Achieving High Quality Mobile VR Games

ARM

Roberto Lopez Mendez, Senior Software Engineer

Carl Callewaert - Americas Director
& Global Leader of Evangelism, Unity

Patrick O'Luanaigh – CEO, nDreams

GDC 2016

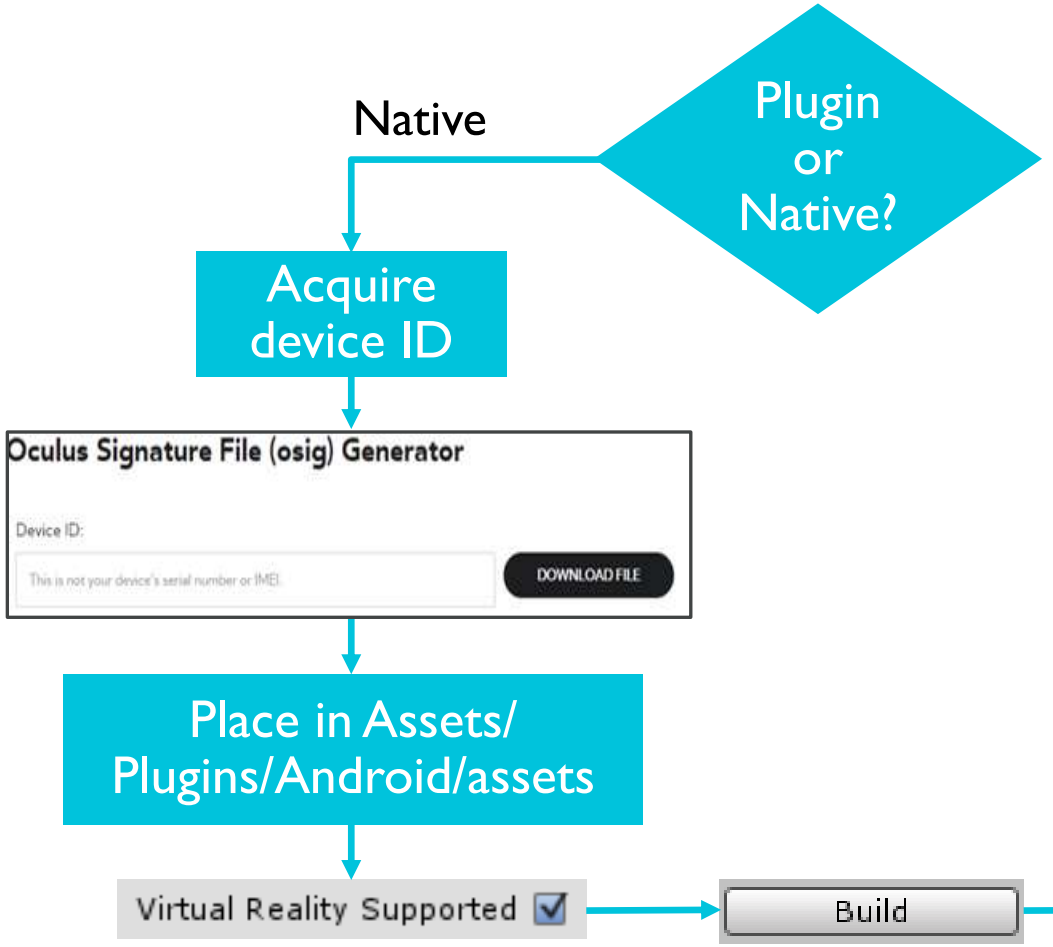
Agenda

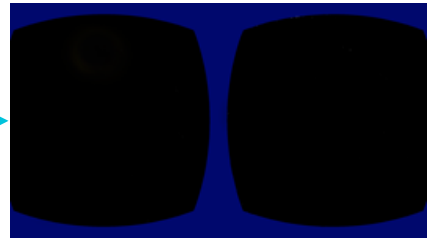
- Ice Cave Demo (Roberto)
  - Porting Ice Cave Demo to Samsung Gear VR
  - Improving VR quality & performance
    - Dynamic soft shadows and reflections based on local cubemaps
    - Stereo reflections
- VR Integration into Unity (Carl)
- Movement in Mobile VR (Patrick)

Demo Ice Cave

Porting Ice Cave Demo to Samsung Gear VR

Porting Ice Cave for Samsung Gear VR





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

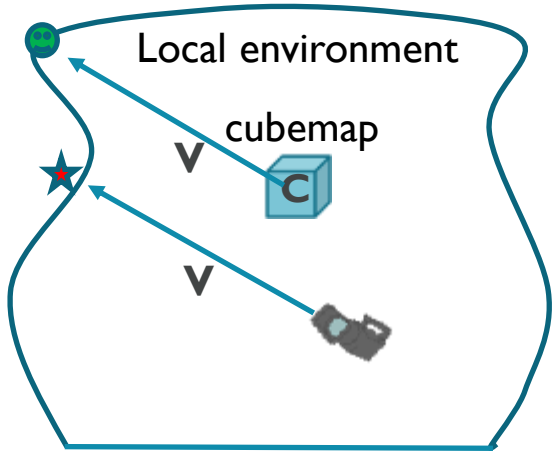
Ice Cave VR Extras

- 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

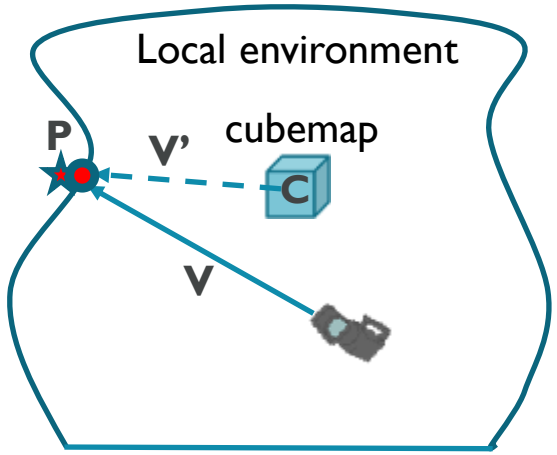
Quality and Performance for VR

Optimized Rendering Techniques Based On Local Cubemaps

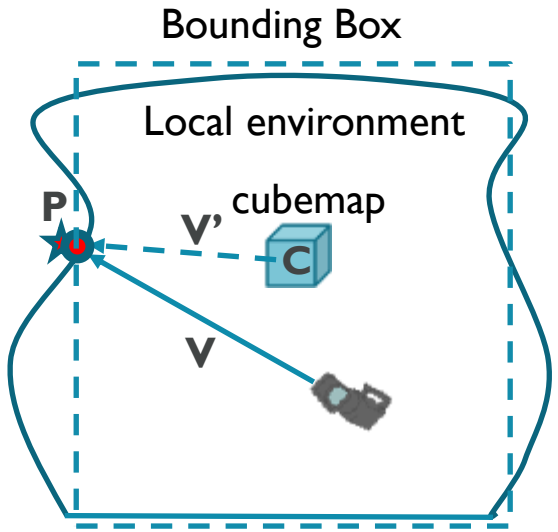
The Concept of Local Cubemaps



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.

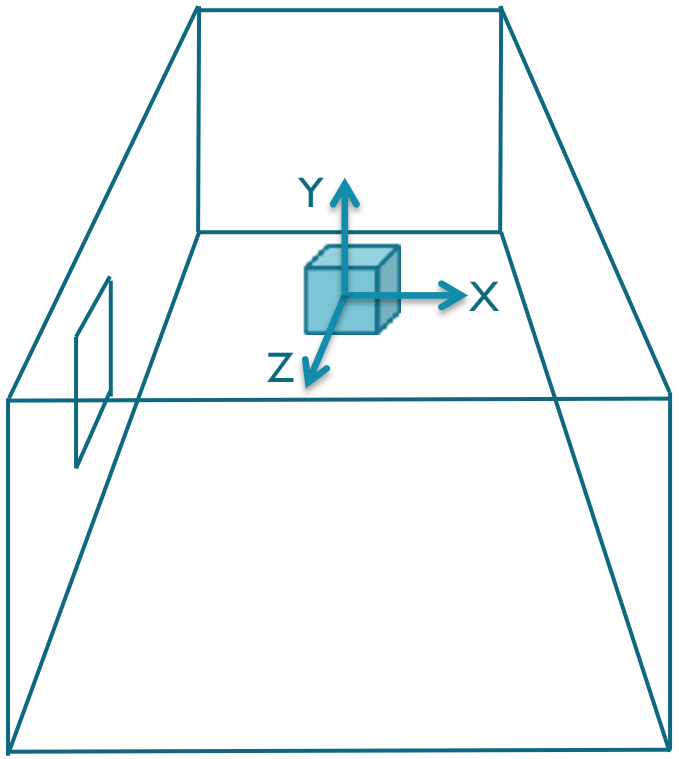
Local Cubemap = Cubemap + Cubemap Position + Scene Bounding Box + Local Correction

ARM

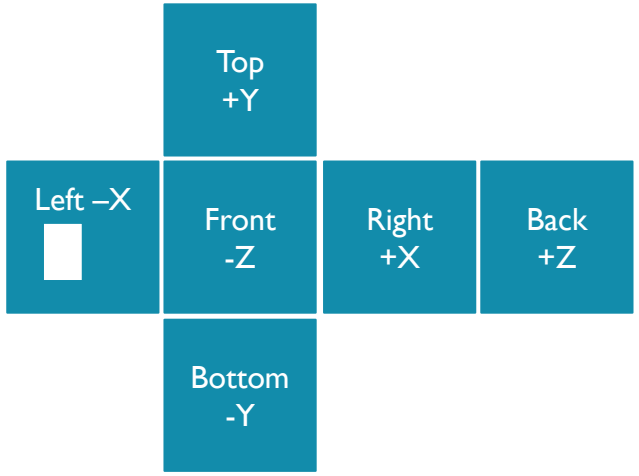
Dynamic Soft Shadows Based on Local Cubemaps

