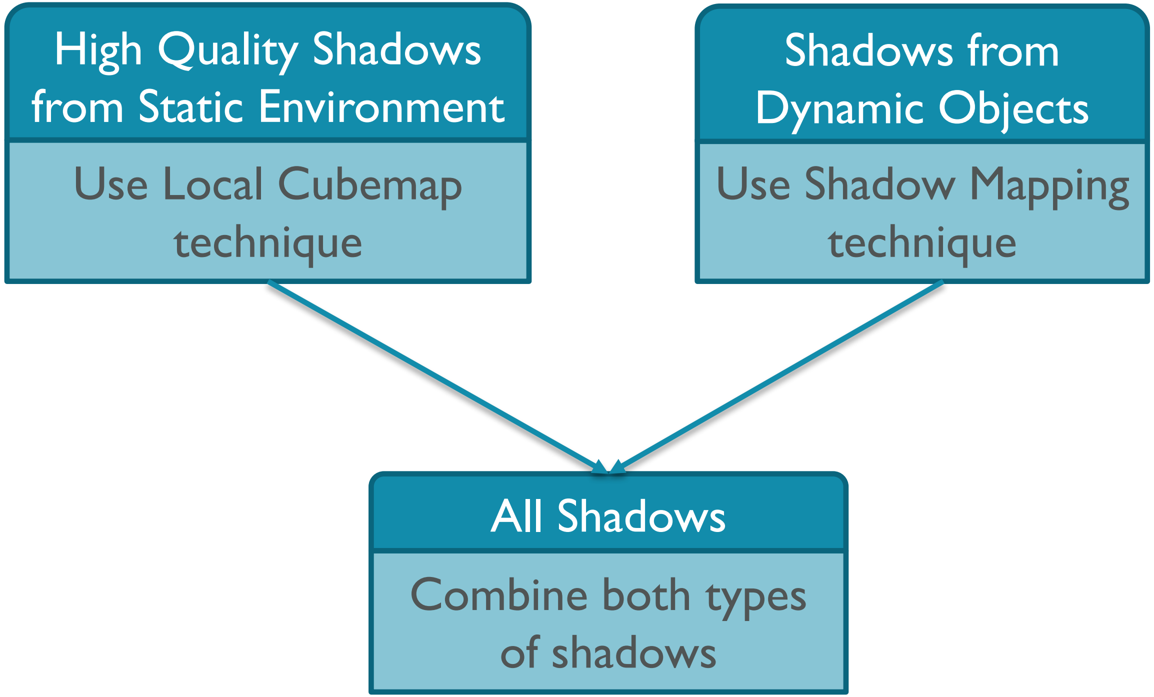
Source code in the ARM Guide for Unity Developers at MaliDeveloper.arm.com

