Optimized rendering techniques based on local cubemaps

The evolution of cubemaps

ARM

Roberto Lopez Mendez
Senior Software Engineer, ARM

ACI Technical Lecture – Imperial College, London
11/10/2016

About me

- Originally a nuclear physicist
- Working on graphics since 1995 in many different products (3D organic modeller, 3D fashion designer, 3D dental prosthesis designer, 3D reconstruction from pics, remote surveillance, 3D architecture visualization, Lynx helicopter simulator, etc.)
- Love graphics because WYCIWYS
- Since 2012 working in ARM on mobile graphics technology

Content

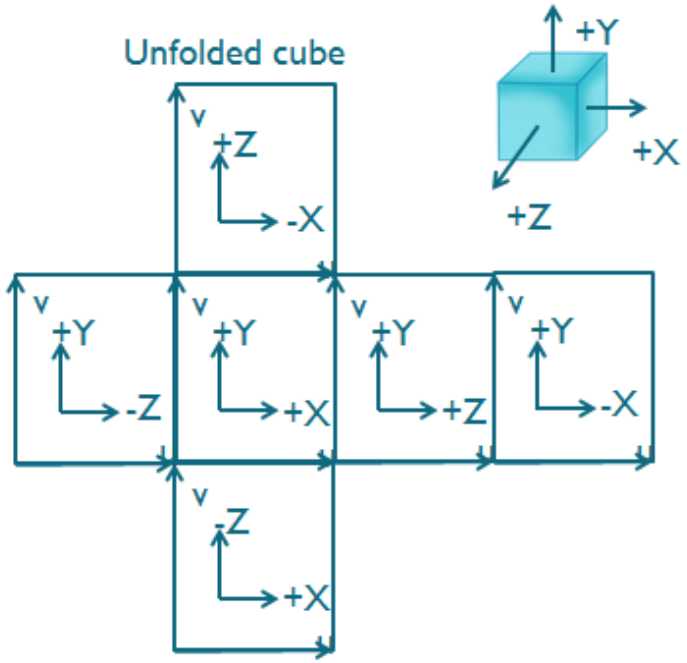
- Infinite cubemaps
  - Reflections based on infinite cubemaps
  - The concept of local cubemap and the local correction
- Generalized concept of local cubemap
- Reflections based on local cubemaps
  - Combined reflections
  - Blurred reflections
- New shadows technique: dynamic soft shadows based on local cubemaps
  - Why dynamic? Why soft?
  - Combined shadows
- Refraction based on local cubemaps
- Wrap up