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

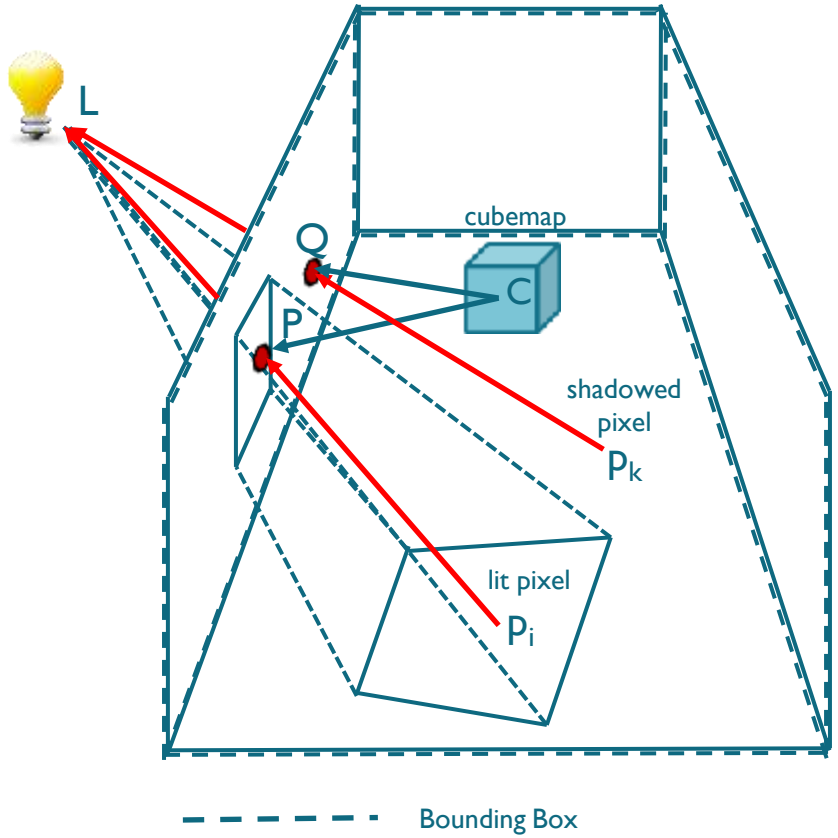
Fully transparent geometry is rendered with alpha 0

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 $\mathbf{p_iL}$.
- 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.

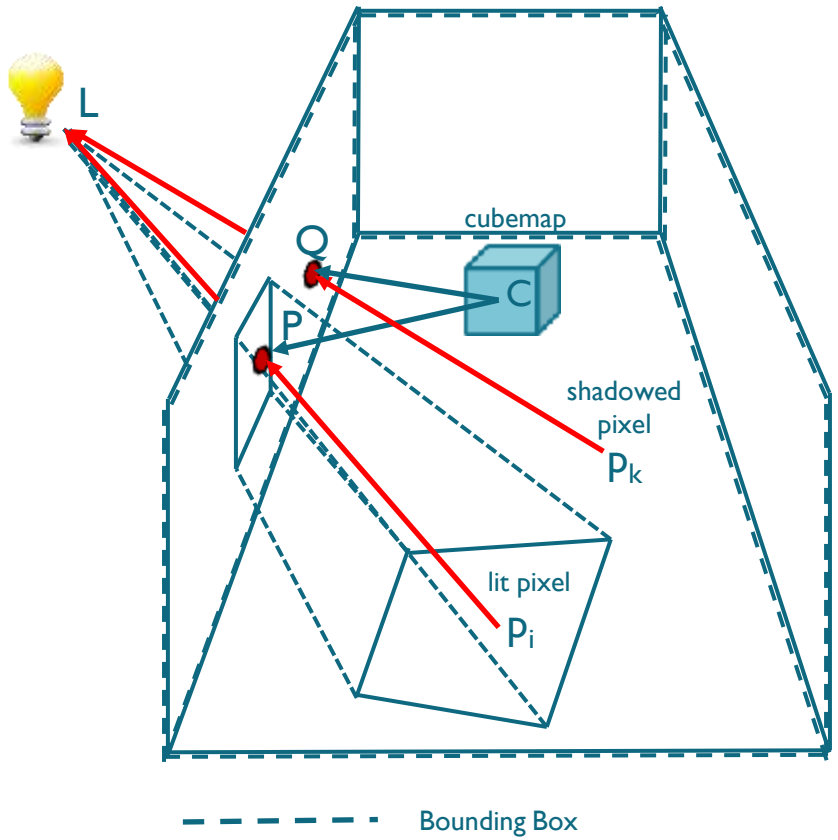
```
float texShadow = texCUBE(_CubeShadows,  CP).a;
```

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

Dynamic Soft Shadows Based on Local Cubemaps

Why soft?

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



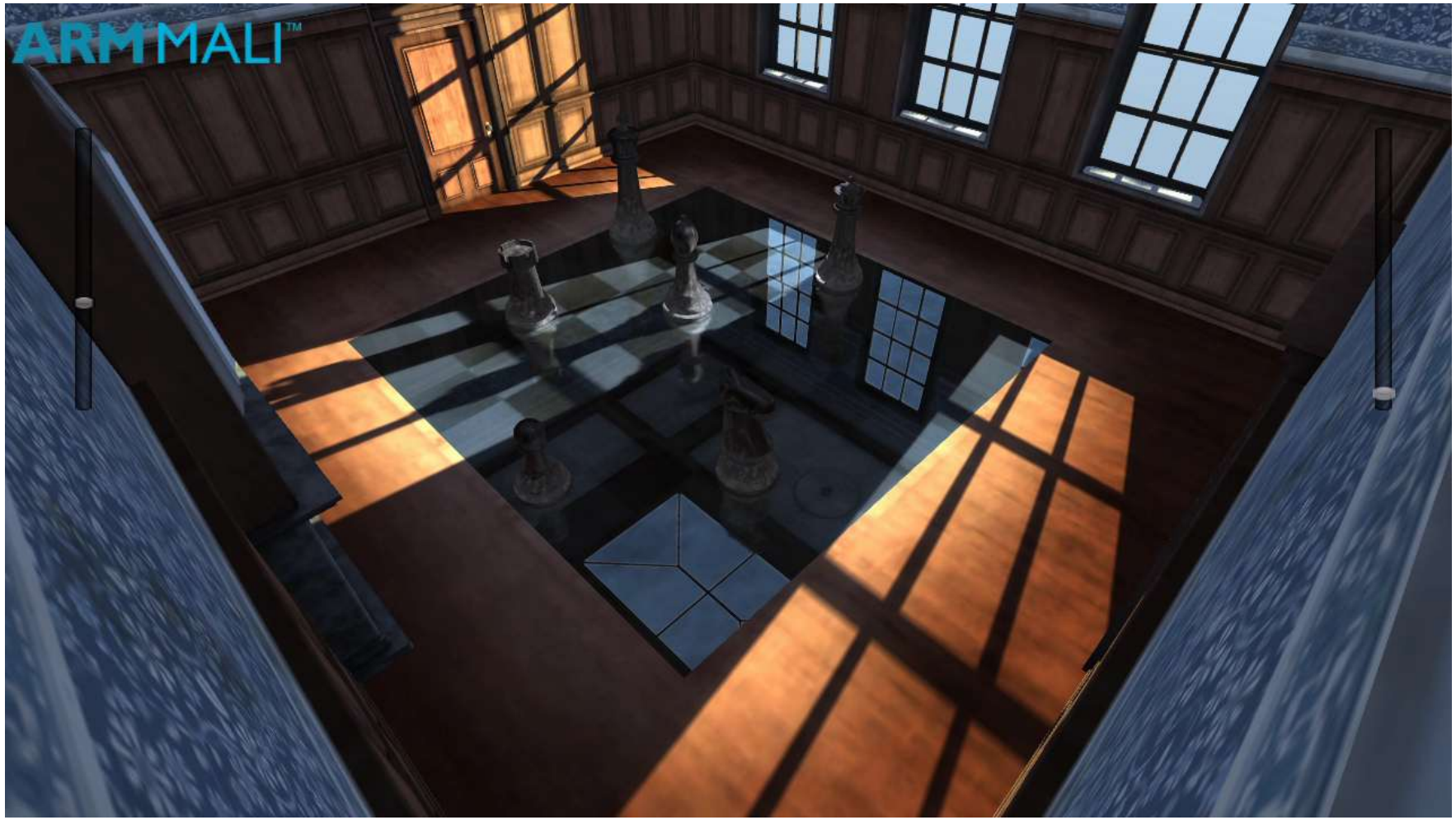
float4 newVec = float4`(CP, factor * length(p_iP))