Dynamic Soft Shadows Based on Local Cubemaps



© ARM 2015

Handling Shadows From Different Types of Geometries



High Quality Shadows from Static Environment

Use Local Cubemap technique

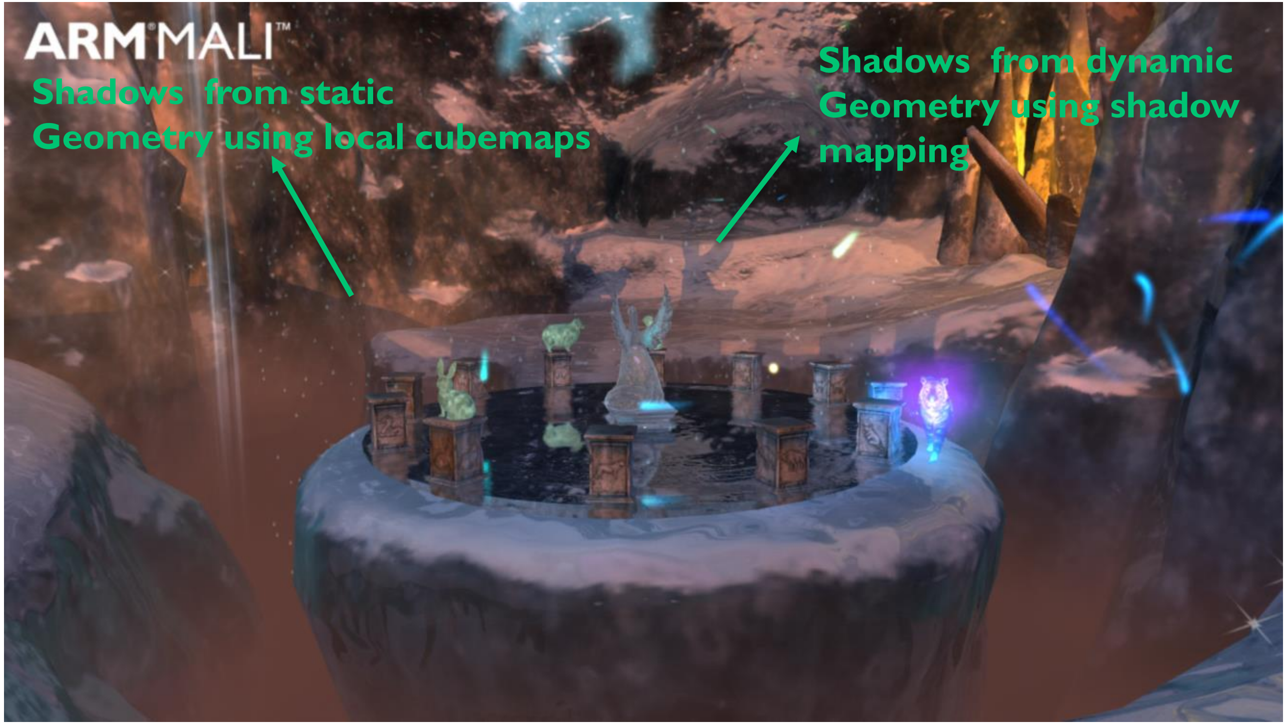
Shadows from Dynamic Objects

Use Shadow Mapping technique

All Shadows

Combine both types of shadows

Combined Shadows in Ice Cave VR



Shadows from static Geometry using local cubemaps

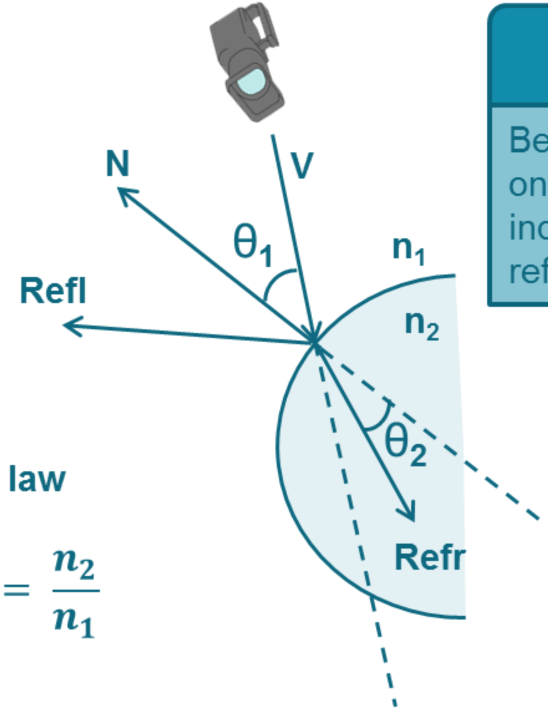
Shadows from dynamic Geometry using shadow mapping

