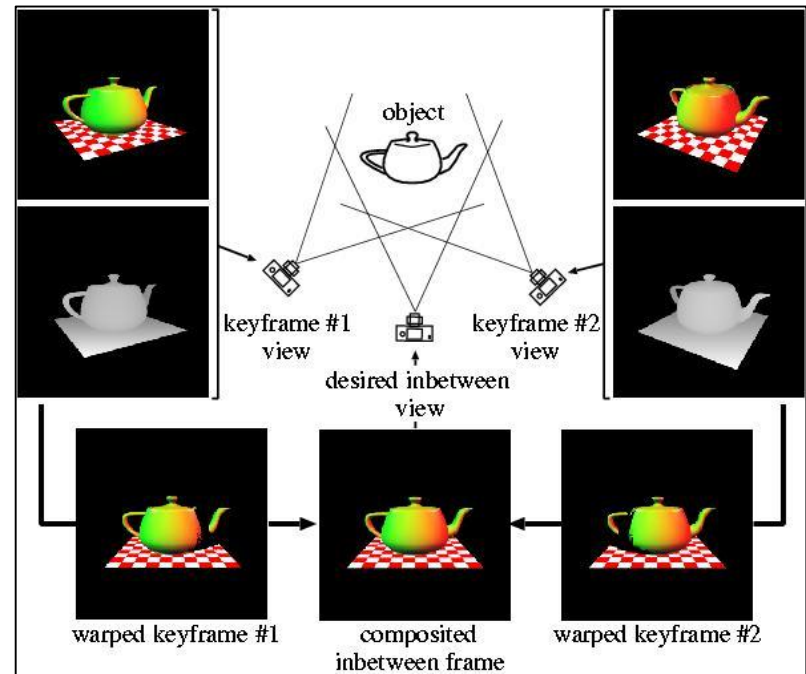


# Construct for Image based MAR Environment

Gerard J. Kim / Korea University

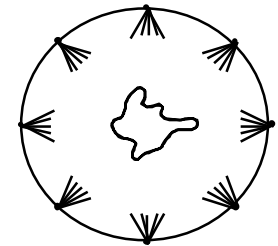
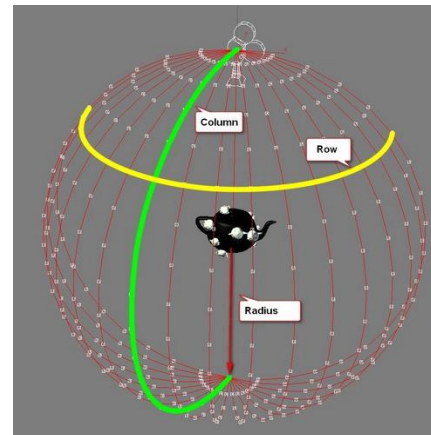
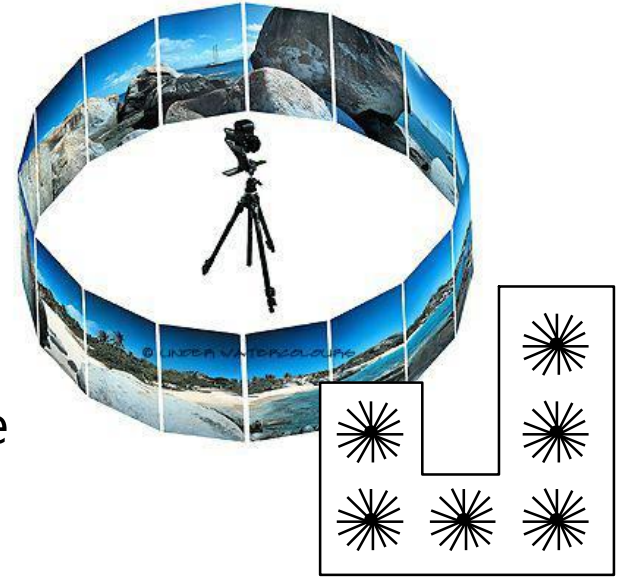
# Image based Rendering

- Rely on a set of two-dimensional images of a scene taken from various view points and then render novel/arbitrary views of this scene by view interpolation
- Many variant approaches, but generally:
  - Computationally demanding
    - Requires pixel level processing (e.g. blending, warping, ...) of multiple images  
→ used to be slow
  - But now it is possible with GPUs and advanced, but relatively low cost hardware
  - “Hole” problems
  - Difficulties with correspondences
  - Closely related to image based modeling
    - 3D geometry, if can be reconstructed, can be of help in correct image based rendering