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

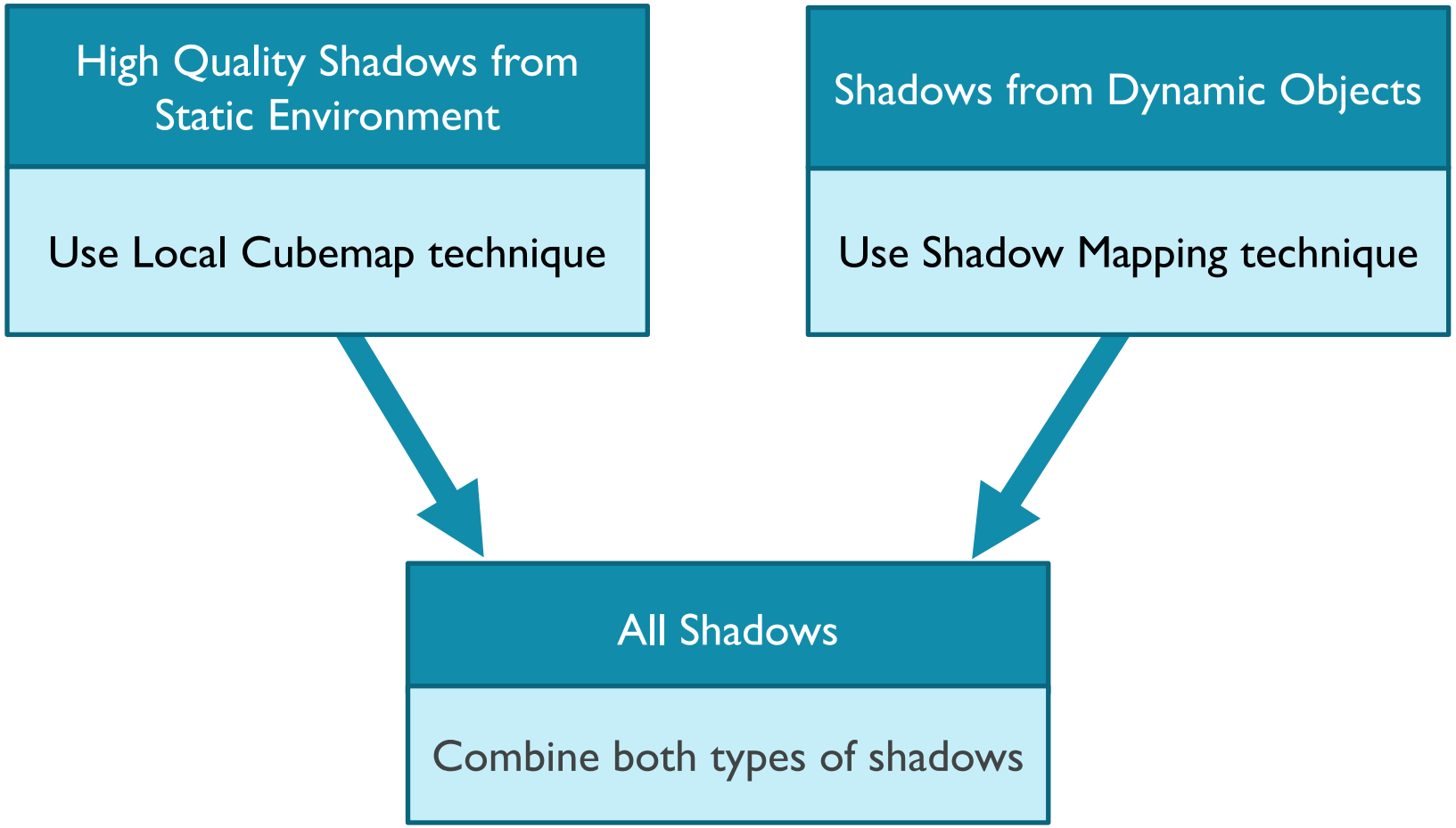
The further from the object the softer the shadows.

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

Dynamic Soft Shadows Based on Local Cubemaps



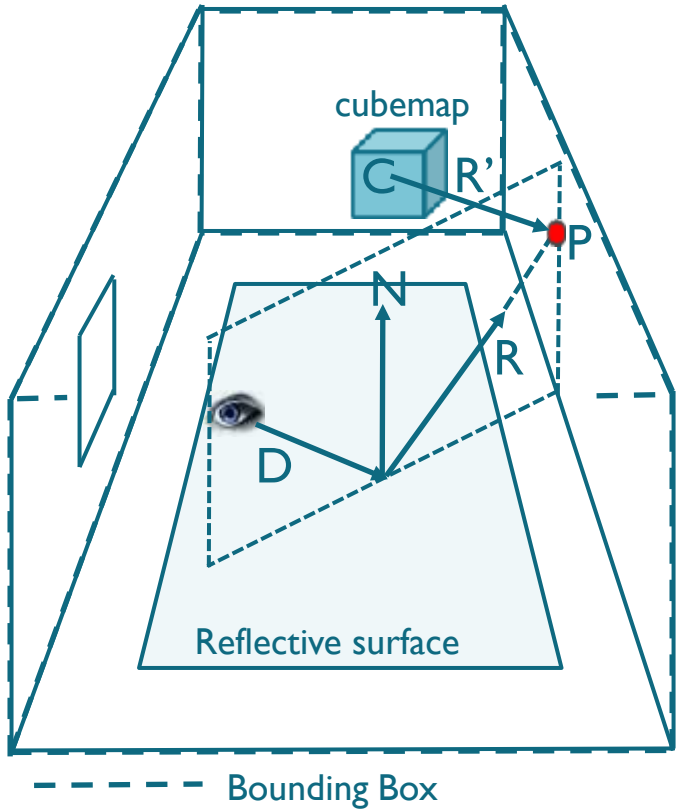
Handling Shadows from Different Types of Geometries



Combined Shadows in Ice Cave VR

Reflections Based on Local Cubemaps

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

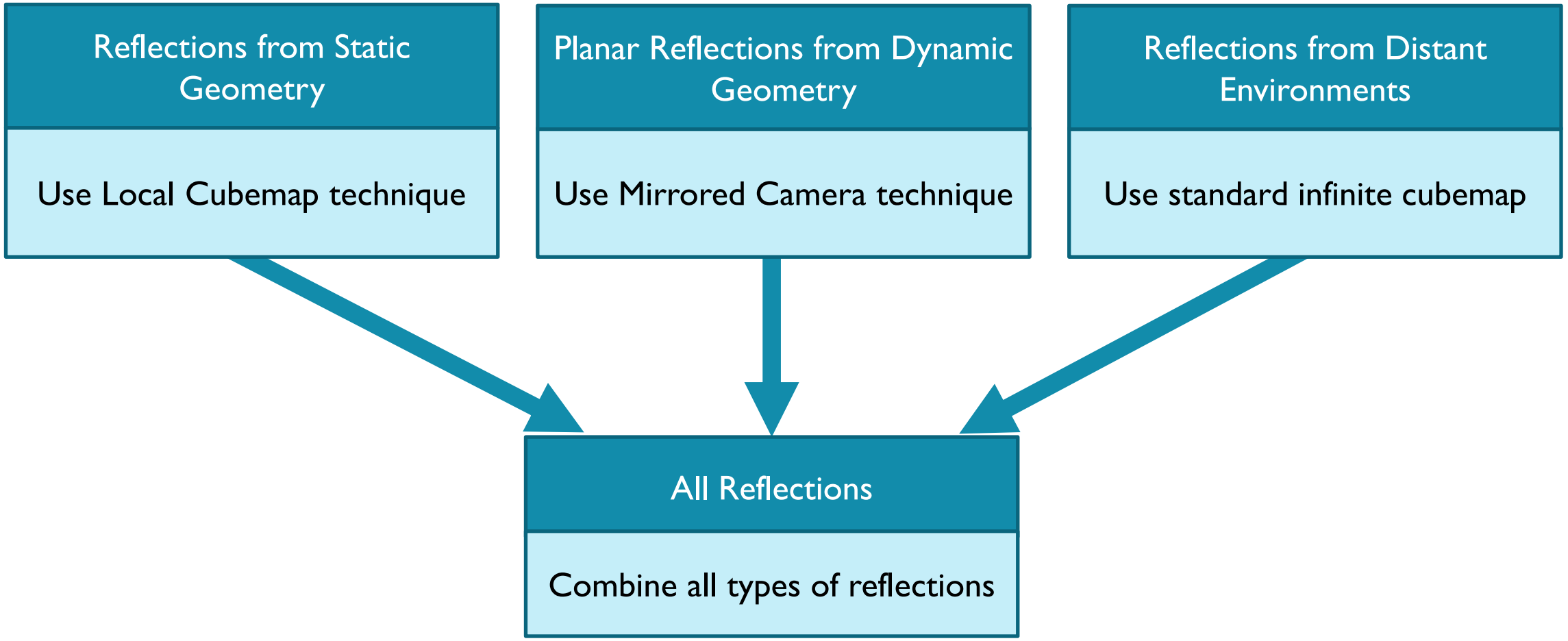
Cubemap Environment Mapping. 2010.
http://www.gamedev.net/topic/568829-box-projected-cubemap-environment-mapping/?&p=4637262