© ARM 2015

Refraction Based on Local Cubemaps

© ARM 2015

Refraction



Refraction

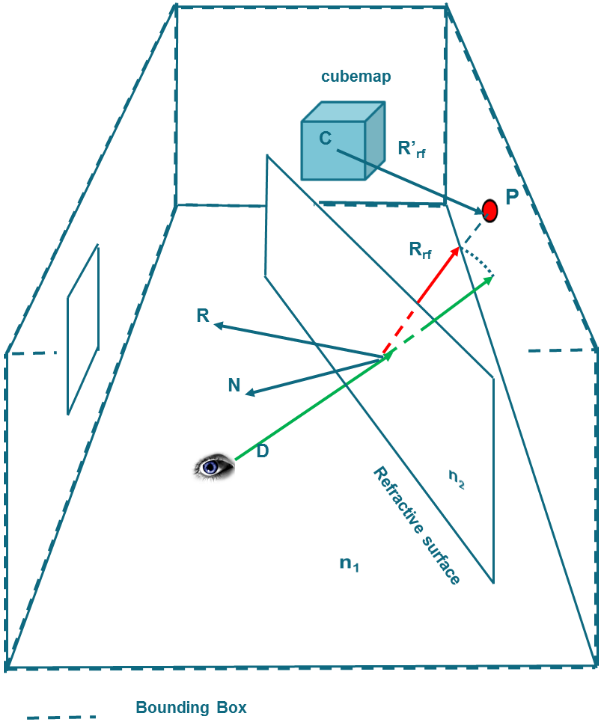
Bending of light as it passes from one medium, with refraction index n_1, to another medium with refraction index n_2.

Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$$

© ARM 2015

Local Correction to Refraction Vector



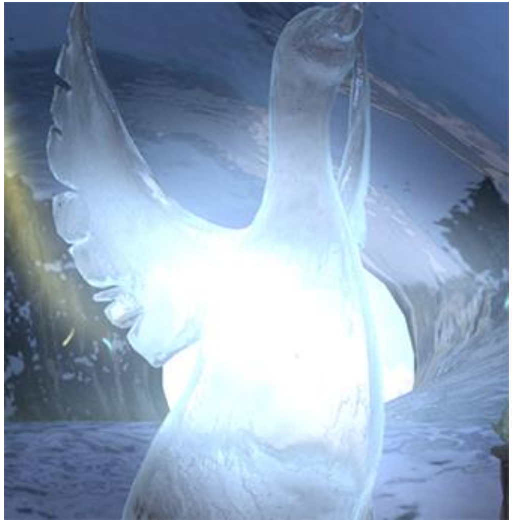
float3 R_{rf} = refract(D_{norm}, N, n_1/n_2);

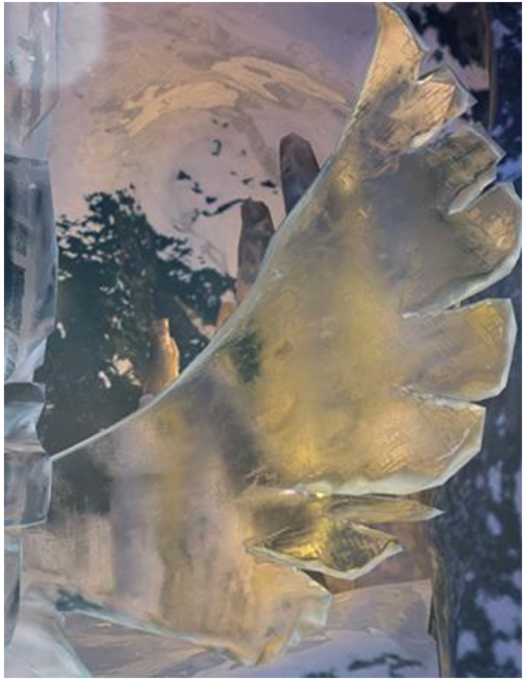
~~*float4 col = texCUBE(Cubemap, R_{rf});*~~

Find intersection point P
Find vector R'_{rf} = CP
float4 col = texCUBE(Cubemap, R'_{rf});

© ARM 2015

Refraction Based on Local Cubemaps in Ice Cave VR

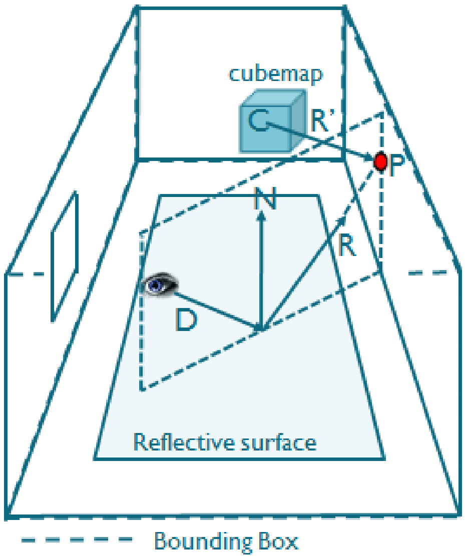




Reflections Based on Local Cubemaps

© ARM 2015

Local Correction to Reflection Vector



float3 R = reflect(D, N);

