Enhancing Your Unity Mobile Games

Nathan Li – SGL & Staff Engineer, ARM

The Architecture for the Digital World®

Agenda

- Introduction
- Notes About ARM Guide To Unity
- Reflections and Shadows Based on Local Cubemaps

Notes About The ARM Guide To Unity

The ARM Guide to Unity: Enhancing Your Mobile Games

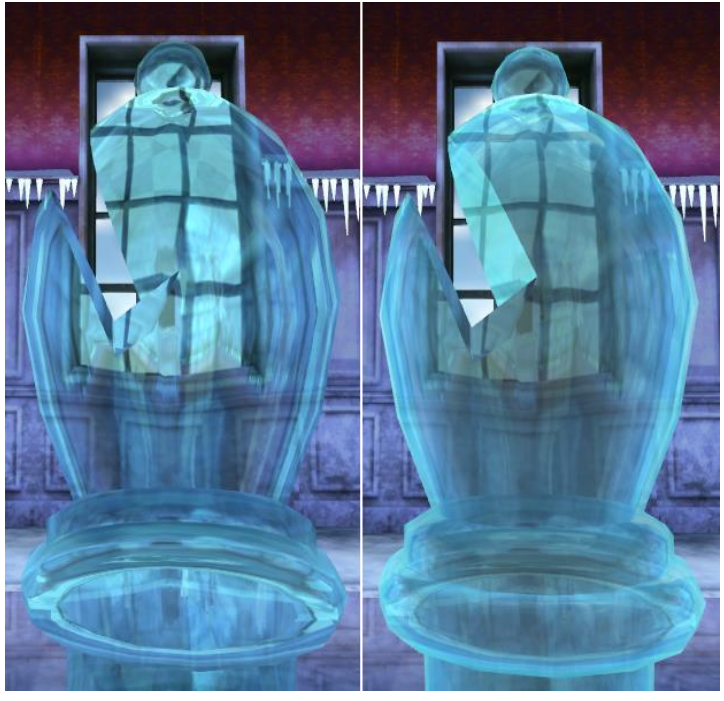
- Released at Unite Seattle 2014
- Initial idea: write about the experience of learning Unity from scratch
- Available for free at MaliDeveloper.arm.com
- Current Content:
  - Optimizing Applications. Quality settings in Unity (Chapter 2)
  - Profiling Applications with the Unity Profiler(Chapter 3)
  - A list of recommended optimizations: Application, GPU and Assets (Chapter 4)
  - Advance Rendering Techniques (Chapter 5)

ARM

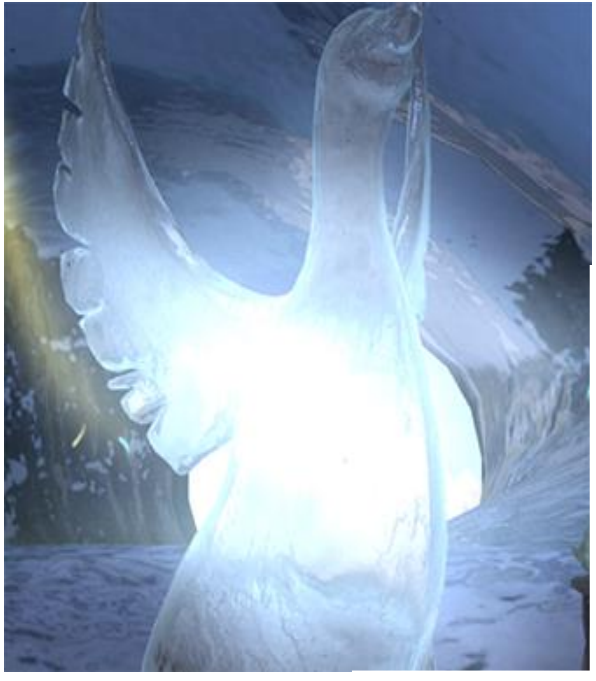
The ARM Guide to Unity: Enhancing Your Mobile Games

- Now more ambitious goals:
  - To provide solutions to more common problems
  - To provide game developers with optimized rendering techniques for mobiles
- V2.0 Release: Unite Europe 2015
  - More optimization tips
  - New tips to make the most from ARM Mali GPUs (Early-Z, Pixel Forward Kill)
  - New technique to render high quality and efficient shadows based on local cubemap
  - Combining efficiently different type of reflections
  - Implementing high quality refractions based on local cubemaps
  - Unity Global Illumination: Enlighten
  - Enlighten in custom shaders

Refraction Based on Local Cubemaps



Very efficient and physically based refractions. High quality refractions combined with reflections.