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

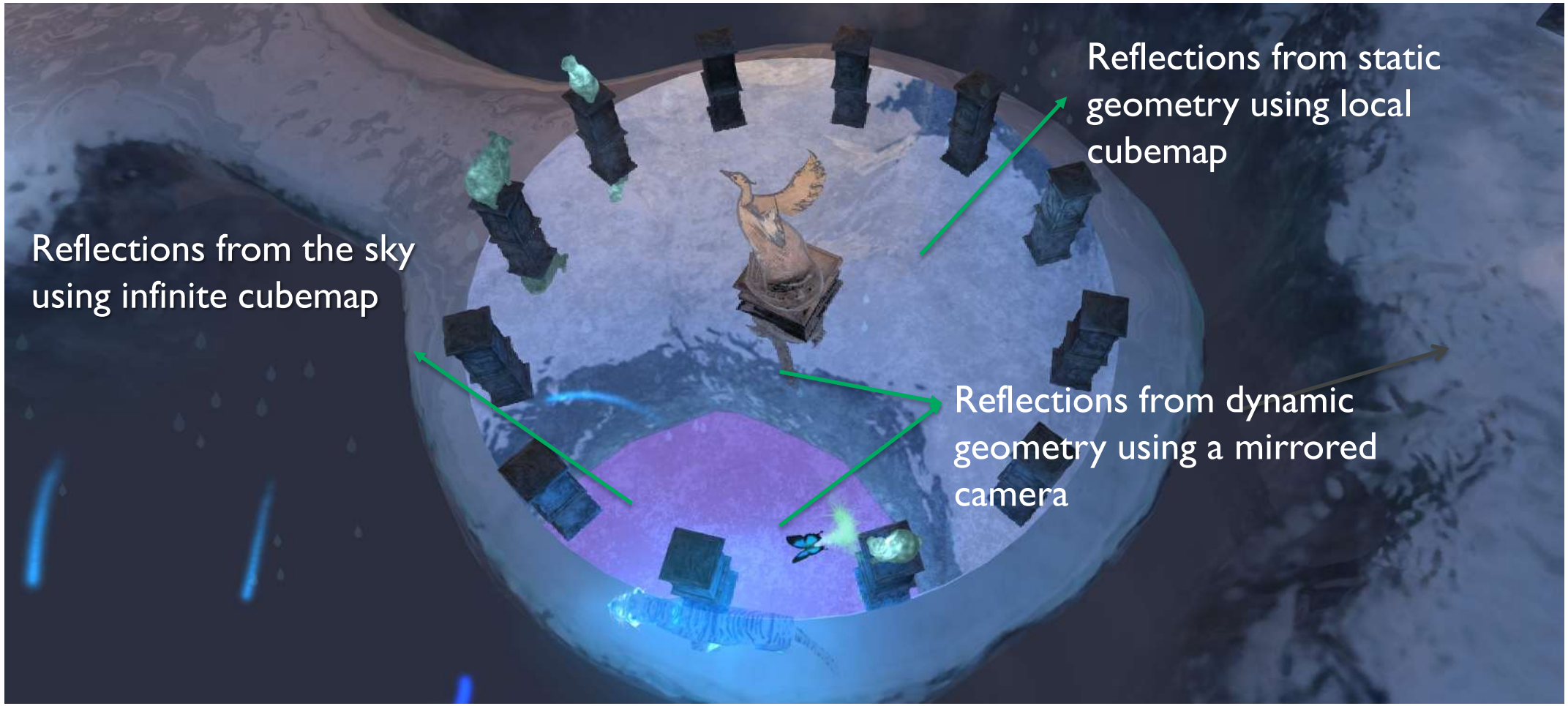
ARM

Combined Reflections in Ice Cave VR



ARM

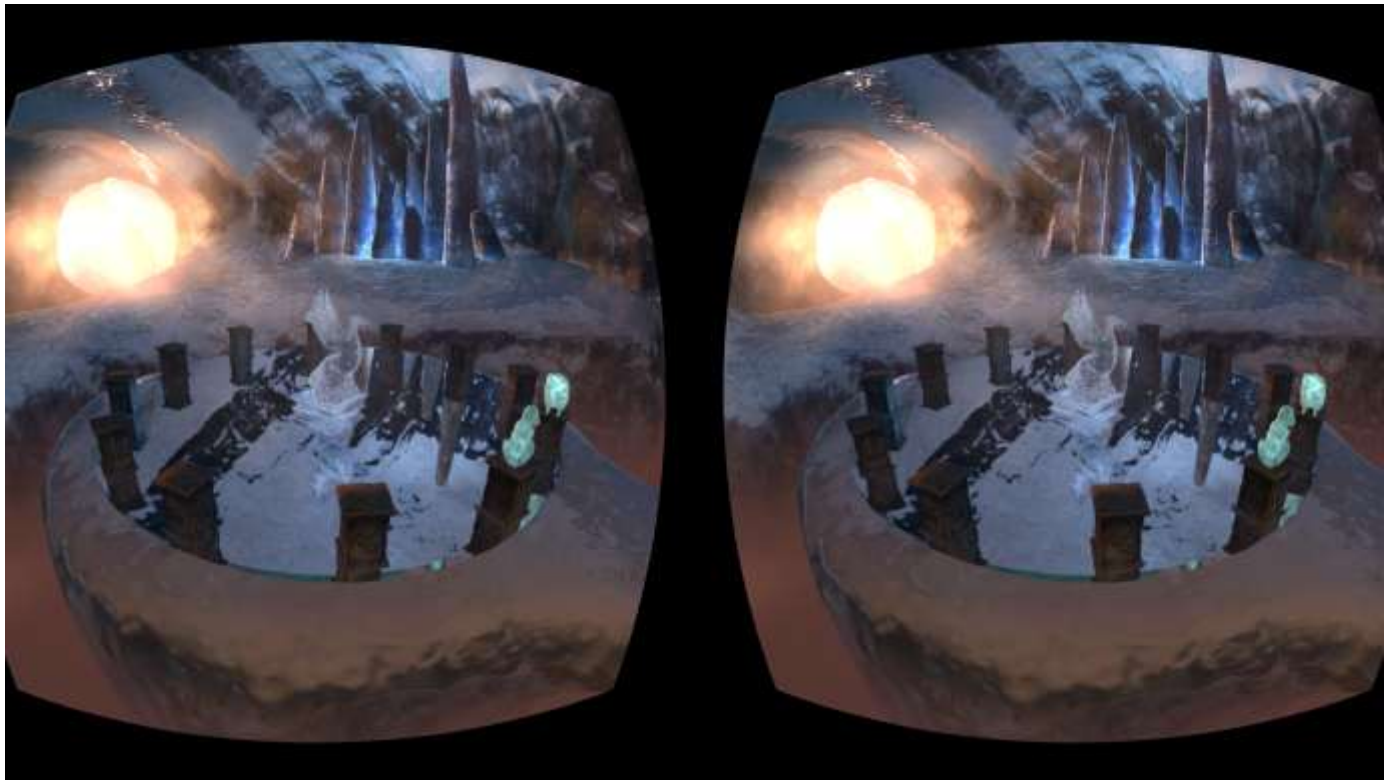
Combined Reflections in Ice Cave VR



Stereo Reflections in Unity

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;
}
```

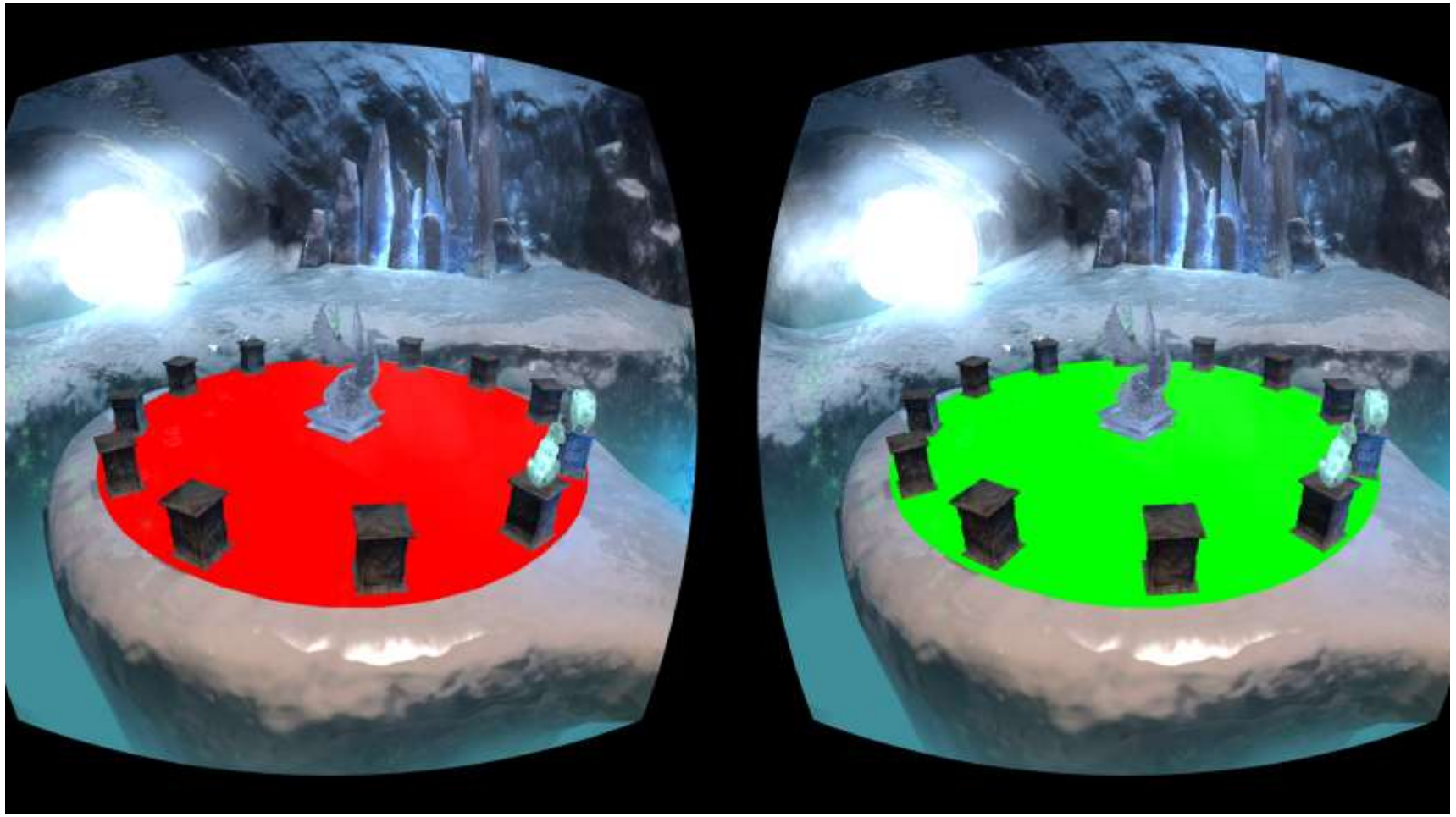