Infinite cubemaps
Reflections based on infinite cubemaps

Cubemaps

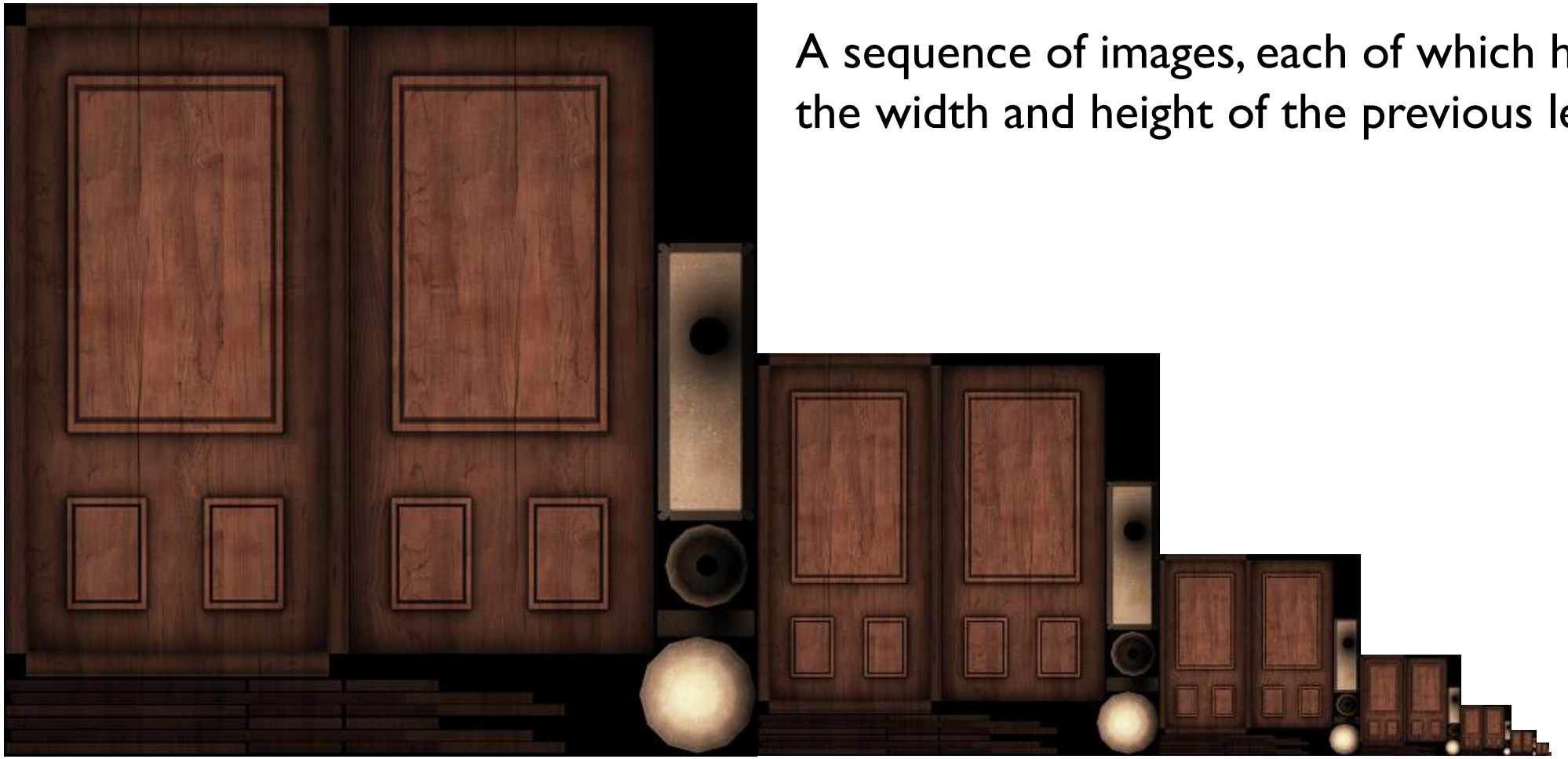
Cubemaps

- Hardware accelerated
- No image distortion
- Efficient computation



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

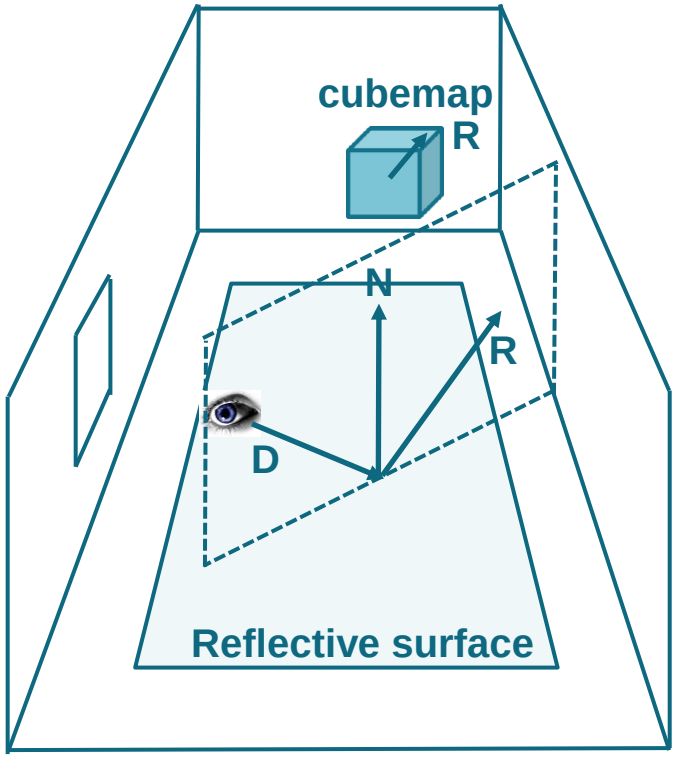
Mipmaps



512x512 256x256 128x128 64x64 … 1x1

A sequence of images, each of which halves the width and height of the previous level.

Reflections with infinite cubemaps



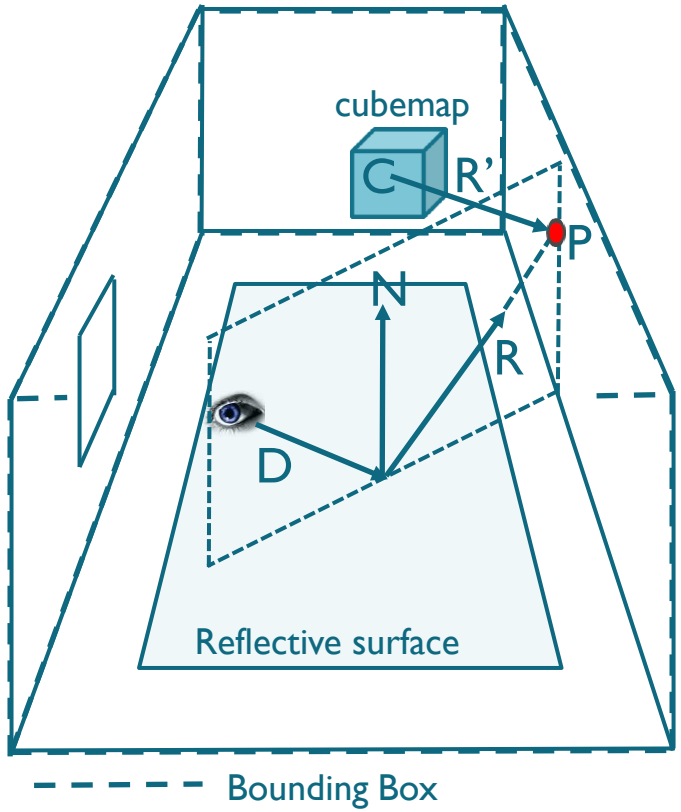
float3 R = reflect(D, N);

float4 col = texCUBE(Cubemap, R);

Incorrect reflections

Reflection generated using a cubemap without any local binding

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');

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/

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

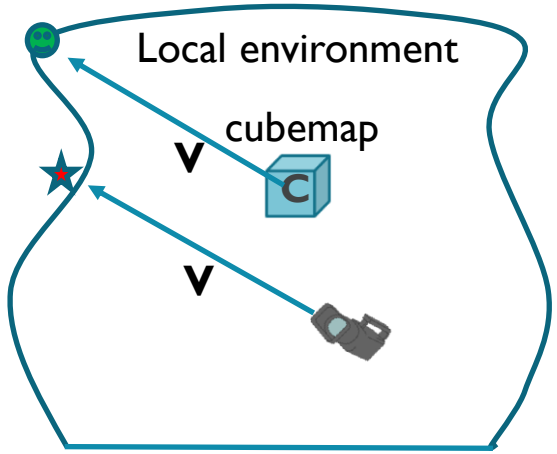
Correct and incorrect reflections

Reflection generated after applying the "*local correction*"

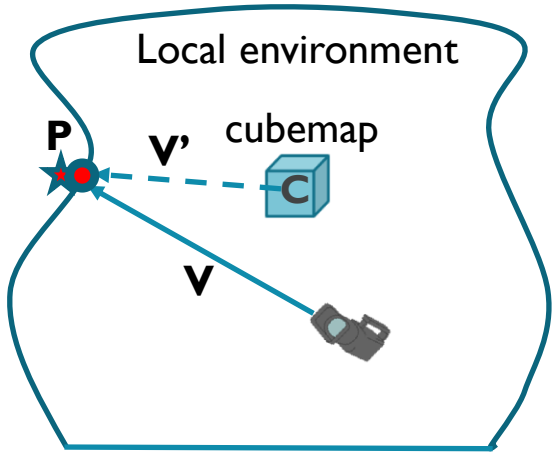
Reflection generated without "*local correction*"

Generalized concept of local cubemaps

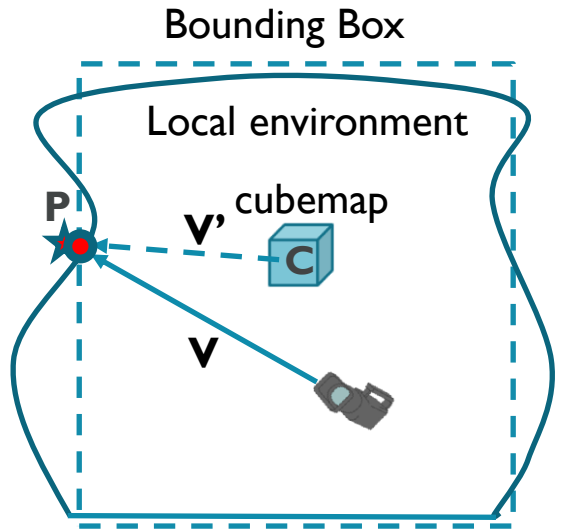
The Concept of local cubemaps



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.



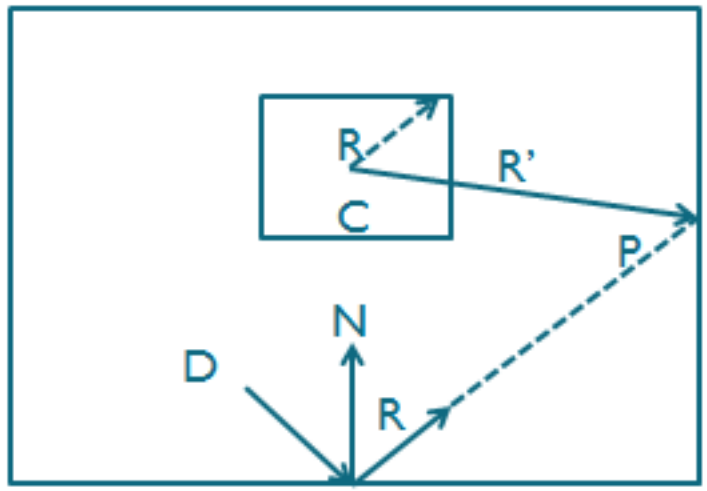
We introduce a proxy geometry to simplify the problem of finding the intersection point **P**.The simplest proxy geometry is the bounding box.