Refraction in the Ice Cave demo

ARM

Reflections and Shadows Based on Local Cubemaps

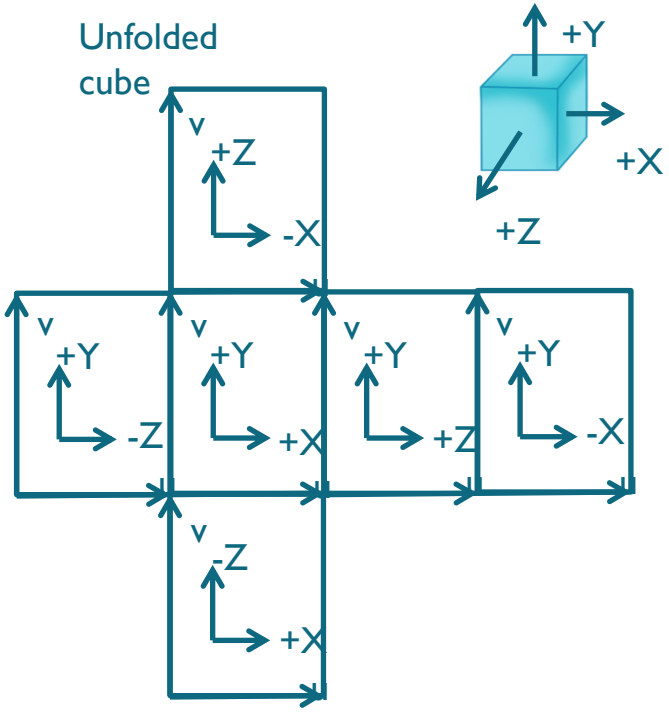
Content

- Reflections based on local cubemaps
  - Reflections with infinite cubemaps
  - Local correction to reflections based on cubemaps
  - Combining reflections based on local cubemaps with reflections rendered at runtime
- New shadows technique: dynamic soft shadows based on local cubemaps
  - The foundations of shadows based on local cubemaps
  - Why dynamic? Why soft?
  - Benefits and limitations of shadows based on local cubemaps
  - Combining shadows based on local cubemaps with shadows rendered at runtime
- Wrap up

Cubemaps

Cubemaps

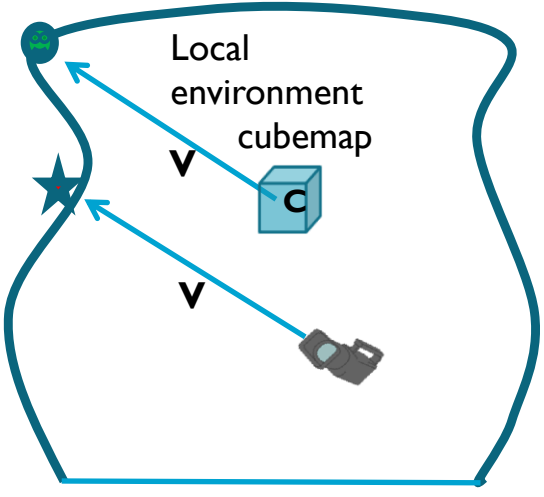
- Hardware accelerated
- No image distortion
- Efficient computation