Implementing Stereo Reflections in Unity

Attach the script to the main camera

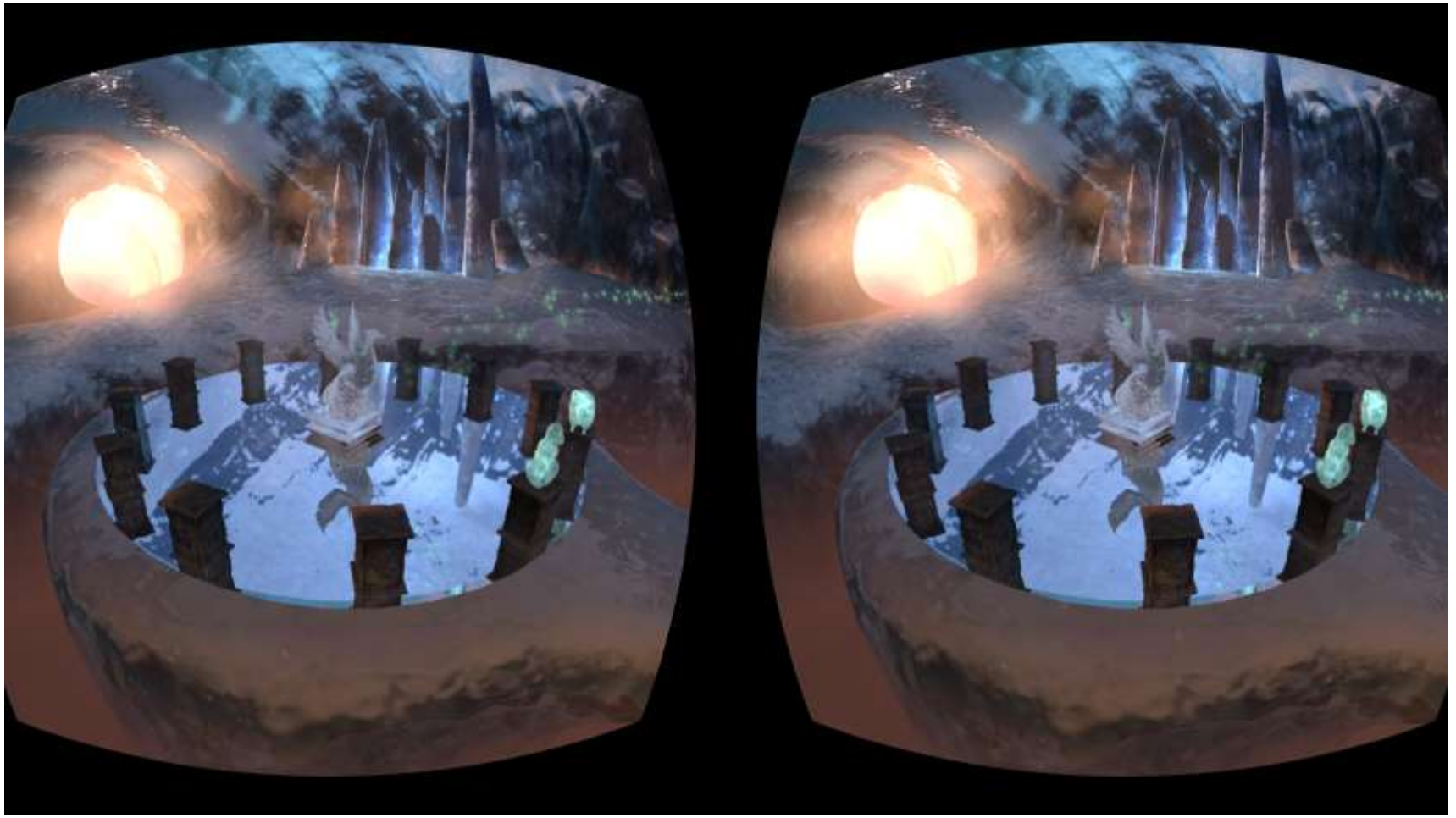
```
public class RenderStereoReflections : MonoBehaviour
{
        public GameObject reflectiveObj;
        public GameObject leftReflCamera;
        public GameObject rightReflCamera;
        int eyeIndex = 0;