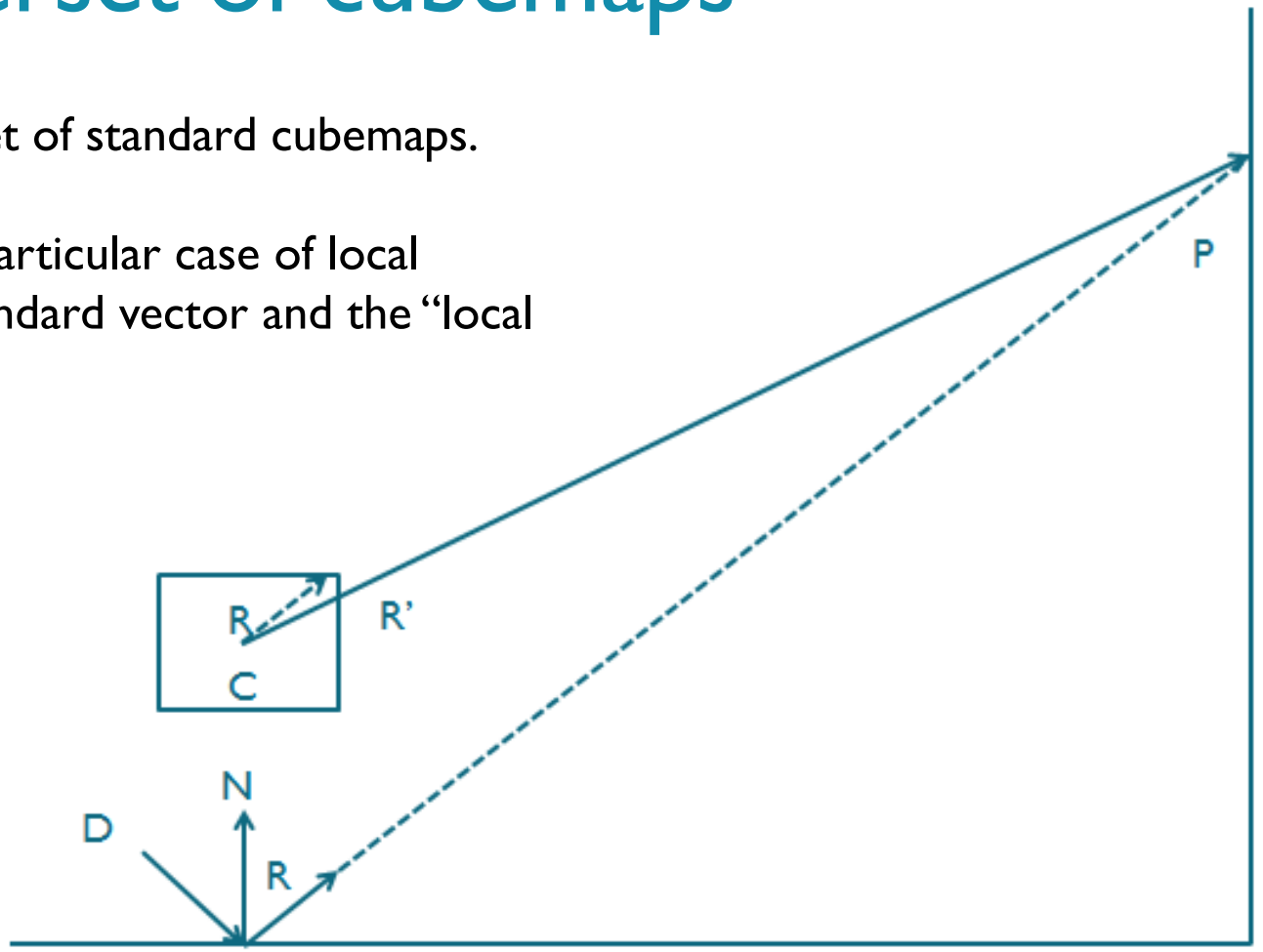
Local Cubemap = Cubemap + Cubemap Position + Scene Bounding Box + Local Correction

Local cubemaps a superset of cubemaps

We can think about local cubemaps as a superset of standard cubemaps.

The standard cubemap can be considered as a particular case of local cubemap when the BBox is so large that the standard vector and the "local corrected vector" are the same.





The larger the BBox the more R' approaches to R.

Local cubemaps a superset of cubemaps

$$\boldsymbol{Infinite\ Cubemap} = \lim_{BBoxSize \to \infty} \boldsymbol{Local\ Cubemap}$$

The local correction can be seen as just the correct way of fetching the texture in the local cubemap!