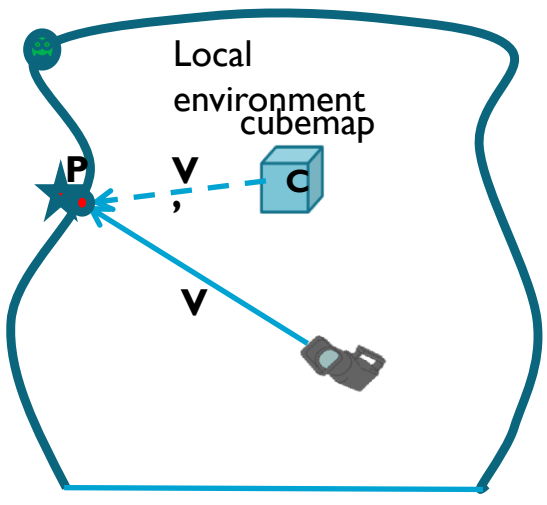

```
float4 col = texCUBE(Cubemap, V);
```


ARM

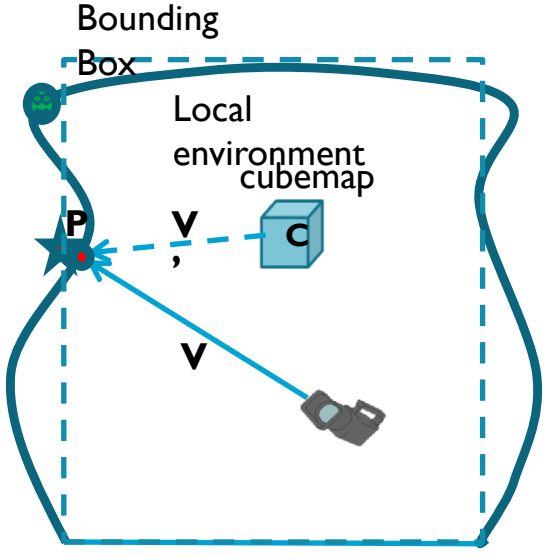
The Concept of Local Cubemap



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.

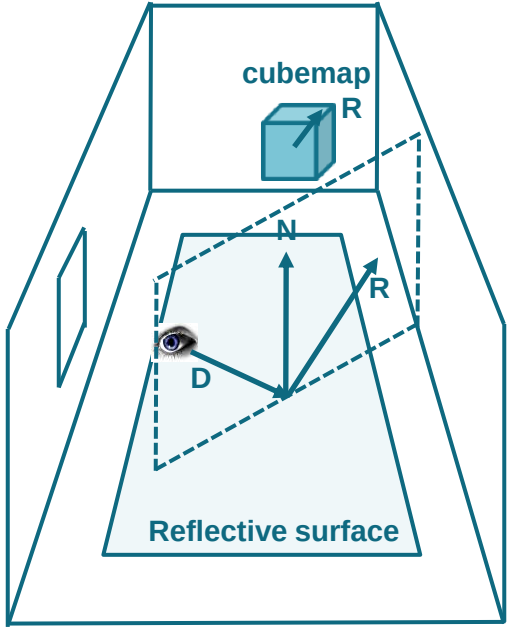


We introduce a proxy geometry to simplify the problem of finding the intersection point **P**. The simplest proxy geometry is the bounding box.

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

ARM

Reflections with Infinite Cubemaps



Normal N and view vector D are passed to fragment shader from the vertex shader.

In the fragment shader the texture colour is fetched from the cubemap using the reflected vector R:

float3 R = reflect(D, N);
float4 col = texCUBE(Cubemap, R);

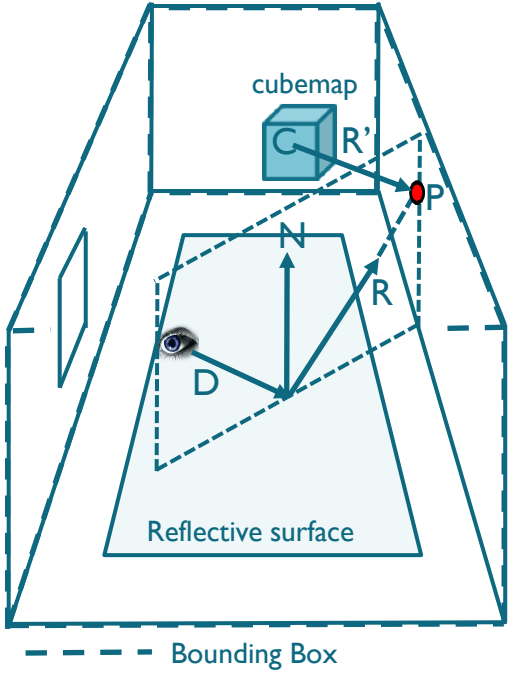
ARM

Incorrect Reflections

Reflection generated using a cubemap without any local binding

ARM

Local Correction Using a Bounding Box as a Proxy Geometry



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 to Unity at MaliDeveloper.ar.m.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/