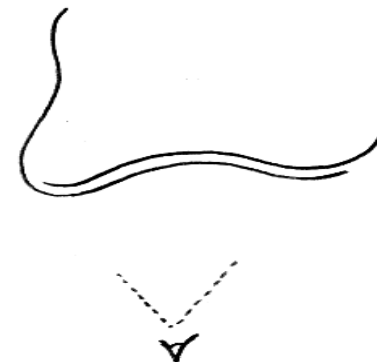
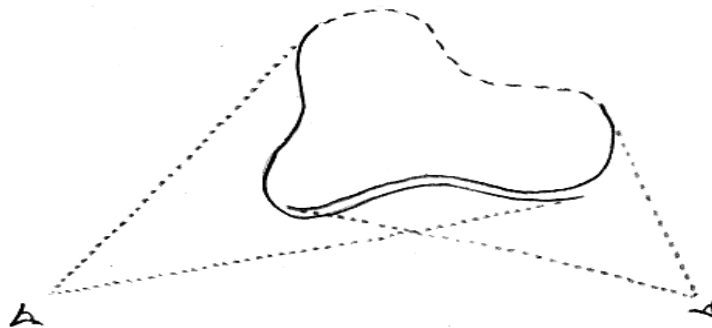
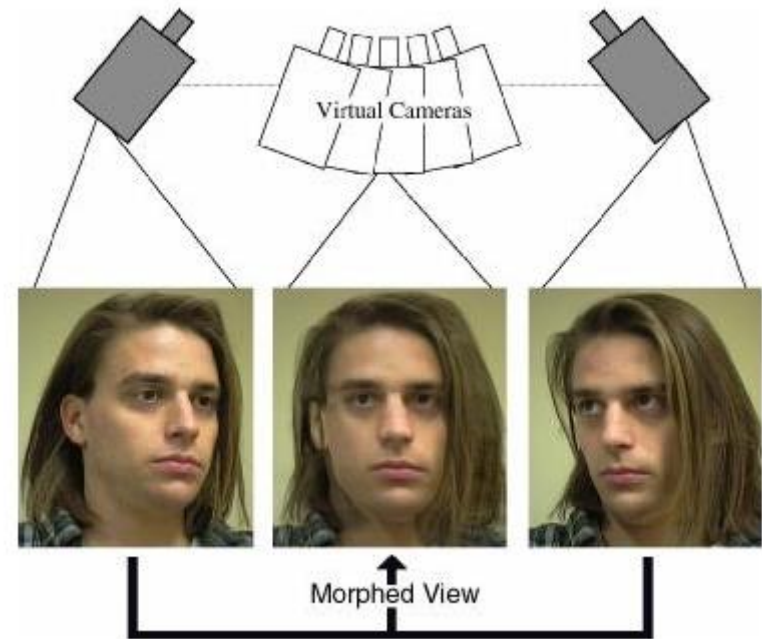
# Apple QuickTime VR

- Outward-looking
  - Panoramic views taken at regularly spaced points
  - Each panorama is stitched from an image
- Inward-looking
  - views taken at points on the surface of a sphere

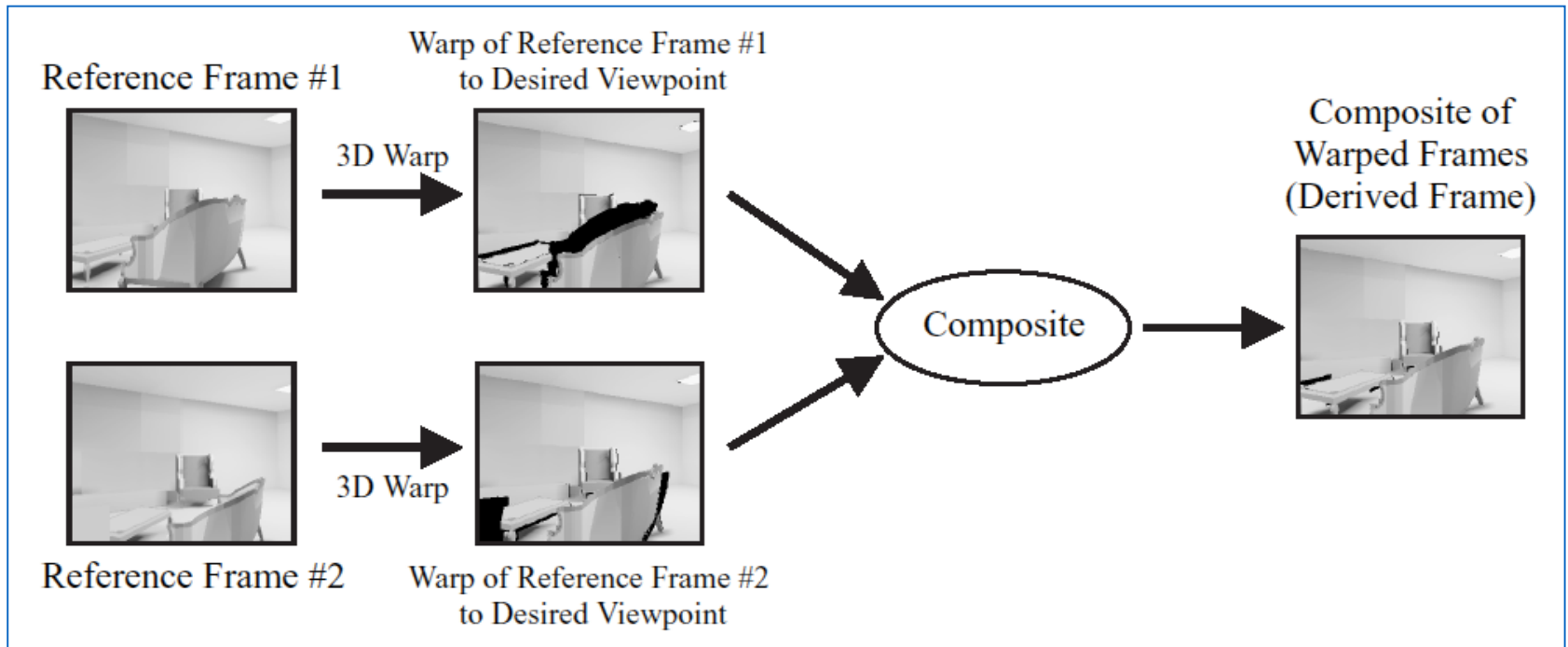


# View interpolation / from multiple views

1. Render object from multiple viewpoints
2. (Convert Z-buffers to range images)
3. Tessellate to create multiple meshes
4. Re-render from new viewpoint
5. Use depths to resolve overlaps
6. Use multiple views to fill in holes