~~*float4 col = texCUBE(Cubemap, R);*~~

Find intersection point P
Find vector R' = CP
float4 col = texCUBE(Cubemap, R');

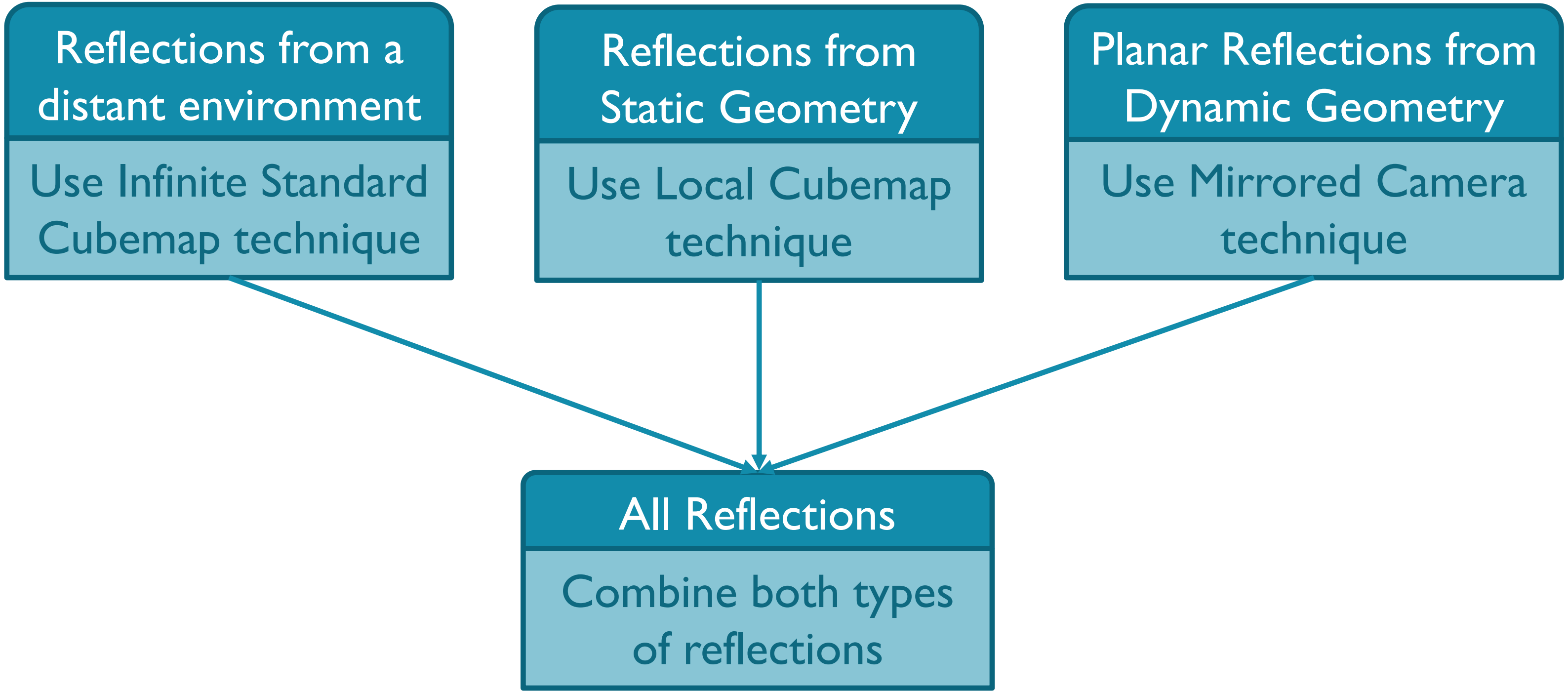
Source code in the ARM Guide for Unity Developers at MaliDeveloper.arm.com

GPU Gems. Chapter 19. Image-Based Lighting. Kevin Bjork, 2004. http://http.developer.nvidia.com/GPUGems/gpugems_ch19.html

Cubemap Environment Mapping. 2010. http://www.gamedev.net/topic/568829-box-projected-cubemap-environment-mapping/?&p=4637262