ARM

Correct Reflections

Reflection generated after applying the "*local correction*"

Reflection generated without "*local correction*"

ARM

Infinite and Local Cubemaps

Infinite Cubemaps

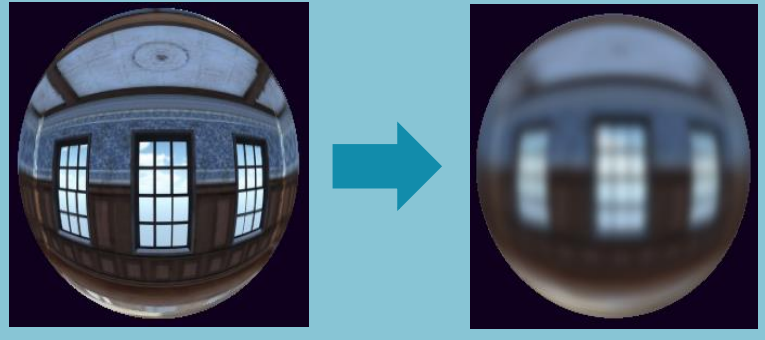
- They are used to represent the lighting from a distant environment.
- Cubemap position is not relevant.

Local Cubemaps

- They are used to represent the lighting from a finite local environment.
- Cubemap position is relevant.
- The lighting from these cubemaps is right only at the location where the cubemap was created.
- *Local correction* must be applied to get the right local reflections.