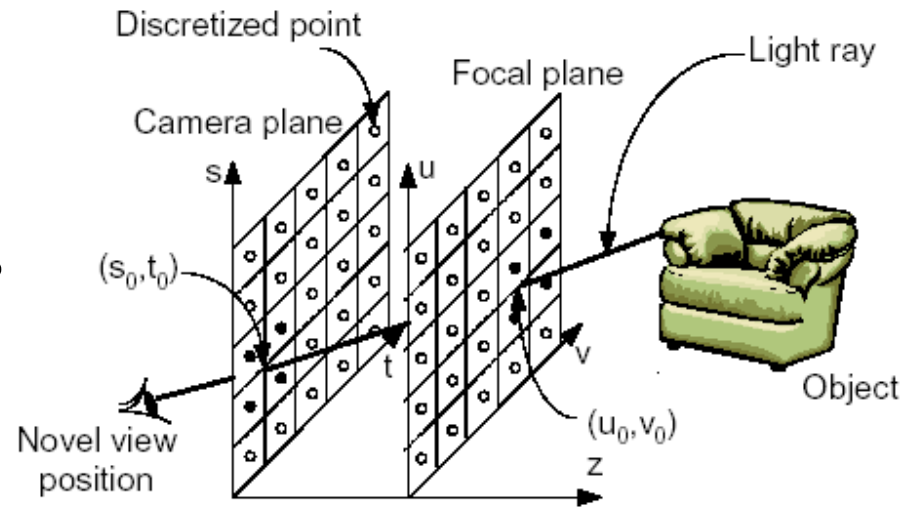
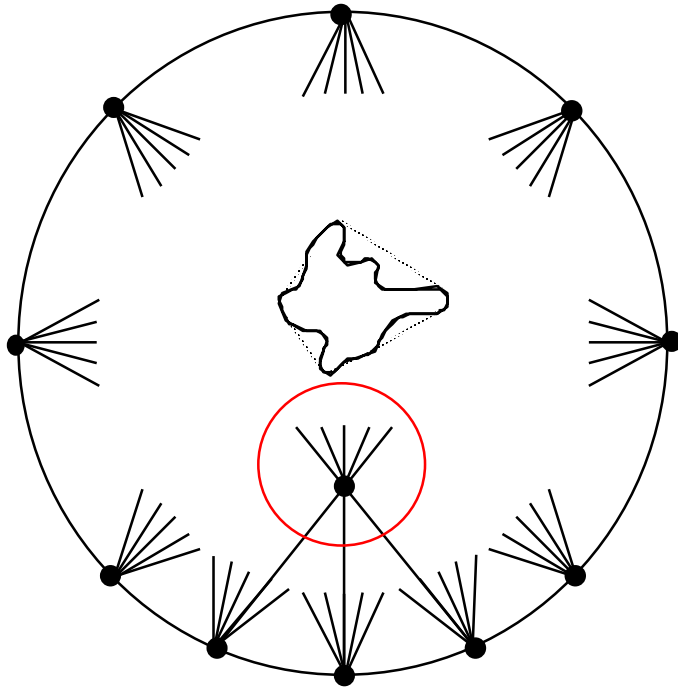


# Post-rendering 3D warping



- Convert reference images to polygon meshes (using depth)
- Warp meshes to interpolated viewpoint
- Composite by Z-buffer comparison and conditional write

# Light field rendering



- must stay outside convex hull of the object
- like rebinning in computed tomography

# The Plenoptic function

*Radiance as a function of position and direction  
in a static scene with fixed illumination*