Reflections based on local cubemaps

Benefits and limitations

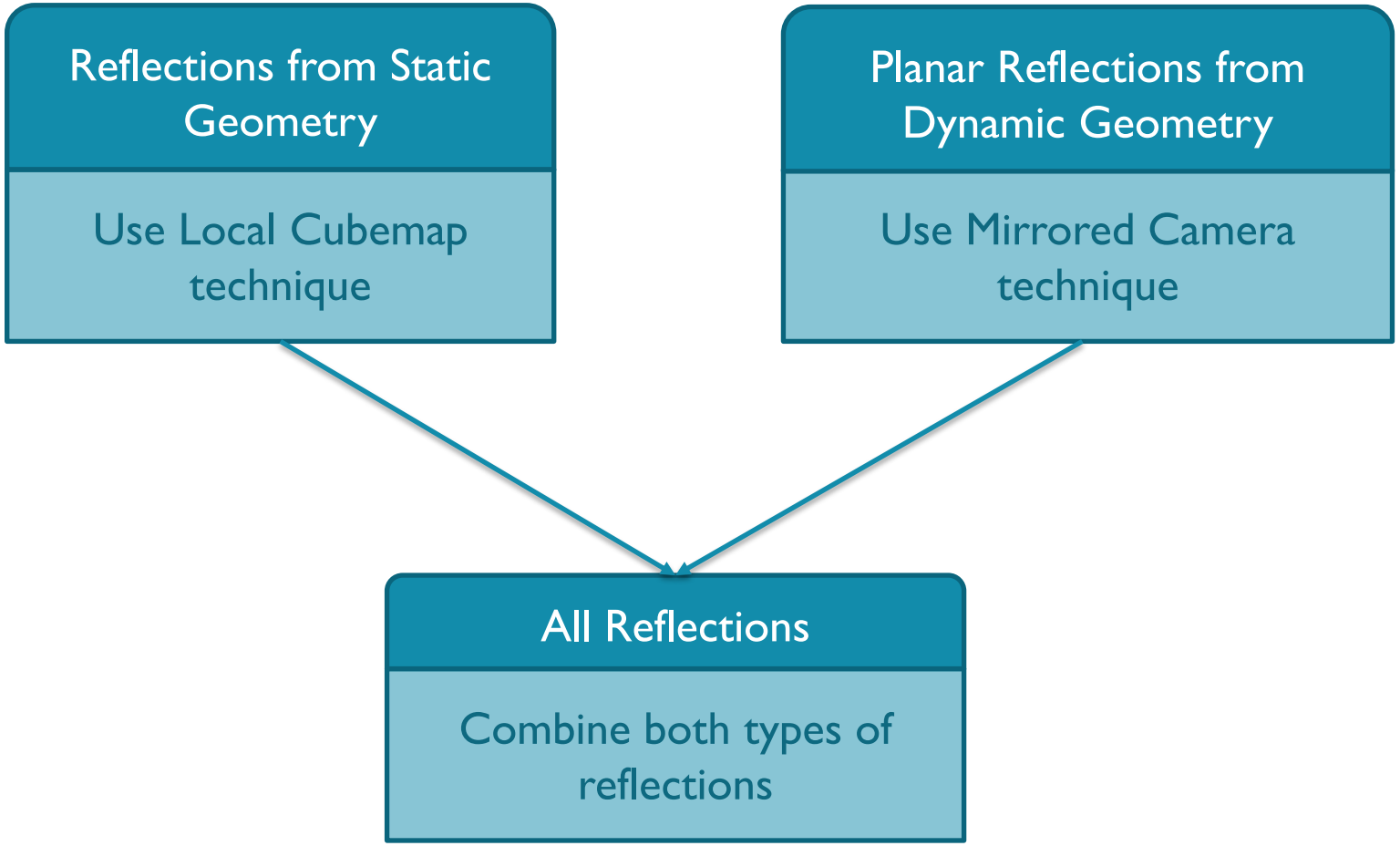
Benefits

1. Simple to implement
2. Very realistic
3. Physically correct
4. High quality of reflections, no pixel flickering/ instability when moving camera
5. Cubemap texture can be compressed
6. Offline filtering effects can be applied which could be very expensive at run time
7. Resource saving technique, important for mobile devices

Limitations

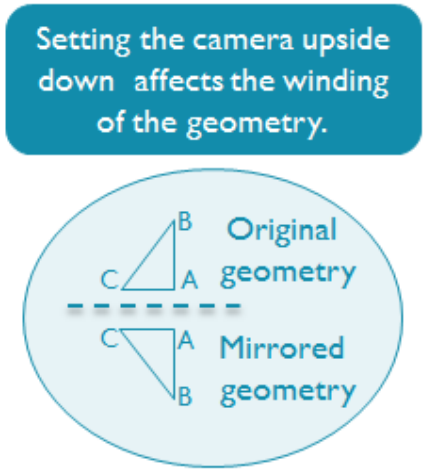
1. Only works well in open plan space with no geometry in the centre where the cubemap will likely be generated
2. Objects in the scene must be close to the proxy geometry for good results
3. Only effective for simulating reflections of static geometry

Handling reflections of dynamic objects

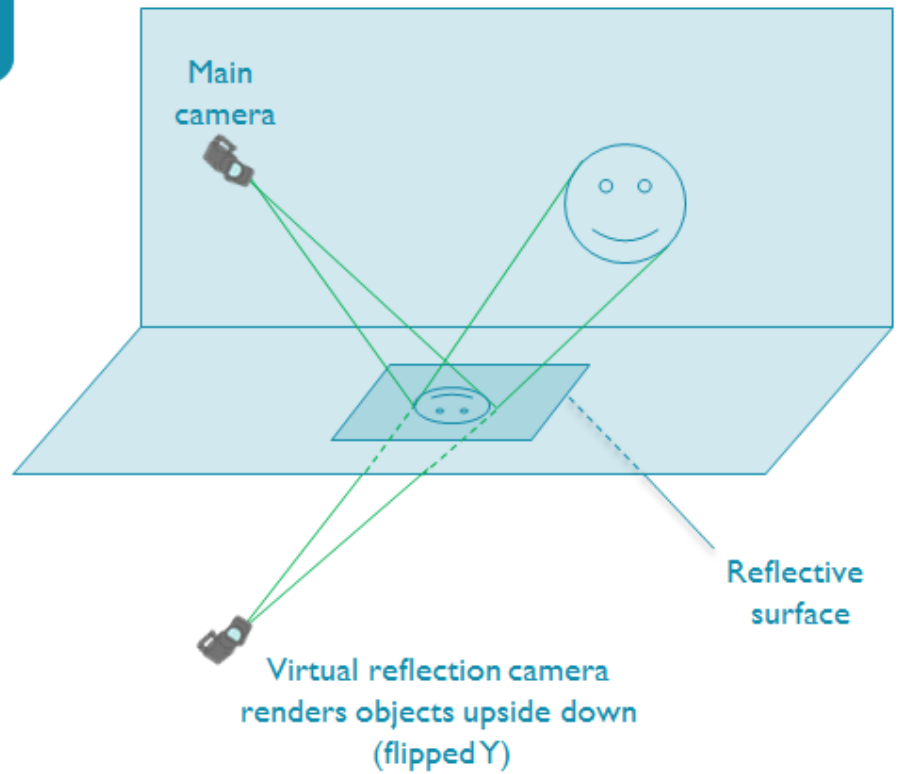


Runtime planar reflections with a mirrored camera

Setting the camera upside down affects the winding of the geometry.