Filtering Cubemaps to Achieve Visual Effects

Development Stage



The cost of the process depends on
filter complexity and cubemap resolution

Runtime

Just use the filtered cubemap

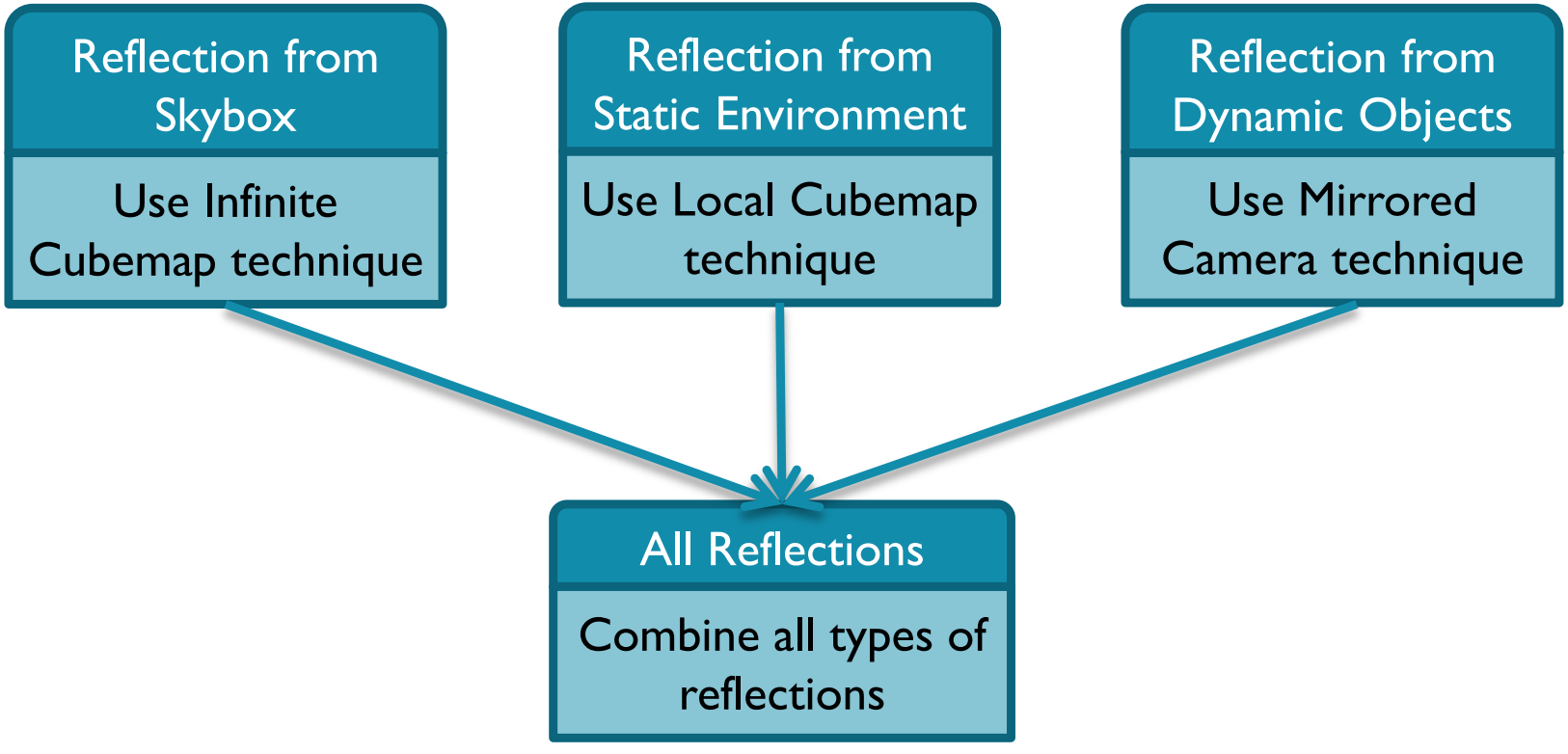
ARM

Reflections Based on Filtered Local Cubemap

Reflection as rendered with blurry effect

ARM

Handling Reflections From Different Types of Geometries



ARM

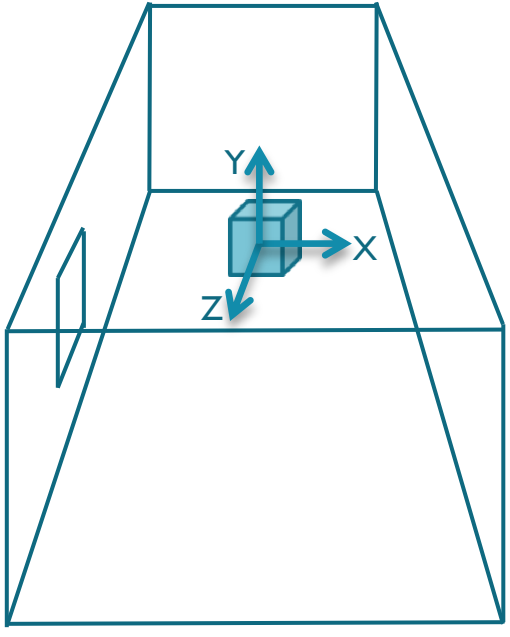
Combined reflections

ARM

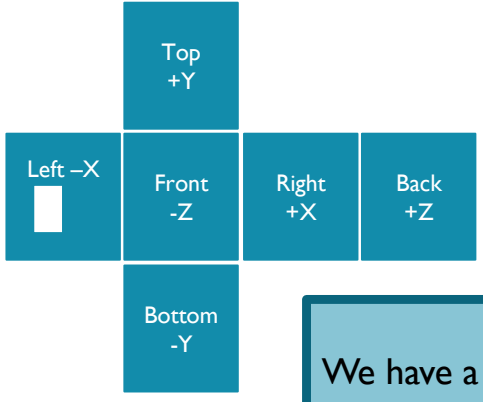
Dynamic Soft Shadows

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.