- For general scenes

⇒ 5D function

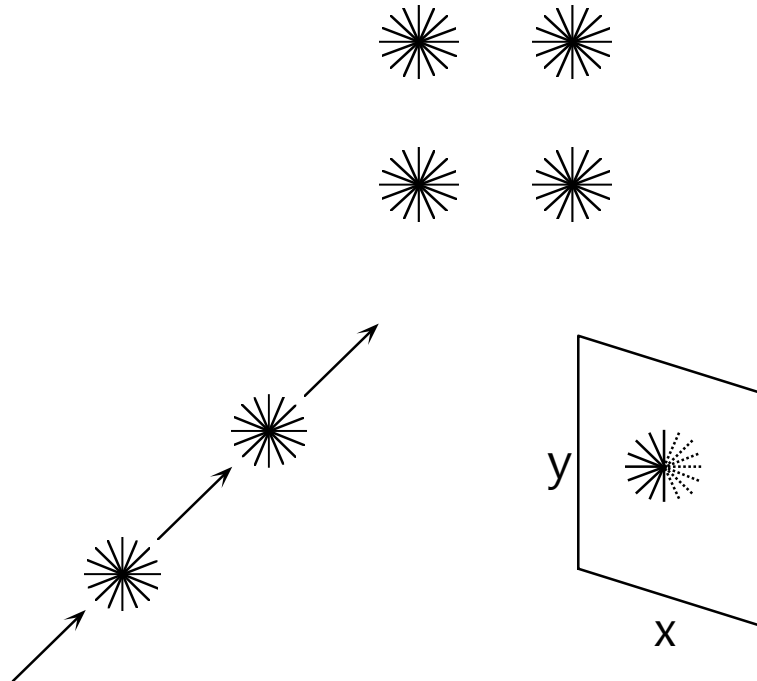
$$L(x, y, z, \theta, \phi)$$



- In free space

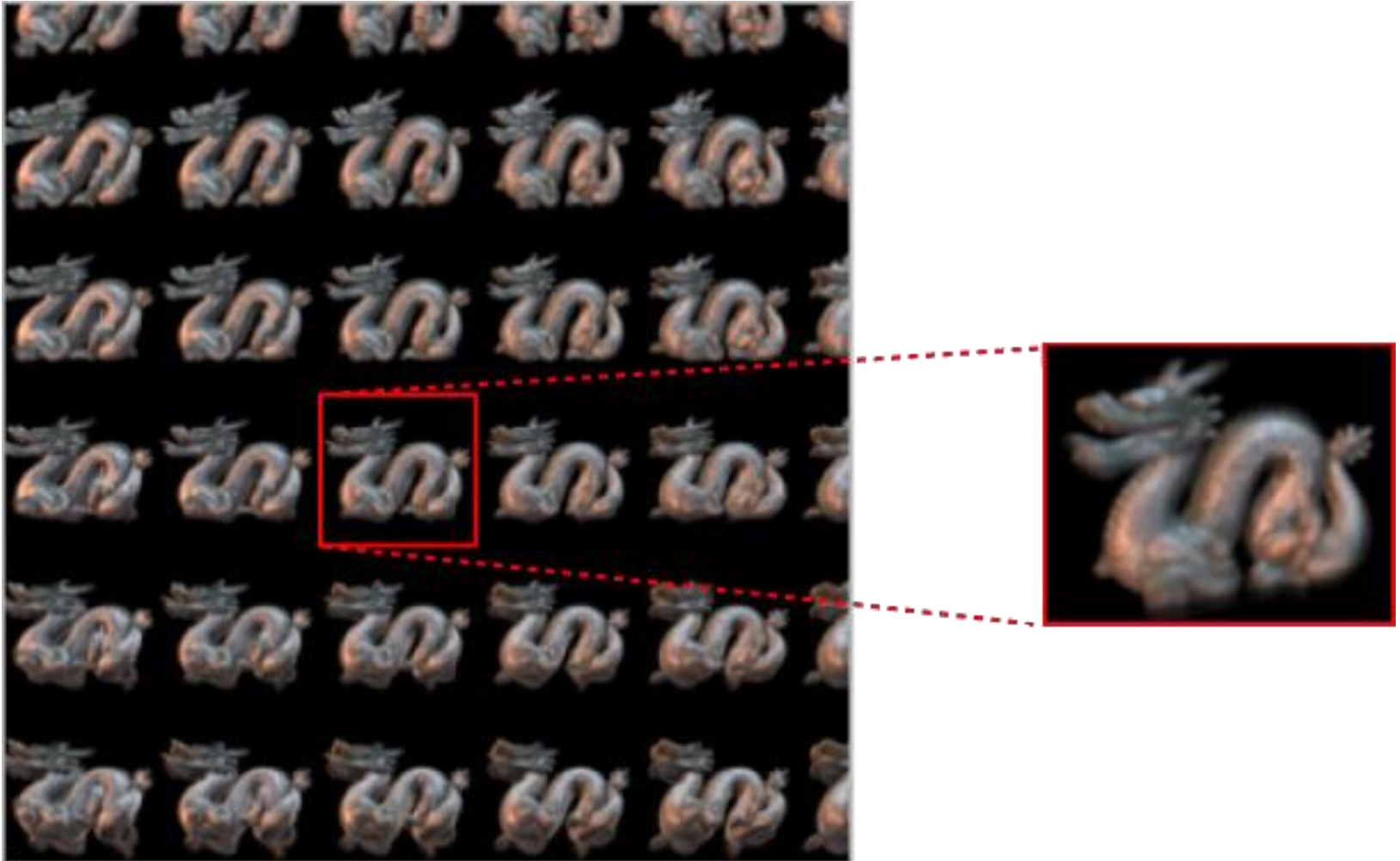
⇒ 4D function

"the (scalar) **light field**"