Image-based Lighting approaches and parallax-corrected cubemap. Sebastien Lagarde. SIGGRAPH 2012. http://seblagarde.wordpress.com/2012/09/29/image-based-lighting-approaches-and-parallax-corrected-cubemap/

Handling Different Types of Reflections



Reflections from a distant environment

Use Infinite Standard Cubemap technique

Reflections from Static Geometry

Use Local Cubemap technique

Planar Reflections from Dynamic Geometry

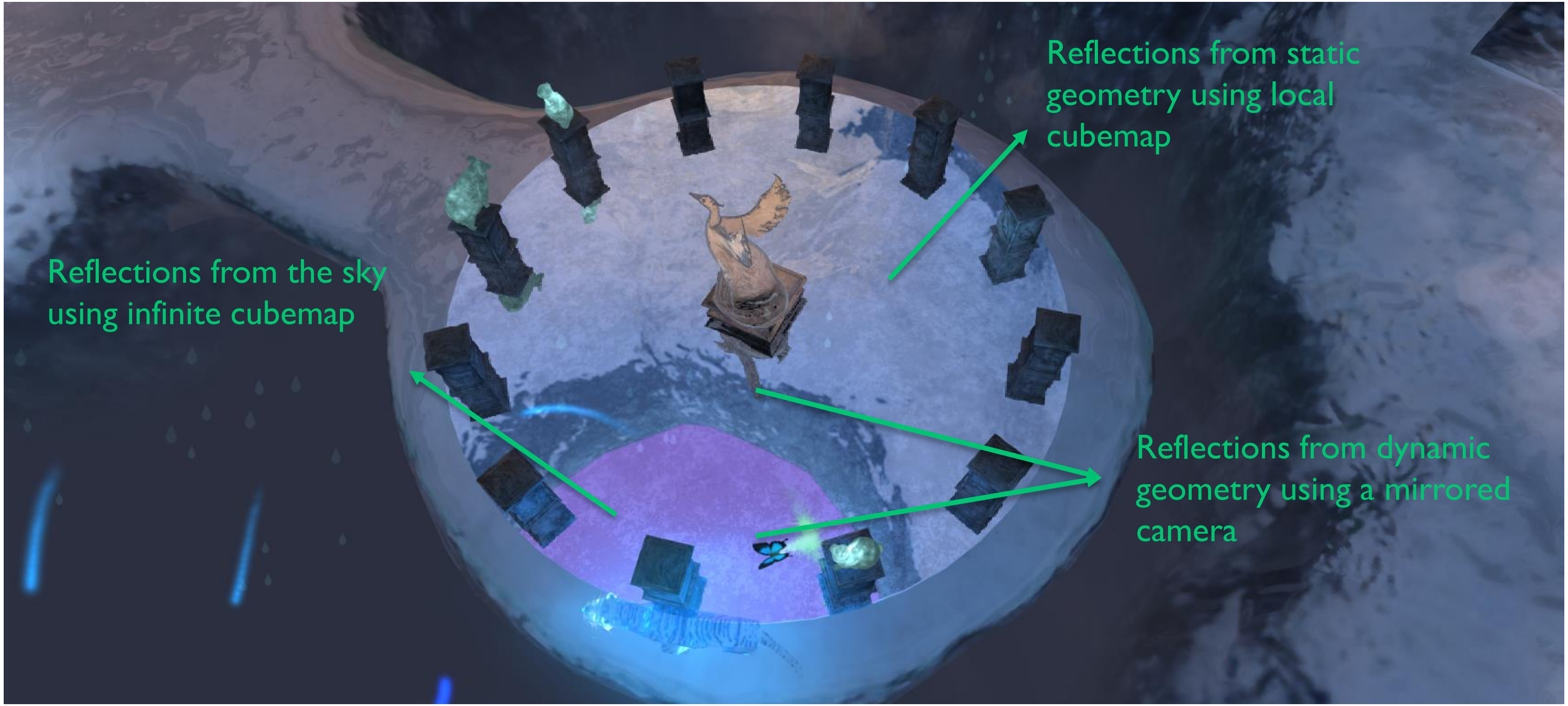
Use Mirrored Camera technique

All Reflections

Combine both types of reflections

© ARM 2015

Combined Reflections in Ice Cave VR



Reflections from static geometry using local cubemap