        void OnPreRender(){                    ← Optimize further by adding a condition about reflective object's visibility
                if (eyeIndex == 0){
                        // Render Left camera
                        leftReflCamera.GetComponent<Camera>().Render();
                        reflectiveObj.GetComponent<Renderer>().material.SetTexture("_DynReflTex", leftReflCamera.GetComponent<Camera>().targetTexture );
                }
                else{
                        // Render right camera
                        rightReflCamera.GetComponent<Camera>().Render();
                        reflectiveObj.GetComponent<Renderer>().material.SetTexture("_DynReflTex", rightReflCamera.GetComponent<Camera>().targetTexture );
                }
                eyeIndex = 1 - eyeIndex;
        }
}
```

Check Left/Right Reflection Synchronization



Stereo Reflections in Ice Cave VR



Wrap Up

- Unity has made a great contribution to VR democratization
- In mobile VR the user is no longer limited to the size of the screen and is now fully embedded in a virtual world
- It is possible to run high quality VR and non-VR content on current mobile devices using optimized rendering techniques
- Stereo reflections in VR improve user experience

  https://community.arm.com/groups/arm-mali-graphics/blog/2016/03/10/combined-reflections-stereo-reflections-in-vr
- Check out The ARM Guide for Unity Developers for optimization tips, recommendations and very efficient rendering techniques to make the most out of Unity when developing VR mobile games

VR Integration Into Unity

Carl Callewaert
Americas Director & Global Leader of Evangelism
Unity Technologies

Movement in Mobile VR

Patrick O'Luanaigh
CEO - nDreams

VR Games & Experiences

THE ASSEMBLY

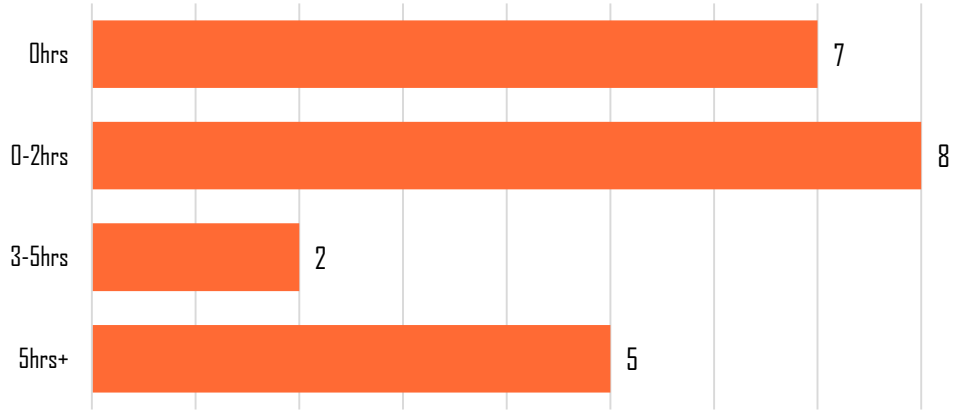

Y
B
A