# A light field is an array of images

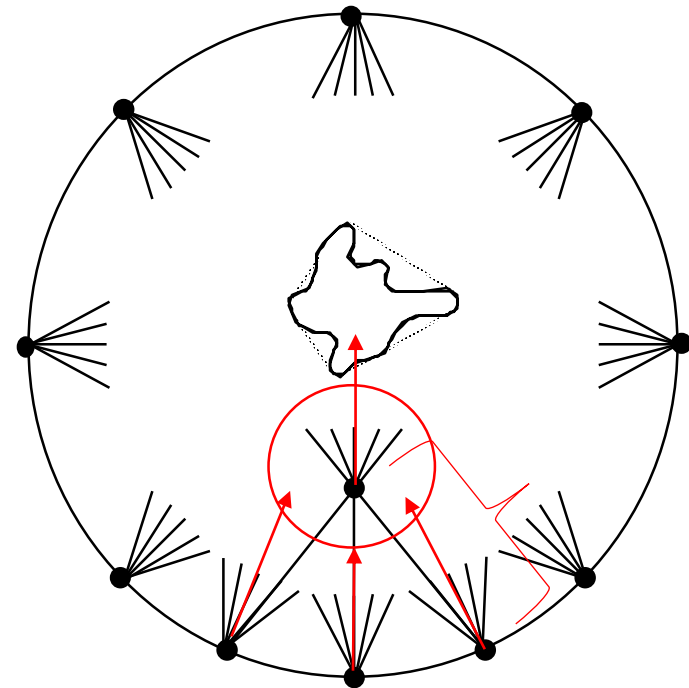
Reconstruct the **plenoptic function** from a set of examples images





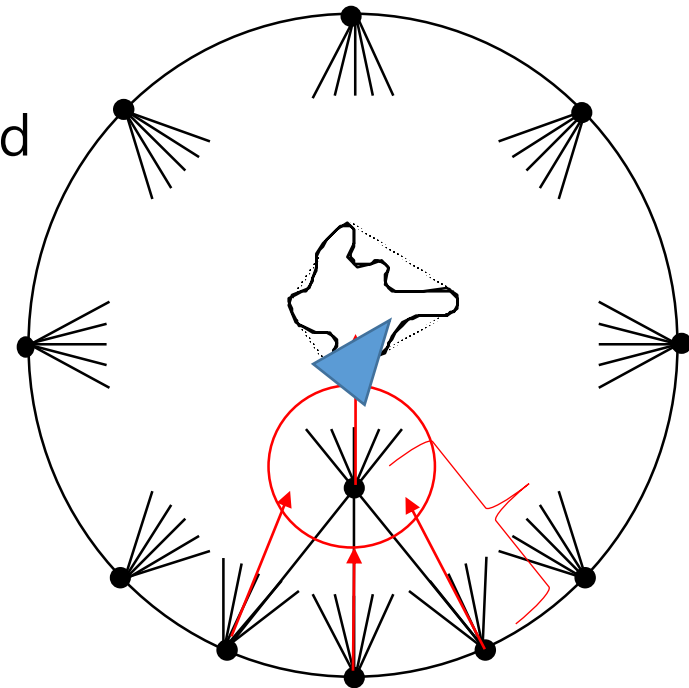
# What do we need to describe a light field?

- Set of images indexed by view position and view direction
- Each image must optionally be specified/supplied with its camera parameters (for flexible rendering purpose)
- Specification of blending policy
  - Which images and how to mix/interpolate given user location and view direction
    - Angle difference
    - Distance
    - Field of view / Eccentricity
    - 3D information?
    - Relative weight



# Hybrid environment: Light field + 3D model

- Set of images indexed by view position and view direction
- Each image must optionally be specified/supplied with its camera parameters (for flexible rendering purpose)
- Specification of blending policy
- 3D model on to which images are mapped
  - Textured indexed face list
    - Image to vertex / face / polygon mapping



# Signature: the first 3 bytes should be "TVR"  
# Version: the 4-th byte is version (currently 0)

# Notations

#  
# file name extension: \*.tvr  
# '#': comments  
# { }: optional elements  
# ...: can be repeated (only for uvc)  
# index: 0-based (first element index = 0)

# Commands

#  
# c: camera  
# i: image  
# v: vertex position and uvc (uv and camera index)  
# n: vertex normal  
# g: group  
# f: face

# Future work

#  
# For the moment, shading is not used.  
# However, OBJ's MTL file can be used together at some point.  
# Material name can be specified in 'g' definition  
# Diffuse colors will use image definitions below, and thereby diffuse colors in MTL will be ignored.

```

# *****
# Data definitions now start here.
# *****

# camera
# c x y z {dx dy dz} {ux uy uz}
# c position, {normalized view direction}, {up vector}

c -0.449365635468153    0.113953661977605 0.390310472806315 -1.3673 -0.1146 0.7147 0 0 1
c -2.31675088419862    -0.490176136938867 0.373652850556974 -3.2411 -0.7173 0.6804 0 0 1
c -3.12999097316749    -0.617123365932605 0.397612906073413 -4.0491 -0.8865 0.6852 0 0 1
c -3.13473441464402    -1.99544743342159 0.386873125068125 -2.9291 -2.9472 0.6146 0 0 1
c -2.40670538610784    -3.96234441628630 0.428324184555085 -2.1814 -4.9085 0.6607 0 0 1
c -2.05578088568405    -4.69899135058515 0.432381118356994 -1.7865 -5.6388 0.6425 0 0 1
c -0.883559926620218   -5.29364862084256 0.438193487117146 0.0545 -5.0263 0.6585 0 0 1
c 0.488647788457424    -4.10436455737432 0.382950536248428 1.4332 -3.8030 0.5134 0 0 1

# image
# i camera_index relative_path_to_this_file

i 0 spot00.jpg
i 2 spot02.jpg
i 5 spot05.jpg
i 7 spot07.jpg

# vertex
# v x y z {u0 v0 c0 u1 v1 c1 u2 v2 c2 ...}
# v position {uvc0 uvc1 ...}
# uvc = u v camera_index

v -2.837600 -6.461940 0.884856 0.0 0.0 0 0.0 0.0 1 ...
v -2.838010 -6.463430 0.787984 0.0 0.1 0 0.0 0.1 1
v -2.857950 -6.365120 0.883447 0.0 0.2 0 0.0 0.2 1
v -2.858360 -6.366610 0.786575 0.1 0.2 0 0.1 0.2 1