Reflections from the sky using infinite cubemap

Reflections from dynamic geometry using a mirrored camera

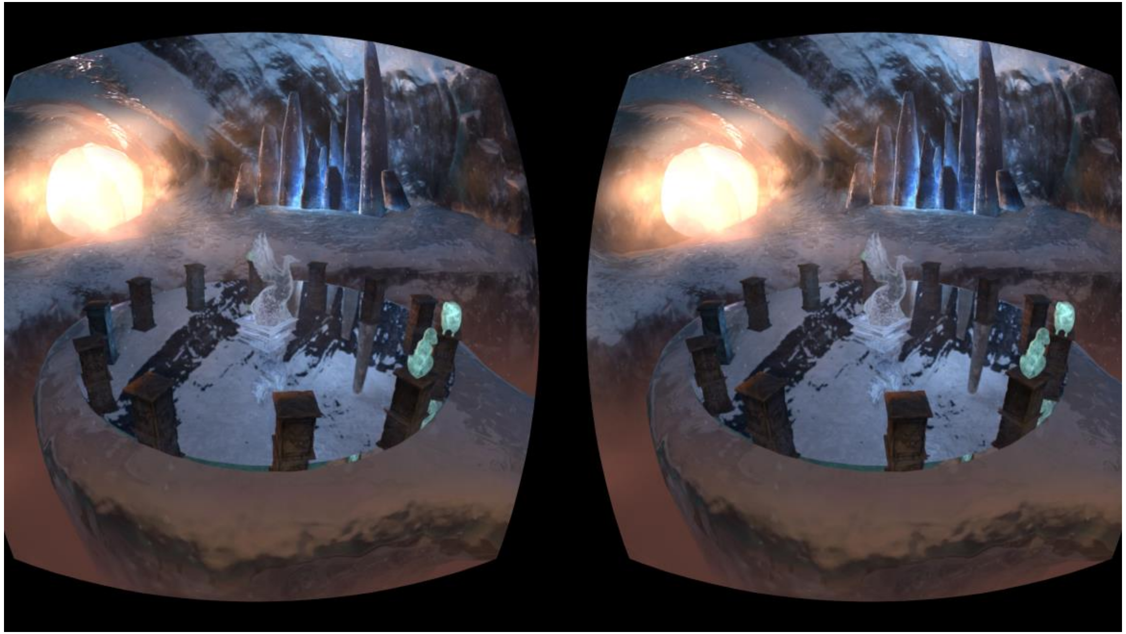
© ARM 2015

Stereo Reflections in Unity

© ARM 2015

Why Stereo Reflections?

- Using the same texture for right/left eyes reflections in VR looks plain and breaks the sensation of full immersion.



Implementing Stereo Reflections in Unity

Add two new cameras targeting left/right eye respectively and disable them as you will render them manually.

Create a target texture the cameras will render to.

Attach the script to each camera, the SetUpCamera function places the camera in the right position for rendering reflections for each eye.