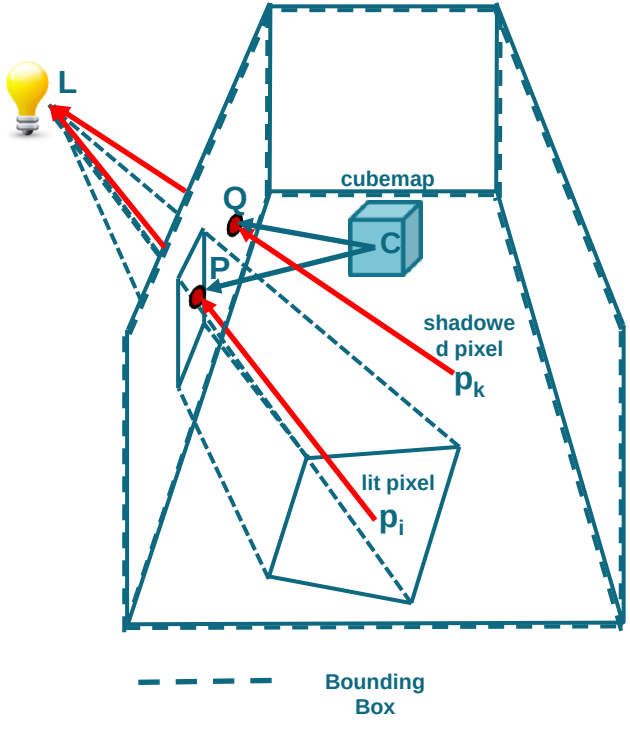
We have a map of the zones where light rays can potentially come from and reach the geometry.

No light information is processed at this stage.

ARM

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 $\mathbf{p_iL}$.
- Find the intersection of the vector $\mathbf{p_iL}$ 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 update of ARM Guide to Unity at Unite Europe 2015

ARM

Dynamic Soft Shadows Based on Local Cubemaps

Why dynamic?

If the position of the light source changes the shadows are generated correctly using the same cubemap.

ARM

Dynamic Shadows

ARM

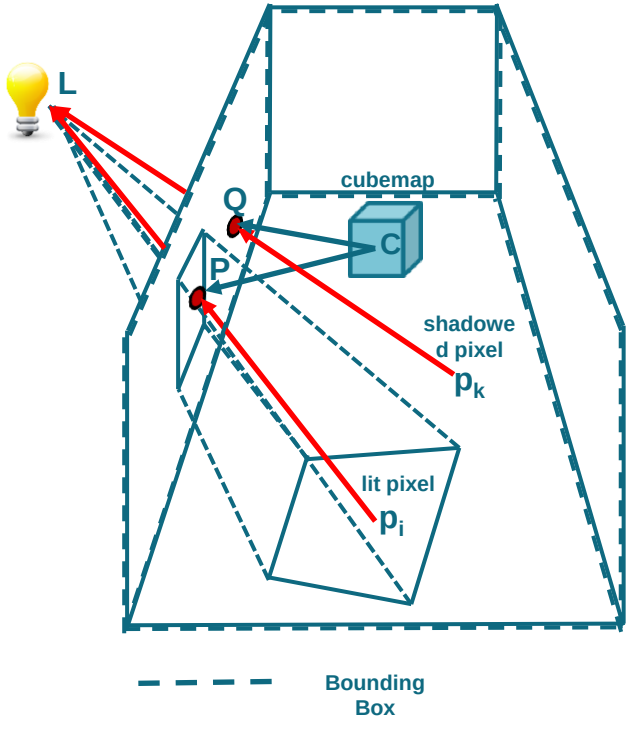
Dynamic Soft Shadows Based on Local Cubemaps

Why soft?

ARM

Dynamic Soft Shadows Based on Local Cubemaps

Why soft?