```

One vertex can go to many textures

```
# vertex normals
# {n x y z}
# normals are optional, but when they are used,
# they should be defined as many as the number of vertices
# this makes position/uv and normals use the same index in face definition
```

```
n 0 0 1
n 0 0 1
n 0 0 1
n 0 0 1
```

```
# group
# g group_name {material_name}
```

```
g group0
```

```
# faces
# - only triangles can be defined
# - defined as counterclockwise (CCW) winding
# - CCW is a standard for OpenGL
# f v0 v1 v2 (zero-based indexing)
```

```
f 0 1 2
f 1 2 3
```

```
g group1
```

```
f 0 2 3
f 0 1 3
```

```
g group2
```

```
f 1 2 3
f 1 0 3
```

[http://teevr.com/page\\_jtiF02](http://teevr.com/page_jtiF02)

# E.g. X3D nodes? – Indexed face list, Multiple texture

```
<IndexedFaceSet DEF='PumpHouseIFS'
```

```
coordIndex='0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12 13 2 -1 2 13 14  
-1 12 15 16 13 -1 15 0 3 16 -1 16 3 17 -1 9 5 6 10 -1 8 9 10 11 -1 4 8 11 7  
-1 4 5 9 8 -1 11 10 6 7 -1 3 2 14 17 -1 13 16 17 14 -1'
```

```
>
```

16 quads

```
<Coordinate
```

```
point='0.0 0.0 0.0 2.0 0.0 0.0 2.0 1.75 0.0 0.0 1.75 0.0 0.625 0.75 0.0 1.0 0.75  
0.0 1.0 1.6 0.0 0.625 1.6 0.0 0.625 0.75 -0.65 1.0 0.75 -0.65 1.0 1.6 -0.65 0.625  
1.6 -0.65 2.0 0.0 -2.7 2.0 1.75 -2.7 2.0 2.5 -1.0 0.0 0.0 -2.7 0.0 1.75 -2.7 0.0  
2.5 -1.0'
```

```
/>
```

17 vertices

```
</IndexedFaceSet>
```

# E.g. X3D nodes? – Indexed face list, Multiple texture

```
ImageTexture : X3DTexture2DNode, X3DUrlObject {
SFNode      [in,out]      metadata NULL [X3DMetadataObject]
MFString    [in,out]      url [] [URI]
SFBool      []            repeats TRUE
SFBool      []            repeatT TRUE
SFNode      []            textureProperties NULL [TextureProperties] }
```

Viewpoint?

```
<Shape DEF='MovieShapeStandardDefinition'>
  <IndexedFaceSet DEF='Quadrilateral320x240' coordIndex='0 1 2 3' solid='false'>
    <Coordinate point='-1.6 -1.2 0 1.6 -1.2 0 1.6 1.2 0 -1.6 1.2 0' />
    <TextureCoordinate DEF='FullImageMapping' point='0 0 1 0 1 1 0 1' />
  </IndexedFaceSet>
  <Appearance>
    <TextureTransform />
    <MovieTexture DEF='X3dQuipMovieStandardDefinition'
      repeatS='false' repeatT='false'
      url=' "X3dQuipBrutzmanStandardDefinitionMPEG1.mpg" "X3dQuipBrutzmanStandardDefin
    </MovieTexture>
  </Appearance>
</Shape>
```

# MultiTexture

For each MT set?

Per shape/IFS

- The MultiTexture node specifies the application of several individual textures to a 3D object to achieve a more complex visual effect. MultiTexture can be used as a value for the texture field in an Appearance node.
- The texture field contains a **list of texture** nodes (e.g., ImageTexture, PixelTexture, and MovieTexture). The texture field may not contain another MultiTexture node.
- The color and alpha fields define base RGB and alpha values for SELECT mode operations.
- The mode field controls the **type of blending operation**. The available modes include MODULATE for a lit Appearance, REPLACE for an unlit Appearance, and several variations of the two. The mode field may contain an additional blending mode for the alpha channel.
  - EXAMPLE The mode value ""MODULATE","REPLACE"" specifies  $\text{Color} = (\text{Arg1.color} \times \text{Arg2.color}, \text{Arg1.alpha})$ .
- The number of used texture stages is determined by the length of the texture field. If there are fewer mode values, the default mode is "MODULATE".