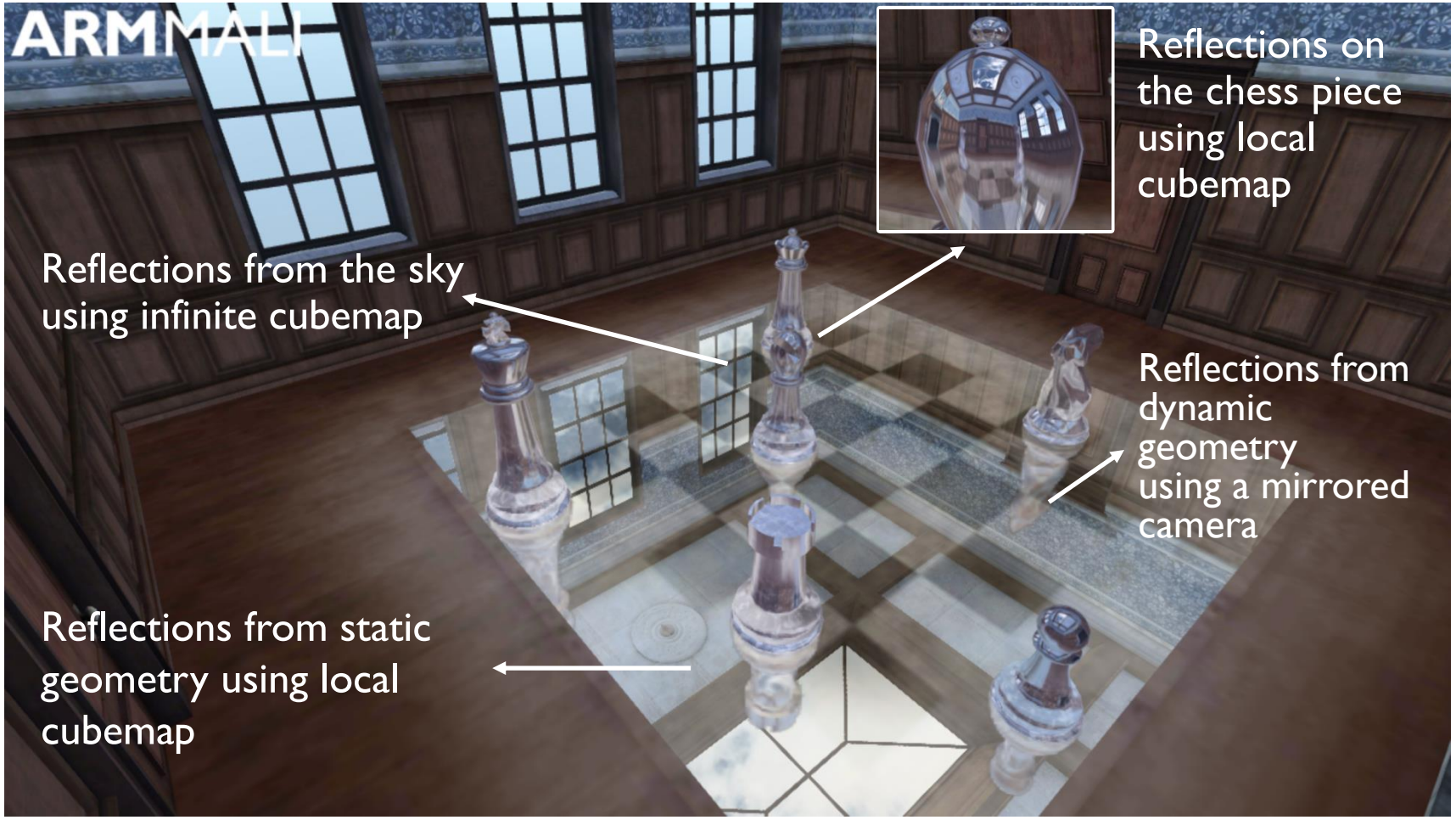
We need to reverse geometry when rendering with the reflection camera to fix the winding order.