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

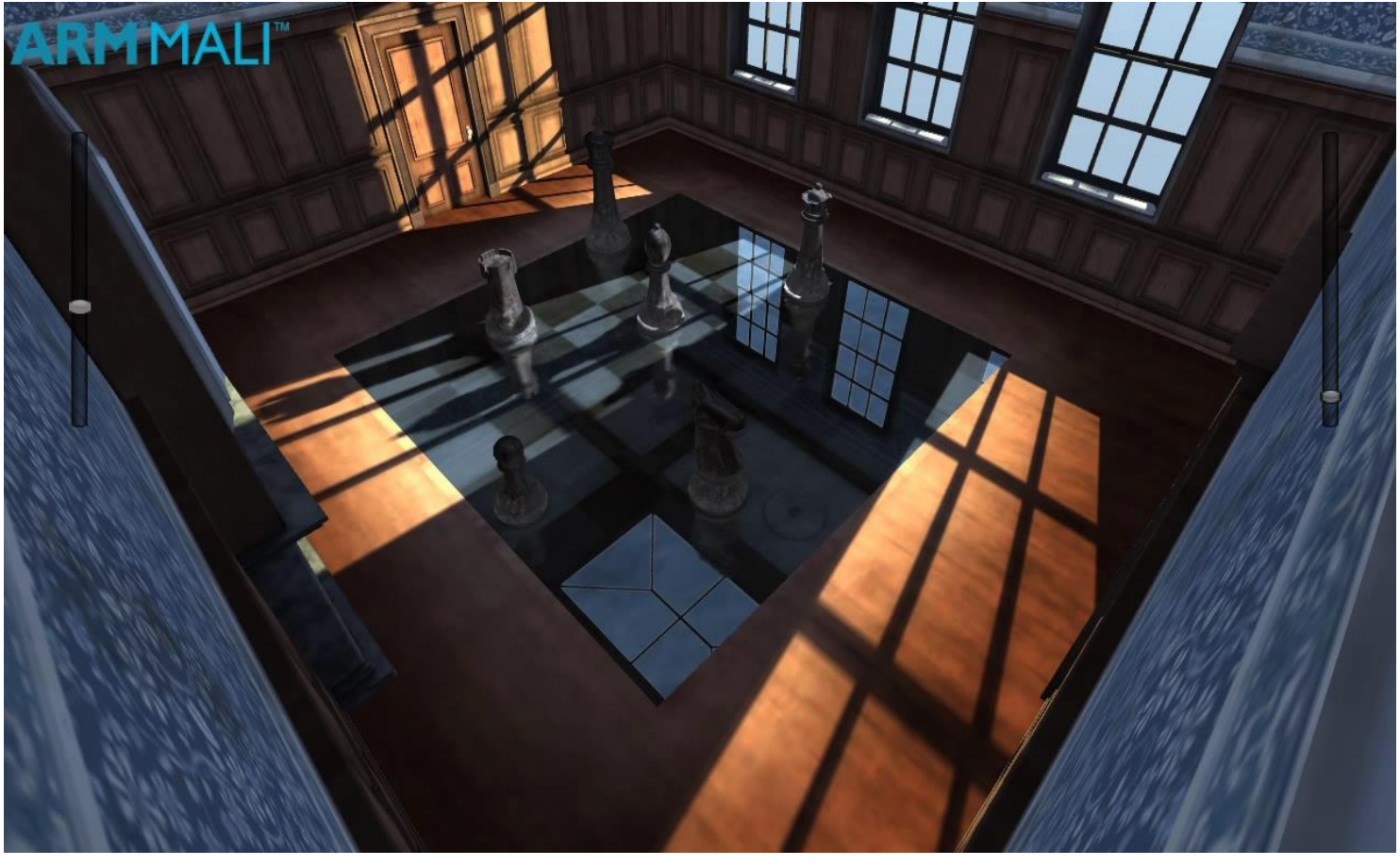
float4 newVec = float4(CP, factor * length(p_iP))

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

Source code in the update of ARM Guide to Unity at Unite Europe 2015.