steelseries
Y
X
B
A


SAMSUNG
PLAY
SELECT
START

Latest Focus Test

- 22 experienced gamers
- 20-30 minute sessions
- Inexperienced in VR



Previous Tests

Internal

Events

Oculus

40-50% | suffer discomfort when walking around in VR*.

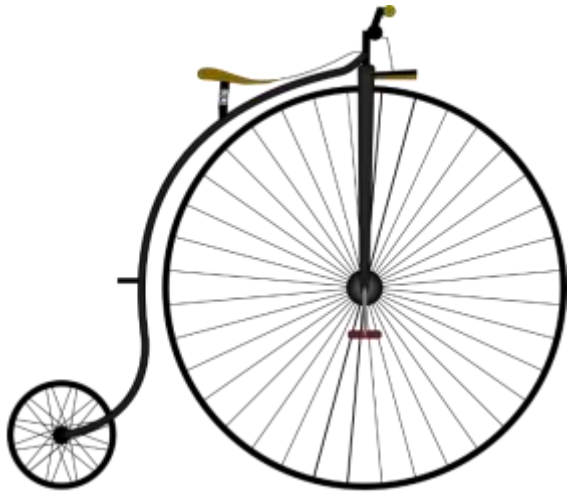
*Varying depending on headset, framerate, room temperature etc.

50-60%

DON’T suffer discomfort when walking around in VR.


ICA

An Easy Solution?



Traditional

Made as accessible as possible

Alternative

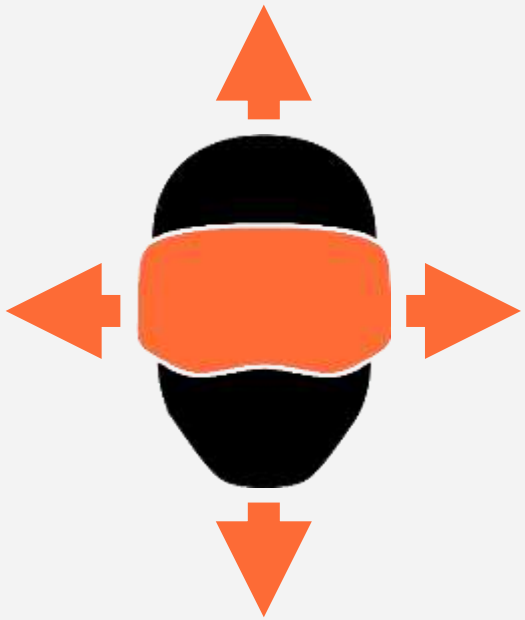
Designed to be comfortable for all


Traditional

The Basics

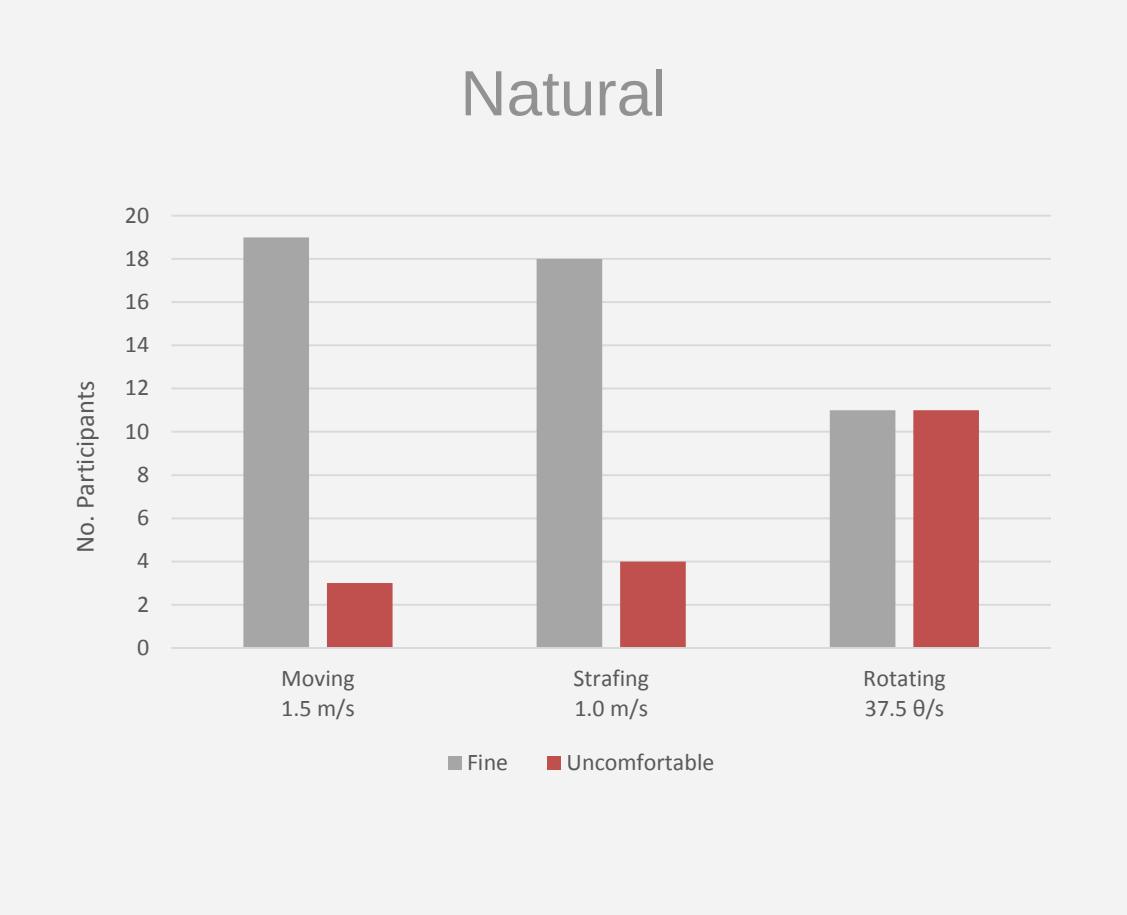
Realistic movement

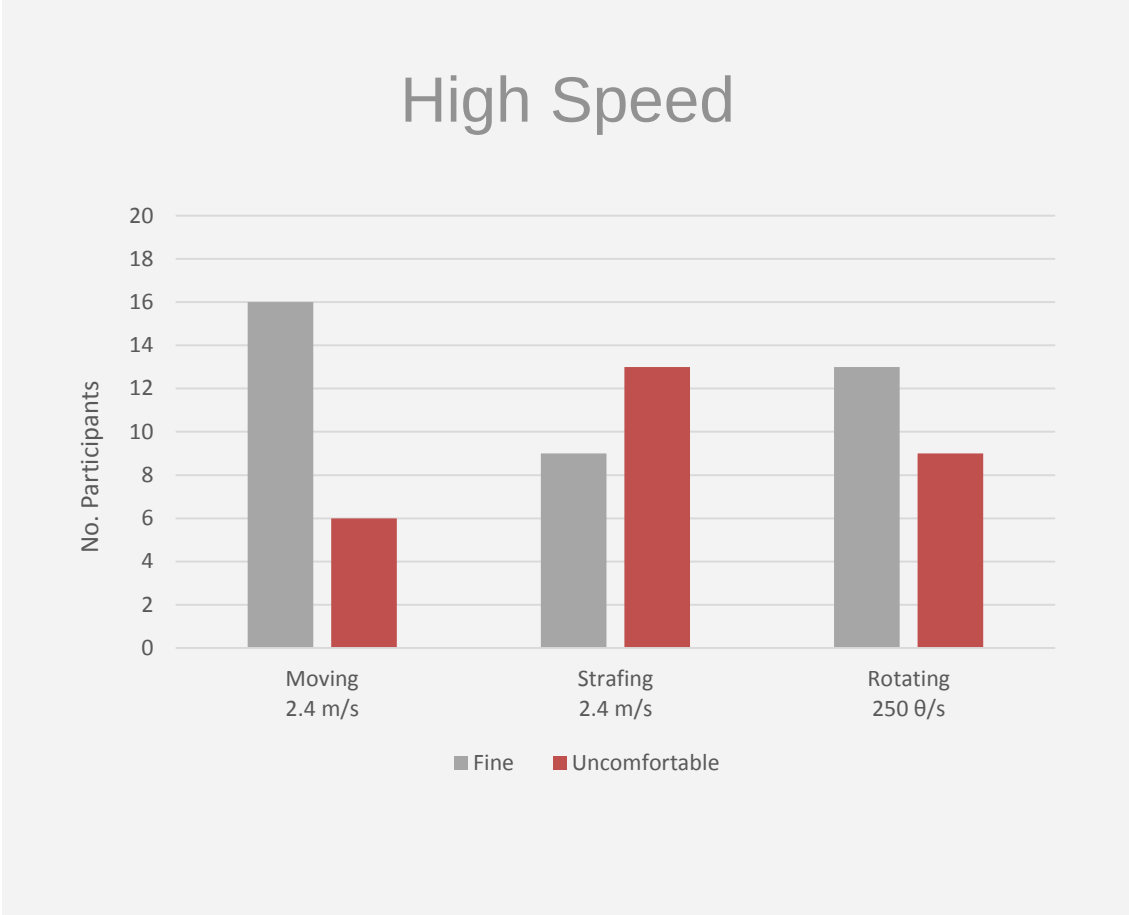
Don’t touch the camera



No perceived acceleration

Findings

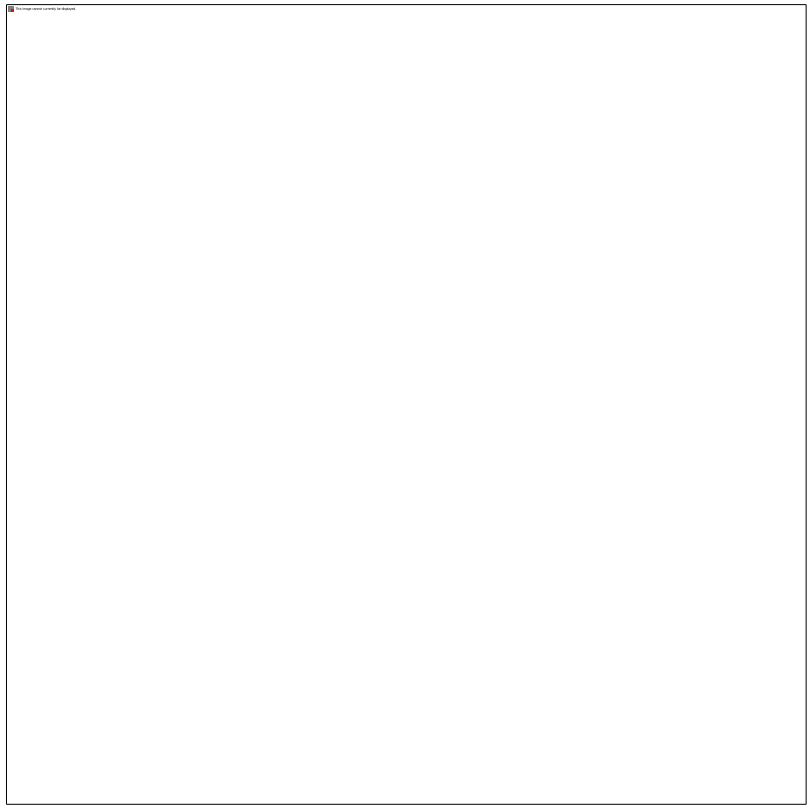

Natural
No. Participants
20
18
16
14
12
10
8
6
4
2
0
Moving
1.5 m/s
Strafing
1.0 m/s
Rotating
37.5 θ/s
Fine
Uncomfortable


High Speed
No. Participants
20
18
16
14
12
10
8
6
4
2
0
Moving
2.4 m/s
Strafing
2.4 m/s
Rotating
250 θ/s
Fine
Uncomfortable

Rotation

18
Of 22

participants preferred a high rotation speed.



Strafing

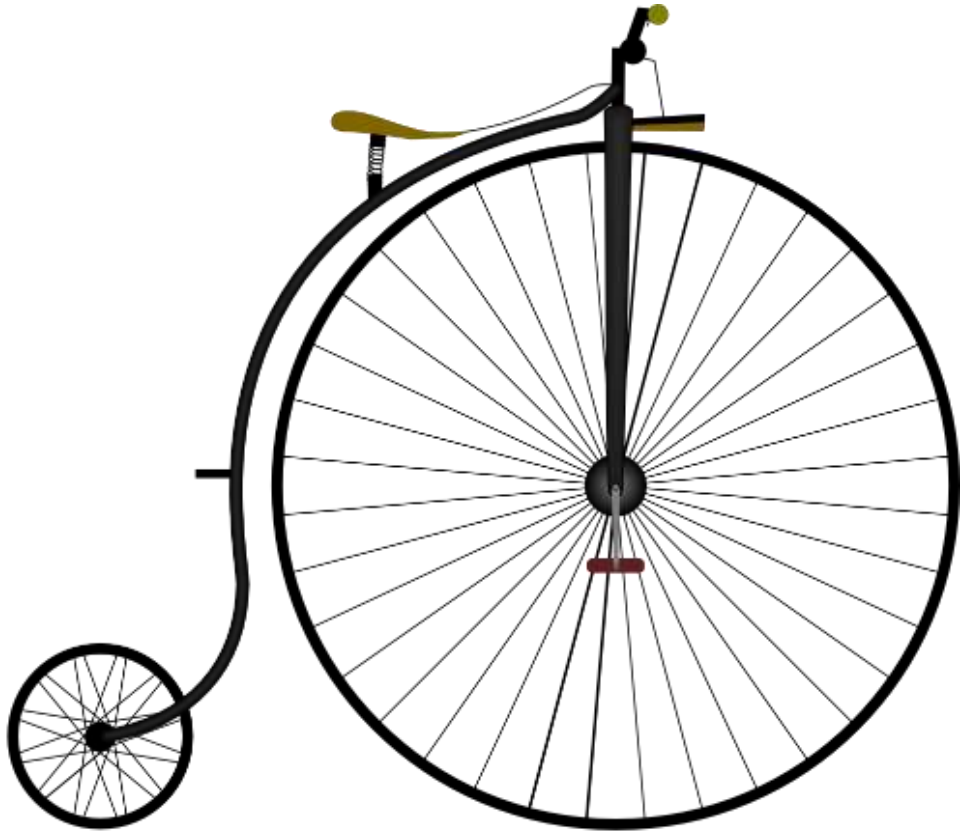
<1m/s

strafe speed

Summary

- Tailor specifically for VR:
  - Natural movement speed.
  - High speed rotation.
  - Consider adding an alternate rotation option.
  - No perceived acceleration.

Not a solution for everyone!




Alternative

~~Movement~~ ~~Rotation~~

100%

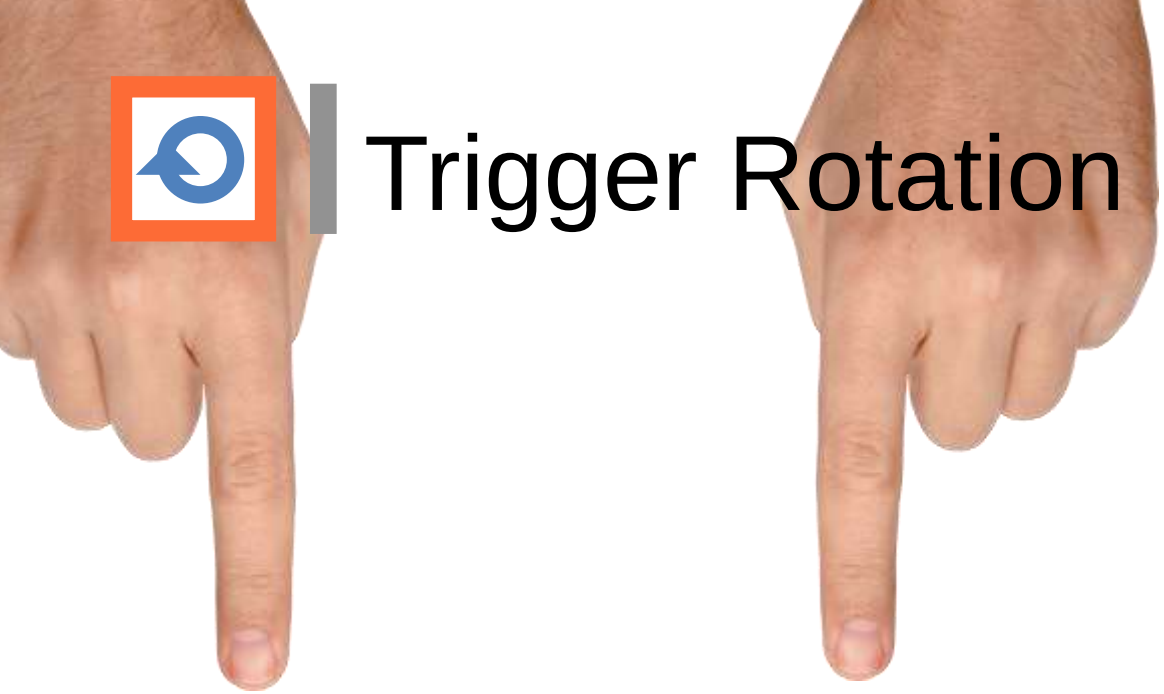
Comfort across all participants*

*As far as we can tell

Alternative Rotation

Alternative controls encourage players to move their body to look around

Trigger Rotation



Snap Rotation

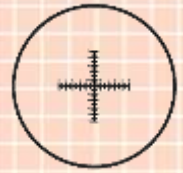
Snap Rotation

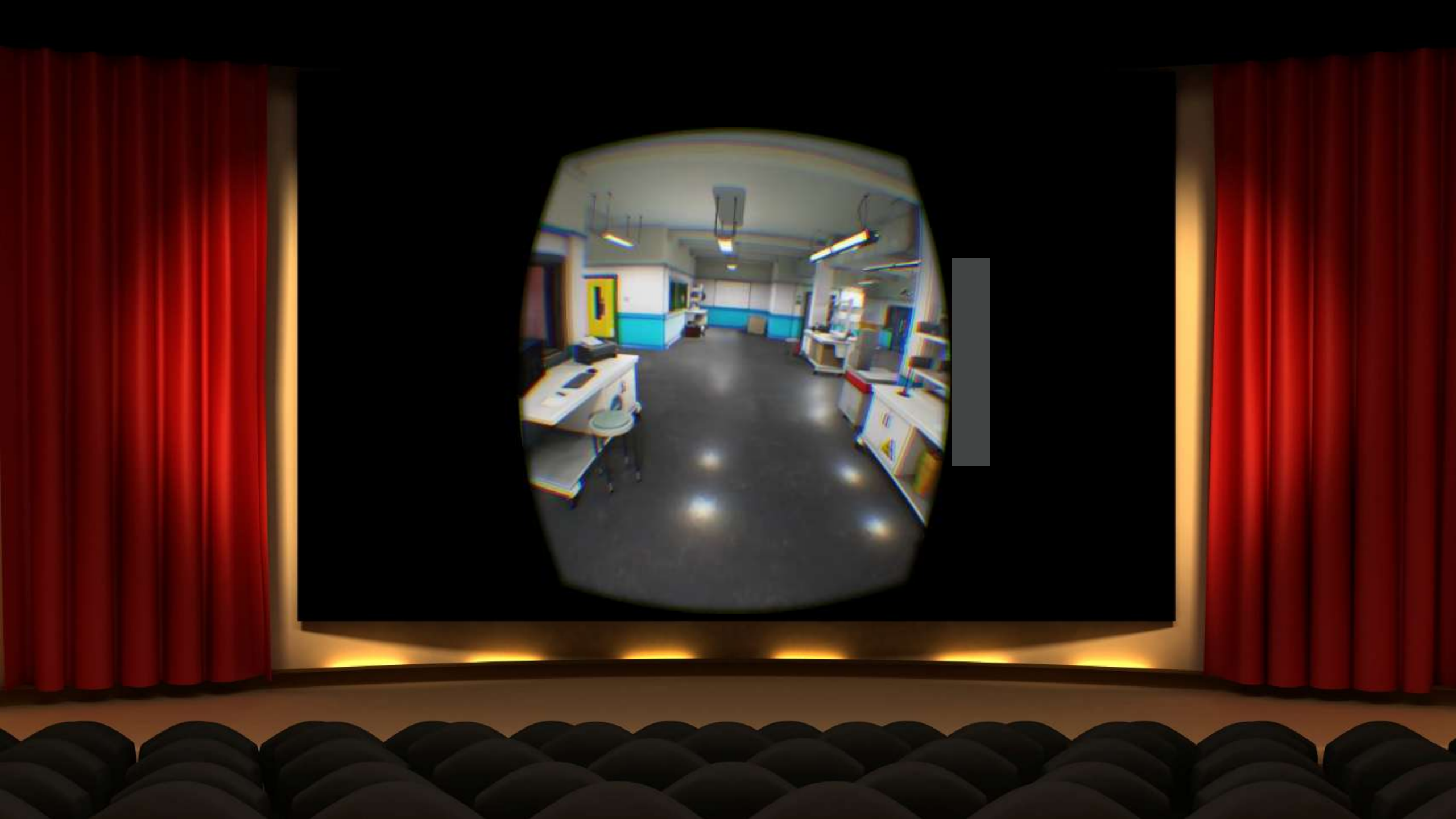
Snap Rotation

Alternative Movement

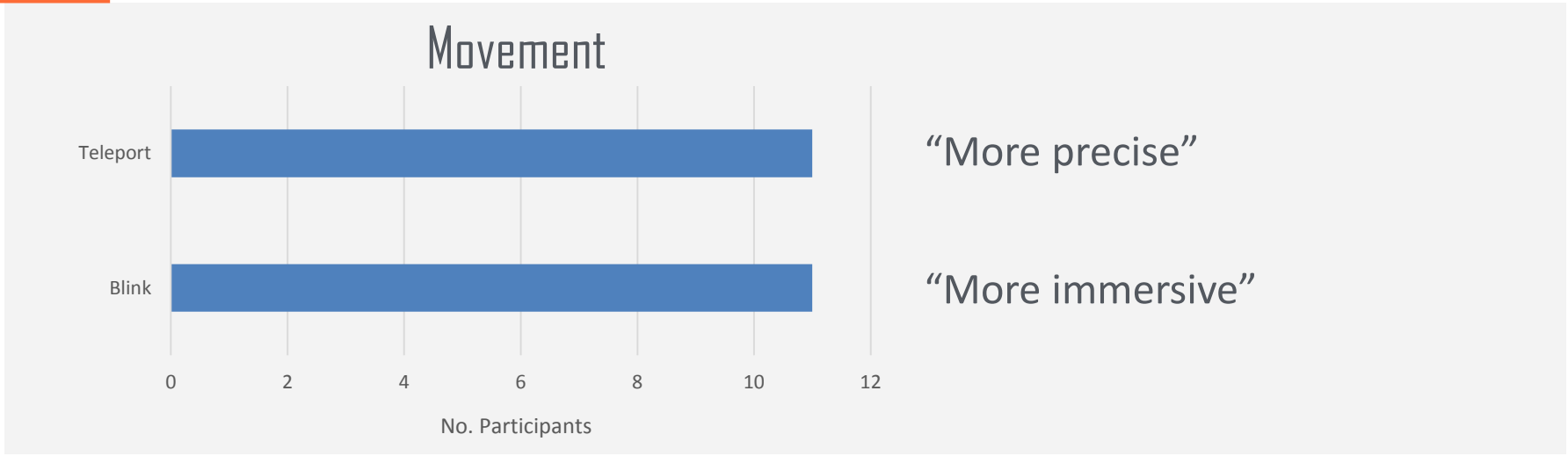