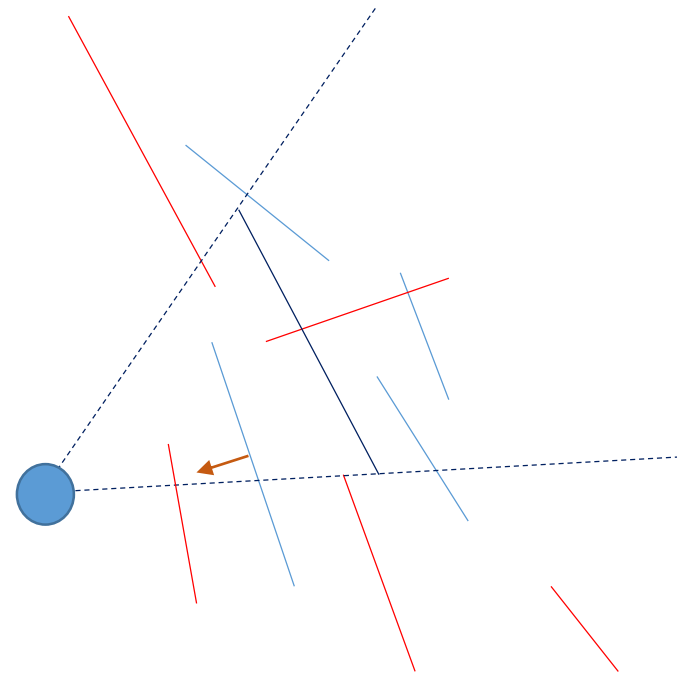


<b>MODE</b>	<b>Description</b>
"MODULATE"	Multiply texture color with current color $\text{Arg1} \times \text{Arg2}$
"REPLACE"	Replace current color $\text{Arg2}$
"MODULATE2X"	Multiply the components of the arguments, and shift the products to the left 1 bit (effectively multiplying them by 2) for brightening.
"MODULATE4X"	Multiply the components of the arguments, and shift the products to the left 2 bits (effectively multiplying them by 4) for brightening.
"ADD"	Add the components of the arguments $\text{Arg1} + \text{Arg2}$
"ADDSIGNED"	Add the components of the arguments with a -0.5 bias, making the effective range of values from -0.5 through 0.5.
"ADDSIGNED2X"	Add the components of the arguments with a -0.5 bias, and shift the products to the left 1 bit.
"SUBTRACT"	Subtract the components of the second argument from those of the first argument. $\text{Arg1} - \text{Arg2}$
"ADDSMOOTH"	Add the first and second arguments, then subtract their product from the sum. $\text{Arg1} + \text{Arg2} - \text{Arg1} \times \text{Arg2} = \text{Arg1} + (1 - \text{Arg1}) \times \text{Arg2}$
"BLENDDIFFUSEALPHA"	Linearly blend this texture stage, using the interpolated alpha from each vertex. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$
"BLENDTEXTUREALPHA"	Linearly blend this texture stage, using the alpha from this stage's texture. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$
"BLENDFACTORALPHA"	Linearly blend this texture stage, using the alpha factor from the MultiTexture node. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$
"BLENDCURRENTALPHA"	Linearly blend this texture stage, using the alpha taken from the previous texture stage. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$
...	...

```
<Shape>
  <!-- TODO add a single geometry node here -->
  <IndexedFaceSet>
    <MultiTextureCoordinate>
      <!-- TODO add multiple TextureCoordinate nodes here, match corresponding MultiTexture children -->
      <TextureCoordinate point='0 0 1 0 1 1 0 1' />
      <TextureCoordinate point='0 0 1 0 1 1 0 1' />
      <TextureCoordinate point='0 0 1 0 1 1 0 1' />
    </MultiTextureCoordinate>
  </IndexedFaceSet>
  <Appearance>
    <Material/>
    <MultiTexture alpha='0.8' color='0.9 1 0.2' function='"" "COMPLEMENT" "ALPHAREPLICATE"'
      mode='""MODULATE" "REPLACE" "BLENDDIFFUSEALPHA"' source='""DIFFUSE" "SPECULAR" "FACTOR"''>
      <!-- TODO add a multiple texture nodes here -->
      <ImageTexture/> <MovieTexture/> <PixelTexture/>
    </MultiTexture>
    <MultiTextureTransform>
      <!-- TODO add multiple TextureTransform nodes here, match corresponding MultiTexture children -->
      <TextureTransform/> <TextureTransform/> <TextureTransform/>
    </MultiTextureTransform>
  </Appearance>
</Shape>
```

# Example 1: Pure IBR

- Just set of indexed/organized textures
  - Augment the current model with
    - Camera information for each texture
      - Use/borrow the viewpoint node model?
    - Provide overall blending policy  
(default is pixel to pixel interpolation)