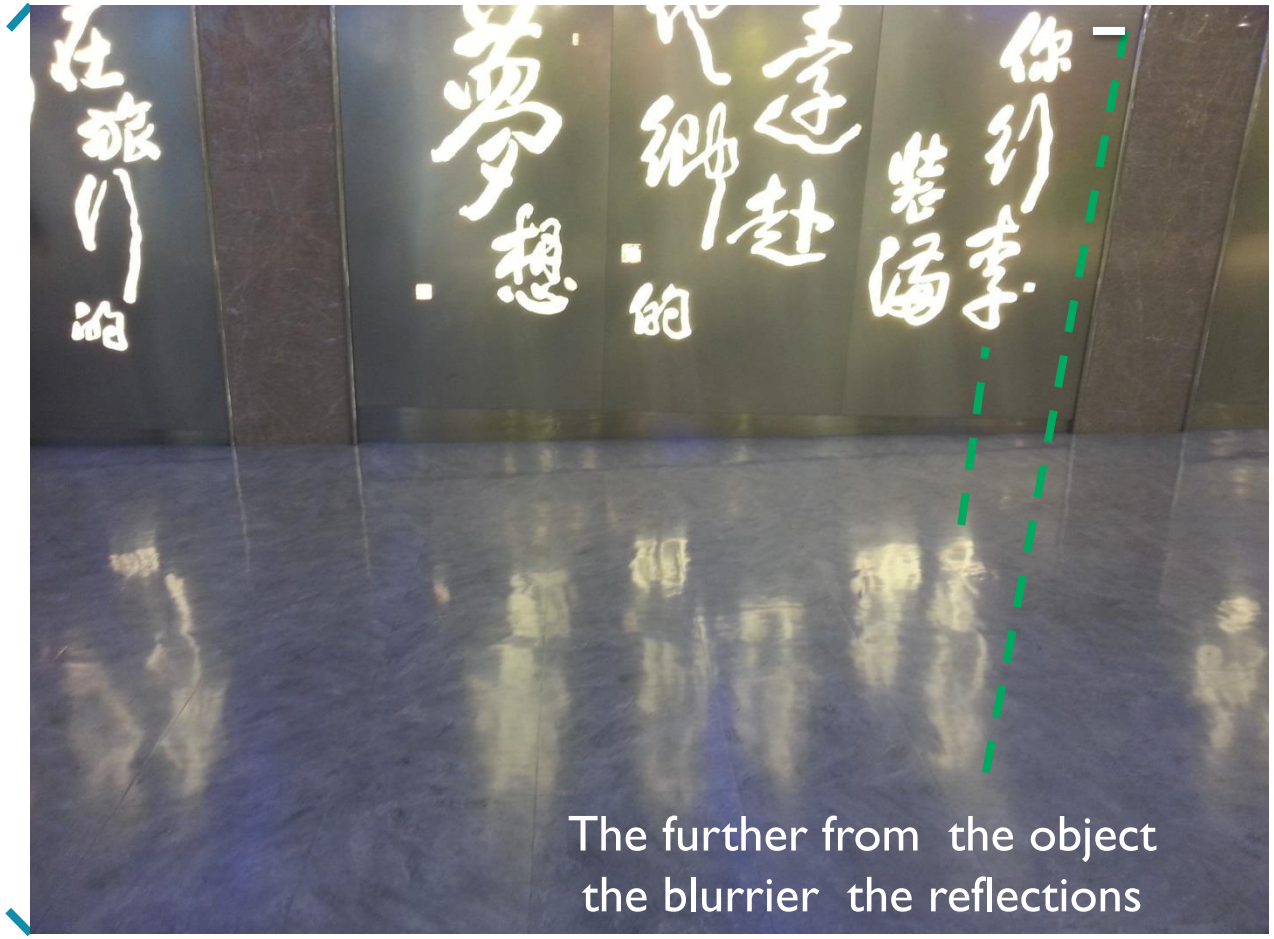
ARM

Combined reflections

Combined reflections in the Chess Room demo

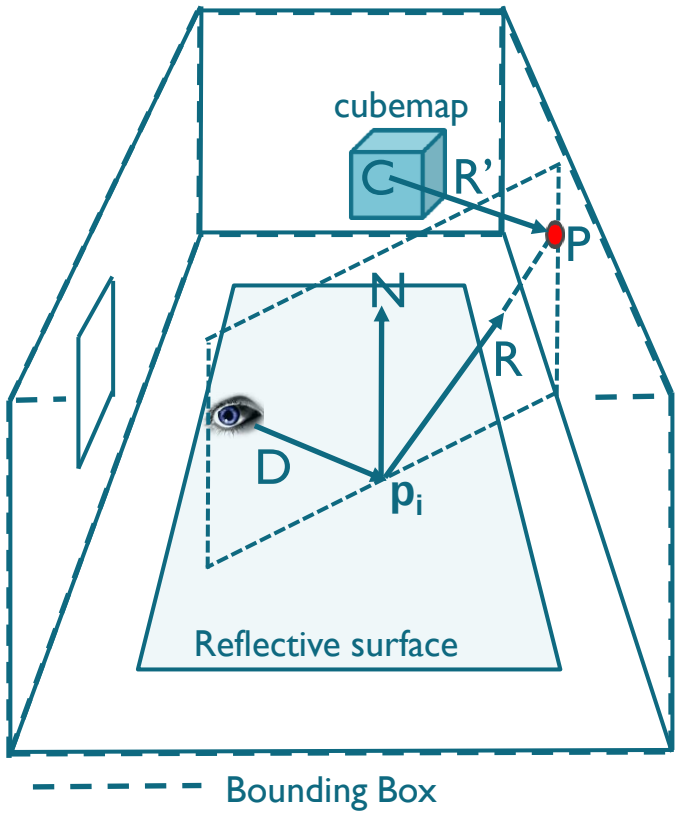


Blurred reflections at the Taiwan intern. airport



ARM

Implementing blurred reflections with local cubemaps



float3 R = reflect(D, N);

Find intersection point P
Find vector R' = CP

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

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

float4 col = texCUBElod(Cubemap, newVec);

ARM

Blurred reflections based on local cubemaps

Why use reflections based on local cubemaps?

Advantages over runtime reflections

1. Up to 2.8 times faster.
2. Resource saving. Bandwidth halved as only read op.
3. Higher quality. No pixel flickering.

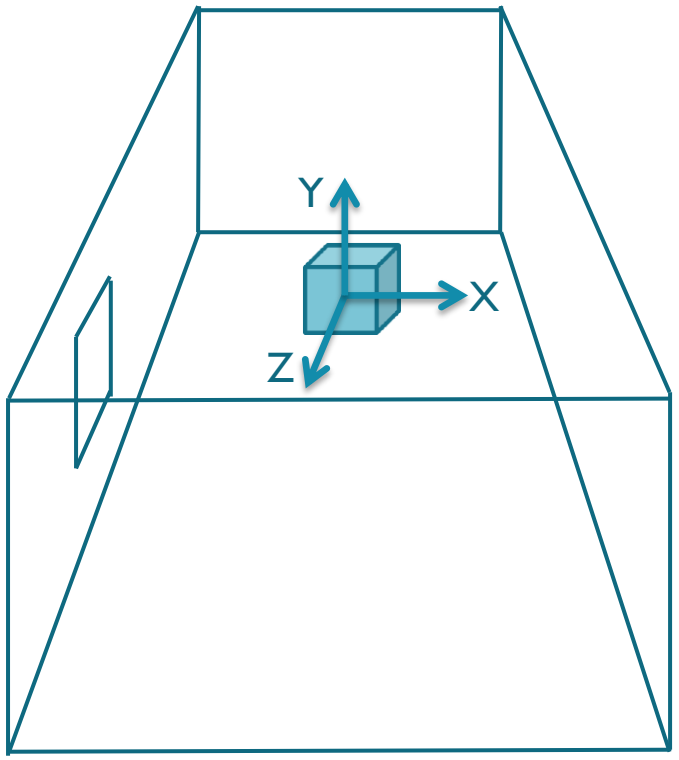
When possible use reflections based on LC

When combined with runtime reflections it helps improving quality at low cost

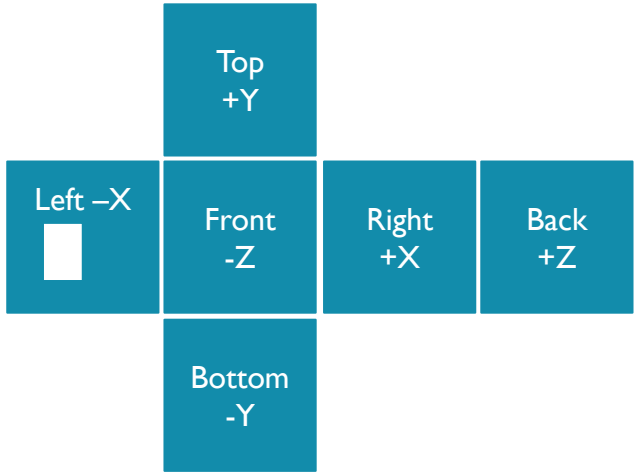
Dynamic soft shadows based on local cubemaps

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

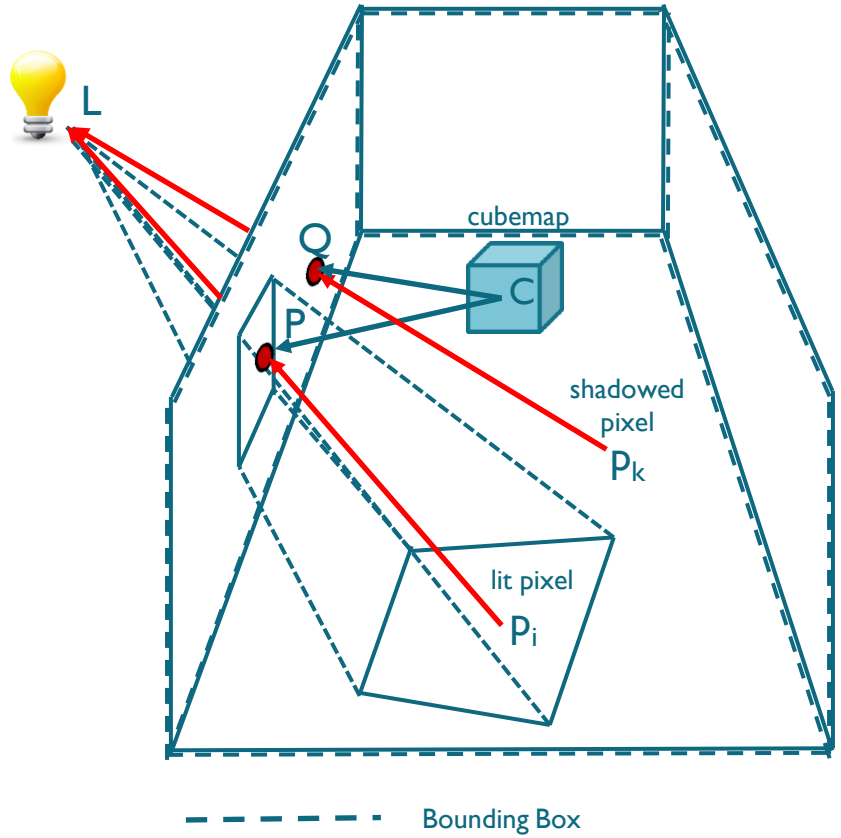
Fully transparent geometry is rendered with alpha 0

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.