Teleport

Blink



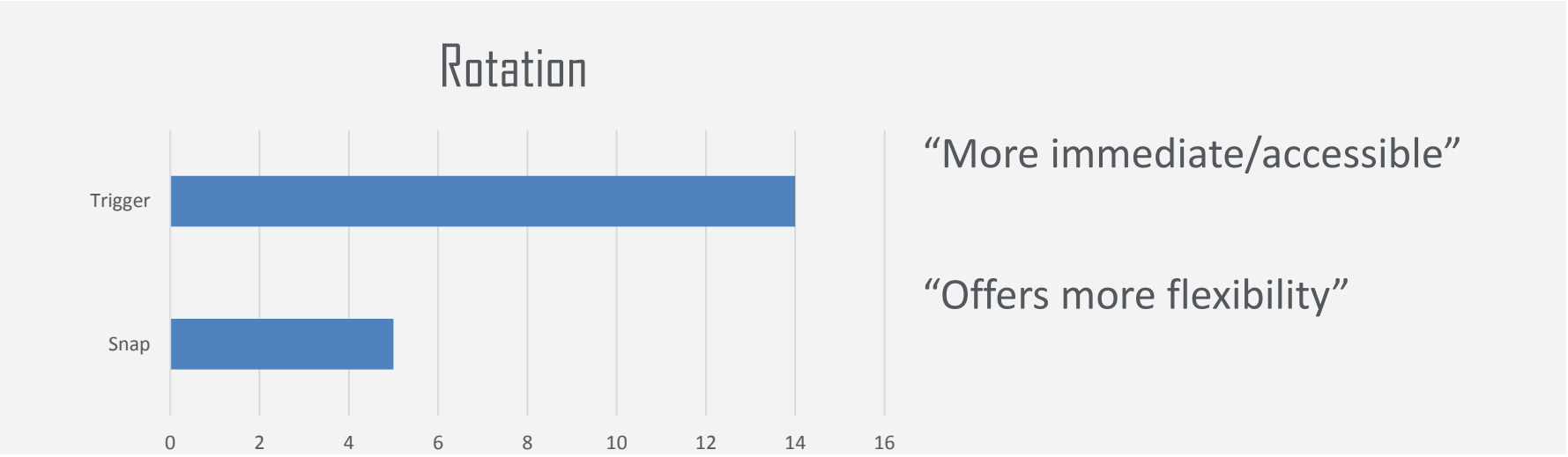


Preferences



"More precise"

"More immersive"



"More immediate/accessible"

"Offers more flexibility"

Further usability testing required

Summary

- Remove movement and rotation altogether.
- Add alternatives!
  - We're going with teleport or blink for moving,
  - Trigger or snap look for rotation.

Google Cardboard

- Much of this is appropriate for even simple headset controls such as the Cardboard.
- Use natural rotation.
- Use gaze plus button to “blink” to another location.

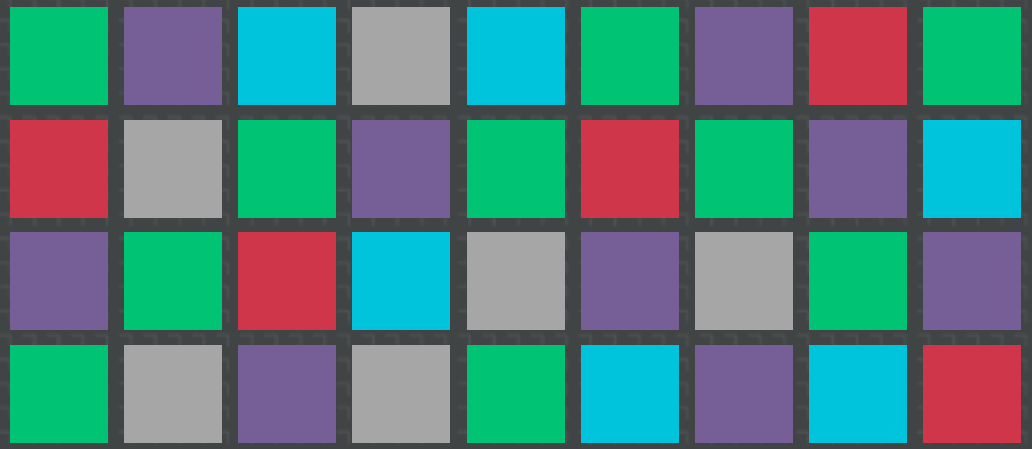
Your Projects

- Experiment with first person movement if appropriate for your title.
- Implement comfortable alternative controls as default that avoid natural movement and rotation.
- You may want to include optional traditional controls tailored to be as comfortable as possible in VR.
- Test everything! What works for your game may be different.

Thank you!

ARM

To Find Out More….

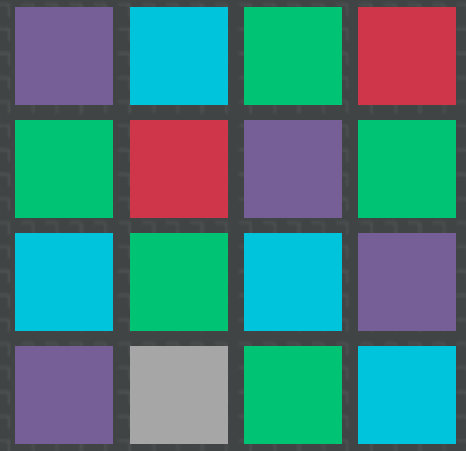


ARM Booth #1624 on Expo Floor:

- Live demos of the techniques shown in this session
- In-depth Q&A with ARM engineers
- More tech talks at the ARM Lecture Theatre

http://malideveloper.arm.com/gdc2016:

- Revisit this talk in PDF and video format post GDC
- Download the tools and resources



More Talks From ARM at GDC 2016

Available post-show at the Mali Developer Center: malideveloper.arm.com/

Vulkan on Mobile with Unreal Engine 4 Case Study

Weds. 9:30am, West Hall 3022

Making Light Work of Dynamic Large Worlds

Weds. 2pm, West Hall 2000

Achieving High Quality Mobile VR Games

Thurs. 10am, West Hall 3022

Optimize Your Mobile Games With Practical Case Studies

Thurs. 11:30am, West Hall 2404

An End-to-End Approach to Physically Based Rendering

Fri. 10am, West Hall 2020