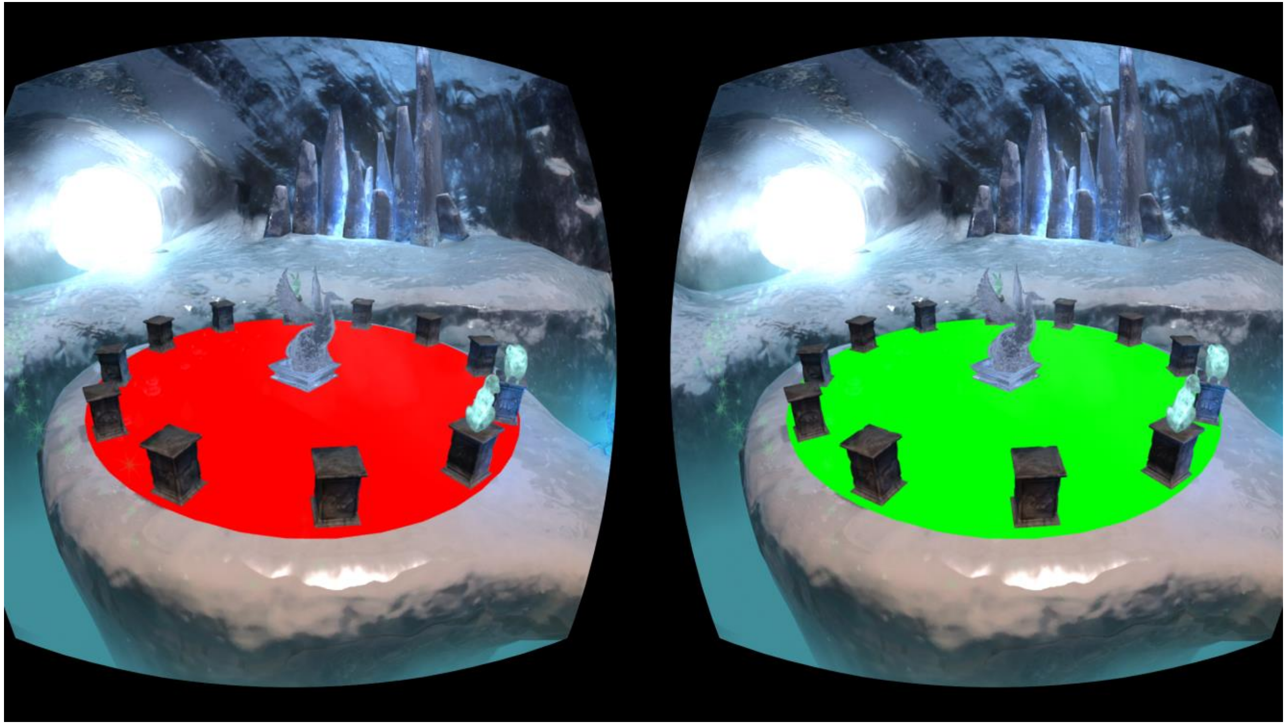
```
void OnPreRender()
{
    SetUpCamera ();
    // Invert winding
    GL.invertCulling = true;
}
void OnPostRender()
{
    // Restore winding
    GL.invertCulling = false;
}
```

Implementing Stereo Reflections in Unity

- Attach the script to the main camera

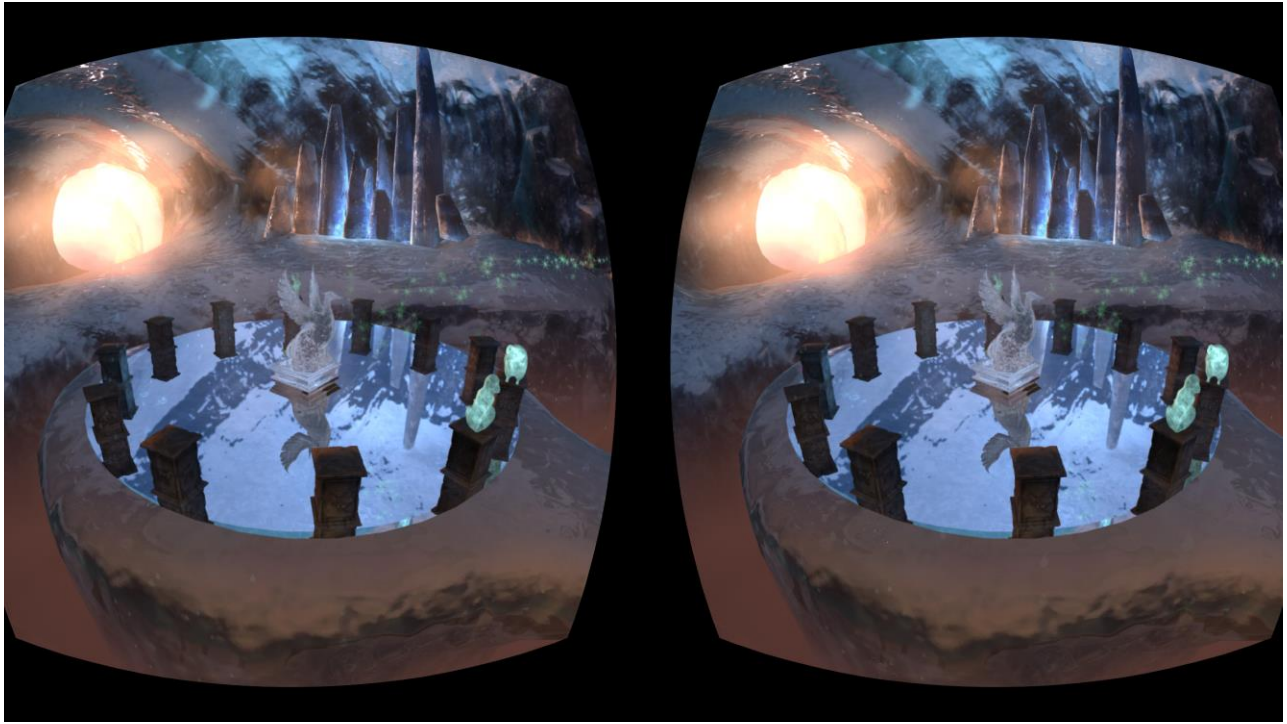
```
public class RenderStereoReflections : MonoBehaviour
{
    public GameObject reflectiveObj;
    public GameObject leftCamera;
    public GameObject rightCamera;
    int eyeIndex = 0;

    void OnPreRender(){
        if (eyeIndex == 0){
            // Render Left camera
            leftCamera.GetComponent<Camera>().Render();
            reflectiveObj.GetComponent<Renderer>().material.SetTexture("_DynReflTex", leftCamera.GetComponent<Camera>().targetTexture );
        }
        else{
            // Render right camera
            rightCamera.GetComponent<Camera>().Render();
            reflectiveObj.GetComponent<Renderer>().material.SetTexture("_DynReflTex", rightCamera.GetComponent<Camera>().targetTexture );
        }
        eyeIndex = 1 - eyeIndex;
    }
}
```