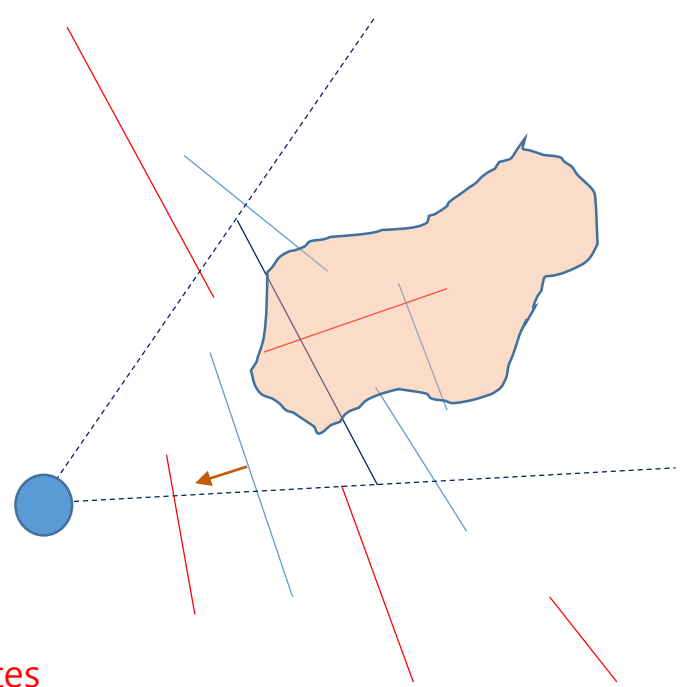


```
<MultiTexture alpha = ... color = ... function = ... mode = ...>  
  <ImageTexture url = "x0.jpg"> <CameraInfo ... /> </ImageTexture>  
  <ImageTexture url = "x1.jpg"> <CameraInfo ... /> </ImageTexture>  
  ...  
</MultiTexture>  
...
```

```
<Viewpoint description='IndexedQuadSet Texture' fieldOfView='3.0307e-01'  
orientation='1.000 0.000 0.000 1.571' position='0.0000e+00 -1.6371e+01 1.0025e-15'/>
```

# Example 2: Hybrid

- Set of indexed/organized textures
  - + Index Face Set
    - Augment the current model with
      - Camera information for each texture
      - Provide overall blending policy (default is pixel to pixel interpolation)
      - Associating each indexed vertex to texture coordinates in a more compact way for a large model



```
<shape>
  <IndexedFaceSet coordIndex='0 1 2 3 -1 4 5 6 7 -1 5 6 7 -1 ...'>
    <Coordinate point = 'x1 y1 z1 x2 y2 z2 .../'>
      <TextureCoordinate point = '0 0 1 0 1 1 0 1 .../'>
      <TextureCoordinate point = '0 0 1 0 1 1 0 1 .../'>
    ...
  </IndexedFaceSet>

  <MultiTexture alpha = ... color = ... function = ... mode = ...>
    <ImageTexture url = "x0.jpg"> <CameraInfo ... /> </ImageTexture>
    <ImageTexture url = "x1.jpg"> <CameraInfo ... /> </ImageTexture>
    ...
  </MultiTexture>
  ...
</shape>
```

<Shape>

```
<IndexedQuadSet index='0 30 31 1 30 60 61 31 60 90 91 61 90 120 121 91 120 150 151 121 150 180
181 151 180 210 211 181 210 240 241 211 240 270 271 241 270 300 301 271 300 330 331 301 330
360 361 331 360 390 391 361 390 420 421 391 420 450 451 421 450 480 481 451 480 510 511 481
510 540 541 ... solid='false' ccw='true' colorPerVertex='true' normalPerVertex='true'>
```

```
<Coordinate point='-1.6730 0.4483 -1.0000 -1.7229 0.4616 -0.9048 -1.7677 0.4736 -0.8069 -1.8073
0.4843 -0.7066 -1.8416 0.4934 -0.6043 -1.8705 0.5012 -0.5002 -1.8939 0.50
0.2879 -1.9240 0.5155 -0.1803 -1.9306 0.5173 -0.0722 -1.9315 0.5176 0.036
1.9164 0.5135 ... />
```



```
<Normal vector='0.8365 -0.2241 0.5000 0.8614 -0.2308 0.4524 0.8838 -0.23
0.3533 0.9208 -0.2467 0.3021 0.9352 -0.2506 0.2501 0.9469 -0.2537 0.1973
0.9620 -0.2578 0.0902 0.9653 -0.2587 0.0361 0.9658 -0.2588 -0.0181 0.9634
0.2568 -0.1260 0.9502 -0.2546 -0.1796 0.9394 -0.2517 -0.2326 0.9259 -0.24
0.3364 0.8907 -0.2387 -0.3868 0.8692 -0.2329 -0.4362 0.8451 -0.2264 -0.4845 0.8185 -0.2195 -0.5509
0.7896 -0.2116 -0.5760 0.7583 -0.2032 -0.6194 0.7248 -0.1942 -0.6610 0.6892 -0.1847 -0.7007 0.6515 -
0.1746 -0.7383 0.6119 ... />
```

```
<TextureCoordinate point='0.0000 0.0000 0.0000 0.0345 0.0000 0.0690 0.0000 0.1034 0.0000 0.1379
0.0000 0.1724 0.0000 0.2069 0.0000 0.2414 0.0000 0.2759 0.0000 0.3103 0.0000 0.3448 0.0000 0.3793
0.0000 0.4138 0.0000 0.4483 0.0000 0.4828 0.0000 0.5172 0.0000 0.5517 0.0000 0.5862 0.0000 0.6207
0.0000 0.6552 0.0000 0.6897 0.0000 0.7241 0.0000 0.7586 0.0000 0.7931 0.0000 0.8276 0.0000 0.8621
0.0000 0.8966 0.0000
```



# But ...

- Light field images are not exactly textures ... (in concept)
  - Can be used as textures, if we had 3D environment models
- Proposal one – WG 6 work item (as part of extending X3D)
  - Extend Texture node (and others) to include camera parameters (simple?)
  - Rest is taken care of by the browser
- Proposal two – WG 9 work item (c.f. Real world reconstruction)
  - Create a structure for a separate light field description
  - Include more elaborate indexing mechanism
  - Addendum to MAR content information model