ARM

Soft shadows



ARM

Dynamic Soft Shadows Based on Local Cubemaps

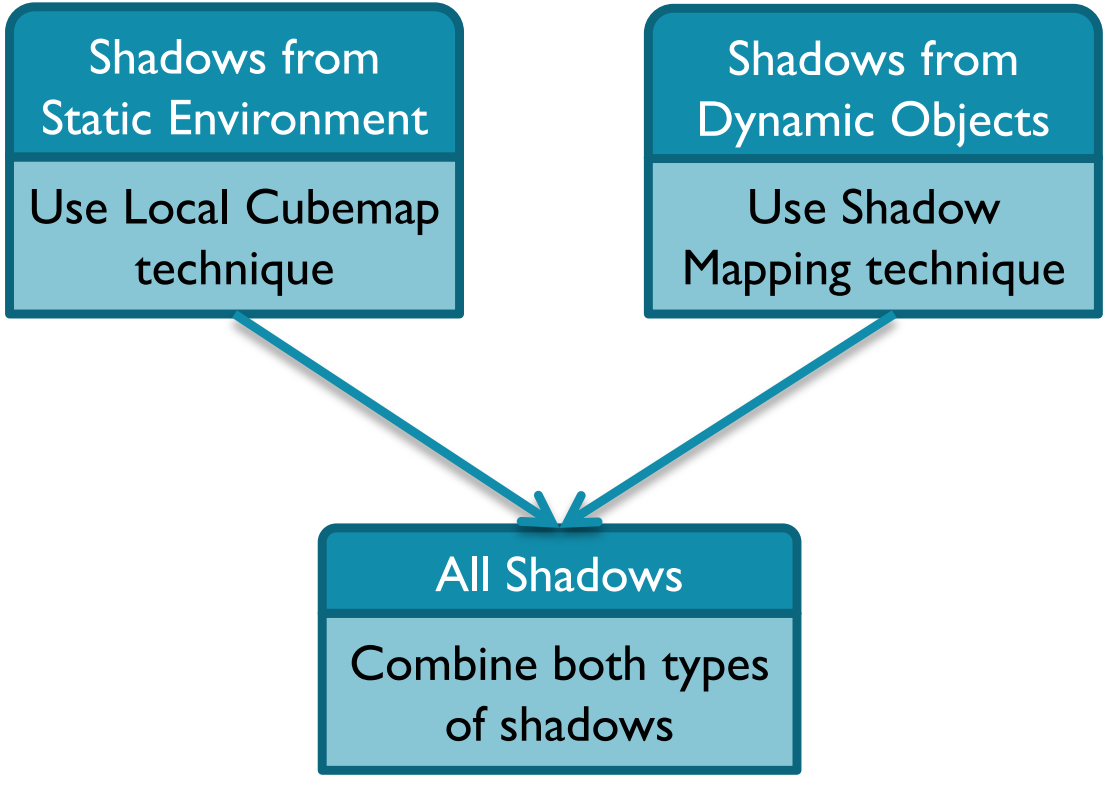
Benefits

1. Simple to implement.
2. Very realistic and physically correct.
3. High quality of shadows.
4. Cubemap texture can be compressed.
5. Offline filtering effects can be applied which could be very expensive at run time.
6. Resource saving technique compared with runtime generated shadows.
7. Very tolerant of deviations from the bounding box shape when compared with reflections.

Limitations

1. Works fine in open space with no geometry in the centre from where the cubemap will more likely be generated.
2. Objects in the scene must be close to the proxy geometry when generating static texture for good results.
3. Does not reflect changes in dynamic objects unless we can afford to update the cubemap at runtime.

Handling Shadows from Different Types of Geometries



ARM

Combined shadows

ARM

Wrap Up

- Reflections and shadows based on local cubemaps are resource saving techniques that work great in mobile devices where available resources must be carefully balanced.

- These techniques can be effectively combined with runtime techniques to render static and dynamic objects together.

- Both techniques are currently used in the Ice Cave demo together with other FX as fog, light shafts, procedural skybox, Enlighten in custom shaders, fireflies, dirty lens, etc.

Source code in the update of ARM Guide to Unity at Unite Europe 2015.

ARM

Demo Ice Cave

ARM

To Find Out More….

- All **presentations** and related **tutorials** from today available at:

MaliDeveloper.arm.com

- Any questions? malidevelopers@arm.com
- Find out more about shadows based on local cubemaps at:
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap
- Find out more about other techniques based on local cubemaps at:
  - 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

ARM

Thank You