Check Left/Right Reflection Synchronization



© ARM 2015

Stereo Reflections in Ice Cave VR



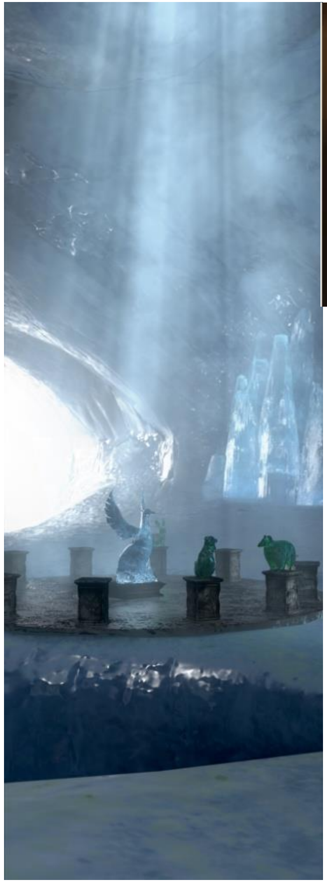
© ARM 2015

Wrap Up

- Unity has made a great contribution to VR democratization
- VR is a new boost to mobile games. The user experience is no longer limited to the mobile screen. The user is now embedded in a virtual world
- It is possible to run high quality VR and non-VR content in mobile devices using optimized rendering techniques. Stereo reflections improves VR user experience.
- Check out The ARM Guide for Unity Developers for optimizations tips, recommendations and very efficient rendering techniques to make the most out of Unity when developing VR mobile games.

© ARM 2015

To Find Out More….



- Find out more about techniques based on local cubemaps at:
  - http://malideveloper.arm.com/documentation/developer-guides/arm-guide-unity-enhancing-mobile-games/
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap
  - http://community.arm.com/groups/arm-mali-graphics/blog/2014/08/07/reflections-based-on-local-cubemaps
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/refraction-based-on-local-cubemaps
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/05/21/the-power-of-local-cubemaps-at-unite-apac-and-the-taoyuan-effect
  - http://malideveloper.arm.com/documentation/tutorials/game-development-tutorials/

Thank you

ARM

Questions

The trademarks featured in this presentation are registered and/or unregistered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

Copyright © 2015 ARM Limited