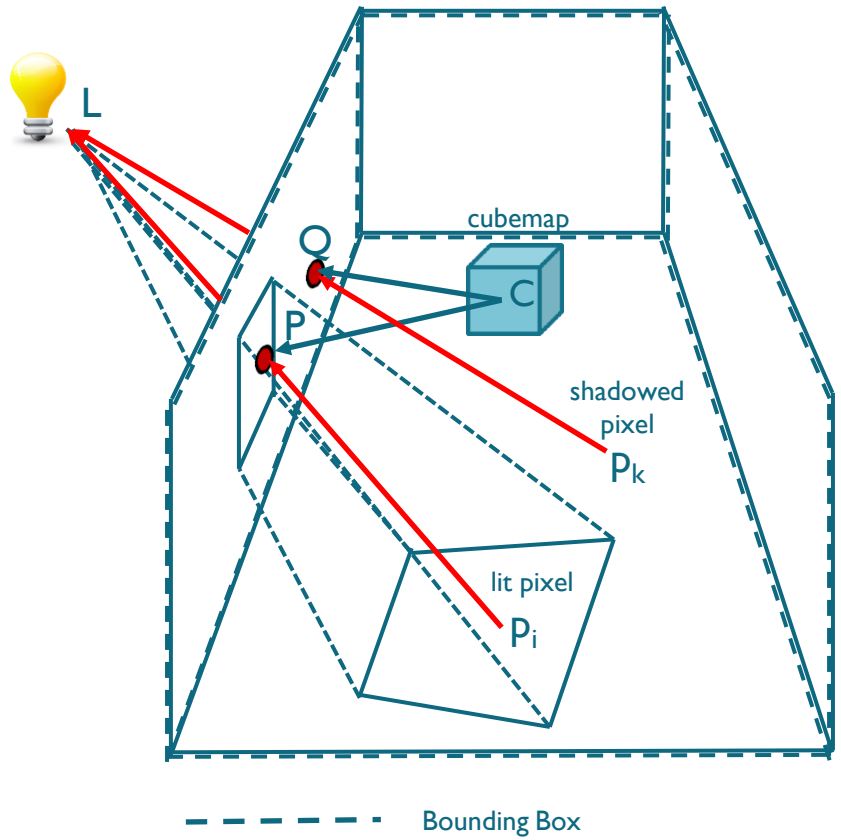
```
float texShadow = texCUBE(_CubeShadows,  CP).a;
```

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

Dynamic soft shadows based on local cubemaps

Why dynamic? Why soft?

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



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

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

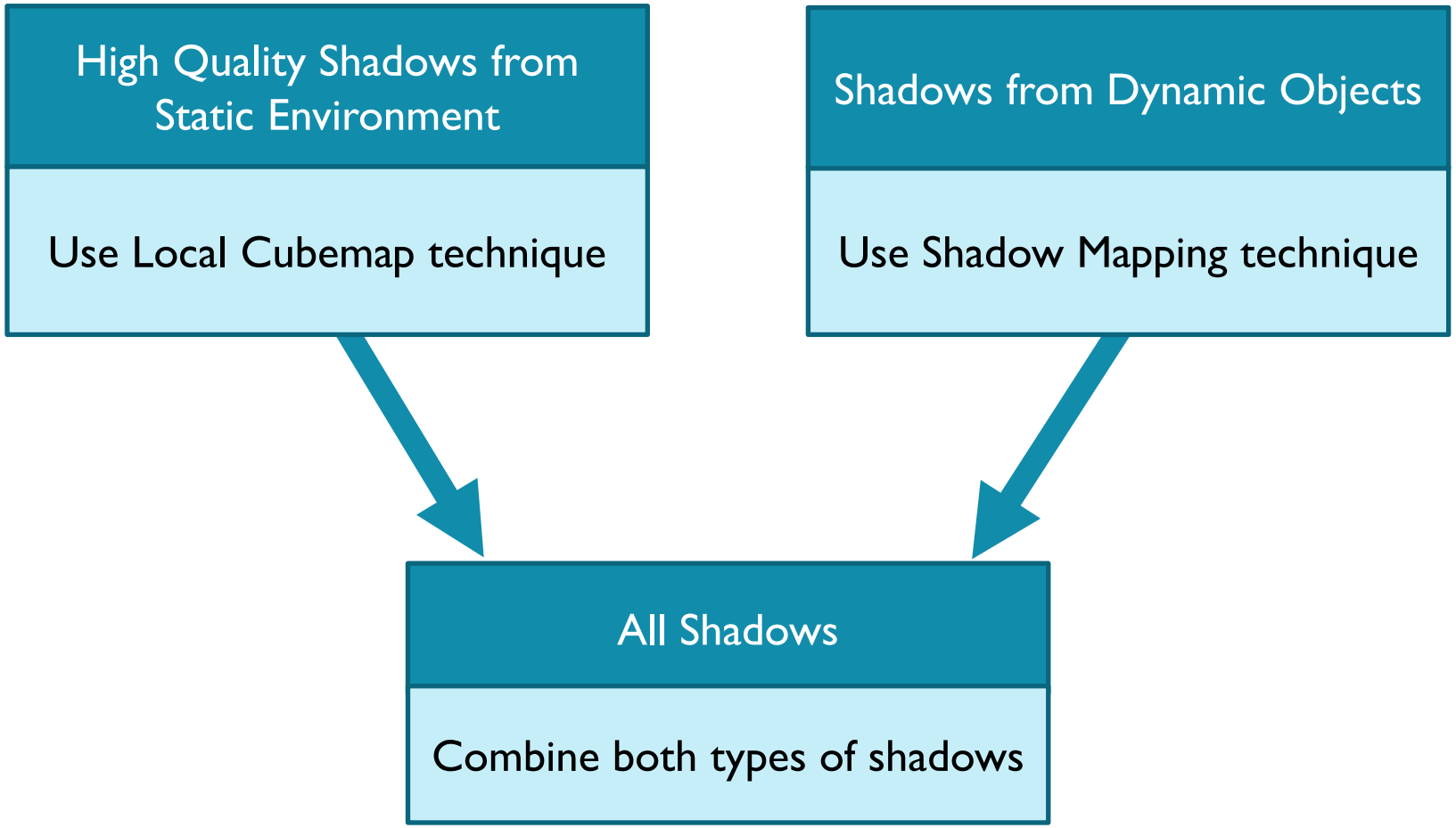
The further from the object the softer the shadows.

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

ARM

Dynamic soft shadows based on local cubemaps

Handling shadows from different types of geometries



Combined shadows

Combined shadows in Ice Cave demo

Why use shadows based on local cubemaps?

Advantages over runtime shadow mapping

1. Up to 1.5 times faster.
2. Resource saving. Bandwidth halved as only read op.
3. Higher quality. No pixel flickering.

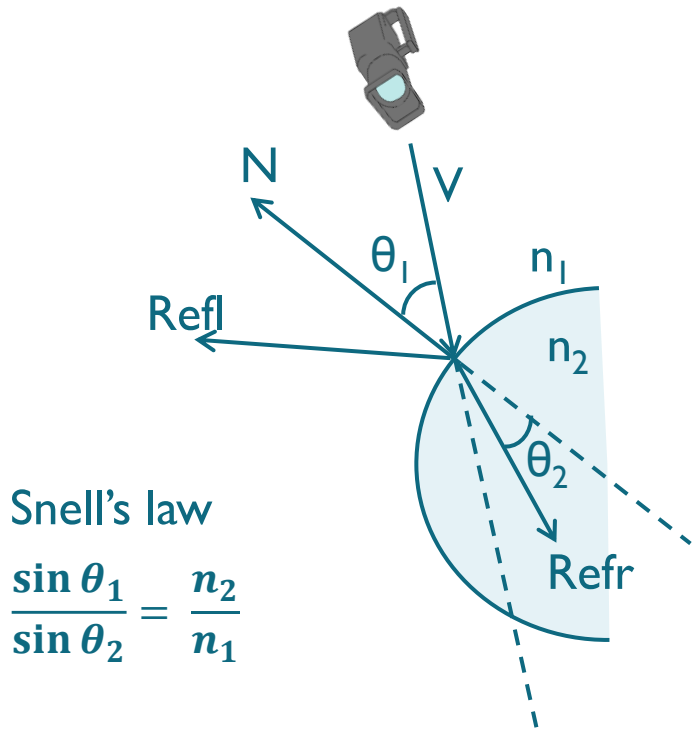
When possible use shadows based on LC

When combined with runtime shadow mapping it helps improving quality at low cost

Download the complete Dynamic Soft Shadows Based on Local Cubemaps project for free from the Unity Asset Store

Refraction based on local cubemaps

Refraction



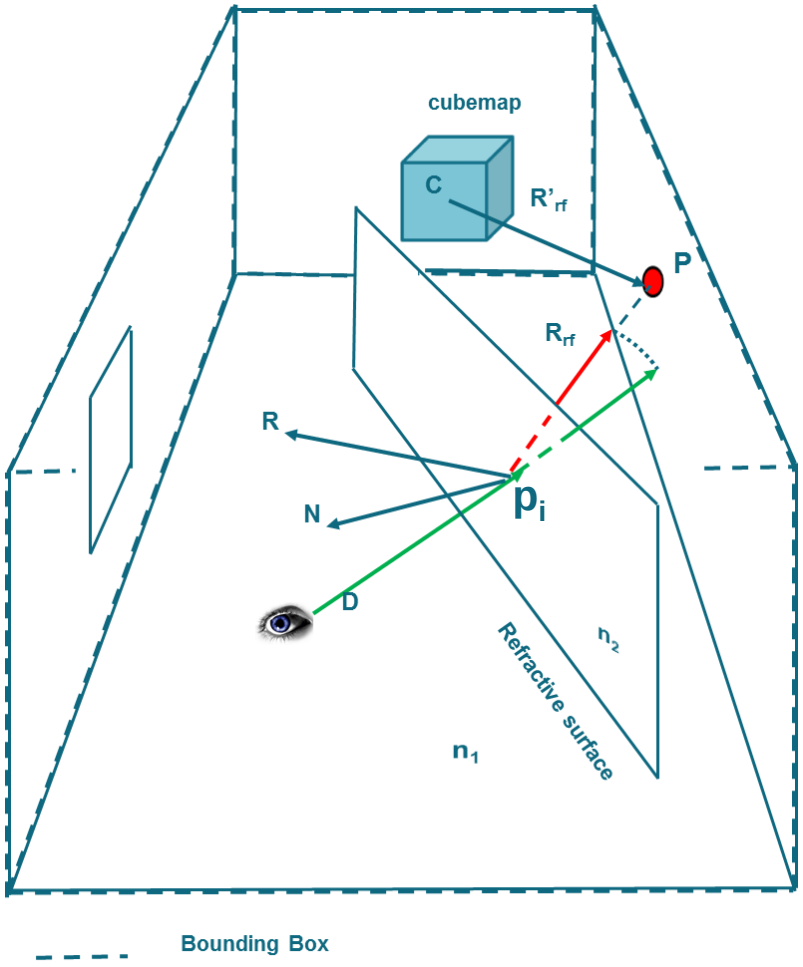
Snell's law

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

Refraction

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

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

Refraction based on local cubemaps in Ice Cave demo

~~*float4 col = texCUBE(Cubemap, R'$_{rf}$);*~~

*float4 newVec = float4(CP, factor * length(p$_i$P))*

float4 col = texCUBElod(Cubemap, newVec);

Demo Ice Cave

Watch the video at https://www.youtube.com/watch?v=mb98QOIZ8ZE

Wrap up

- The concept of infinite cubemap has evolved to a more generalized concept of local cubemap allowing the development of highly optimized rendering technique particularly suitable for mobile devices where runtime resources must be carefully balanced.

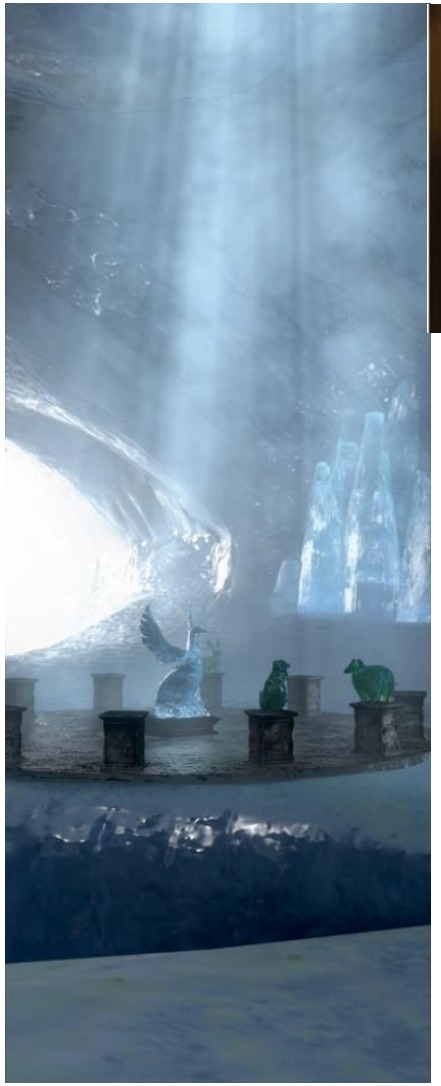
- New rendering techniques based on local cubemaps can be effectively combined with other runtime techniques to render different effects for static and dynamic objects together.

Thank you

ARM

Questions

To find out more….



- Find out more about techniques based on local cubemaps at:
  - https://www.assetstore.unity3d.com/en/#!/content/61640
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap
  - http://malideveloper.arm.com/documentation/developer-guides/arm-guide-unity-enhancing-mobile-games/
  - https://community.arm.com/groups/arm-mali-graphics/blog/2016/03/10/combined-reflections-stereo-reflections-in-vr
  - https://community.arm.com/groups/arm-mali-graphics/blog/2016/07/05/stereo-reflections-in-unity-for-google-cardboard
  - https://community.arm.com/groups/arm-mali-graphics/blog/2016/04/20/achieving-high-quality-mobile-vr-games
  - 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