PRMAN PORTALS

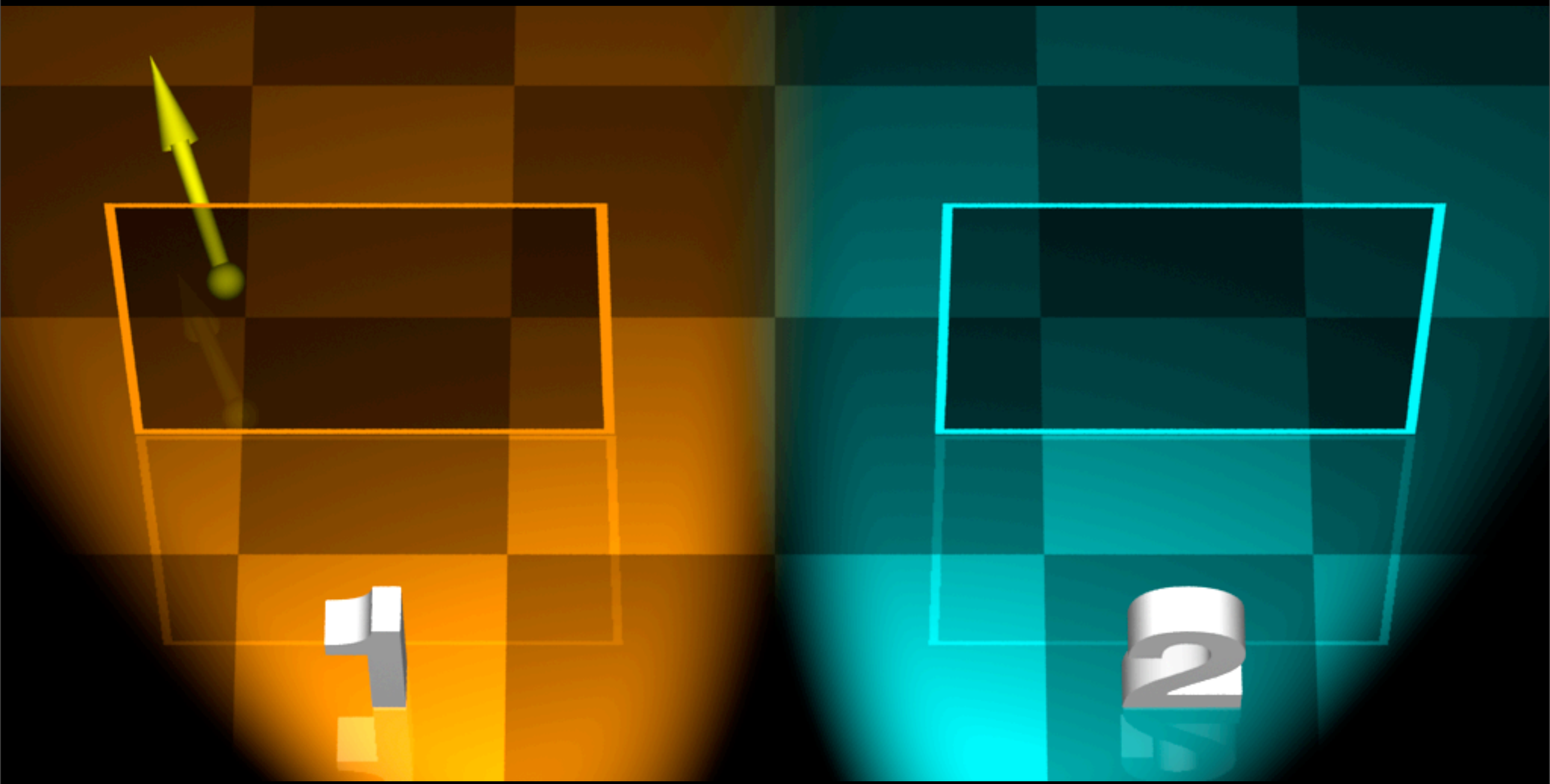
a stupid trick by rob pieké

MPC
Soho, London

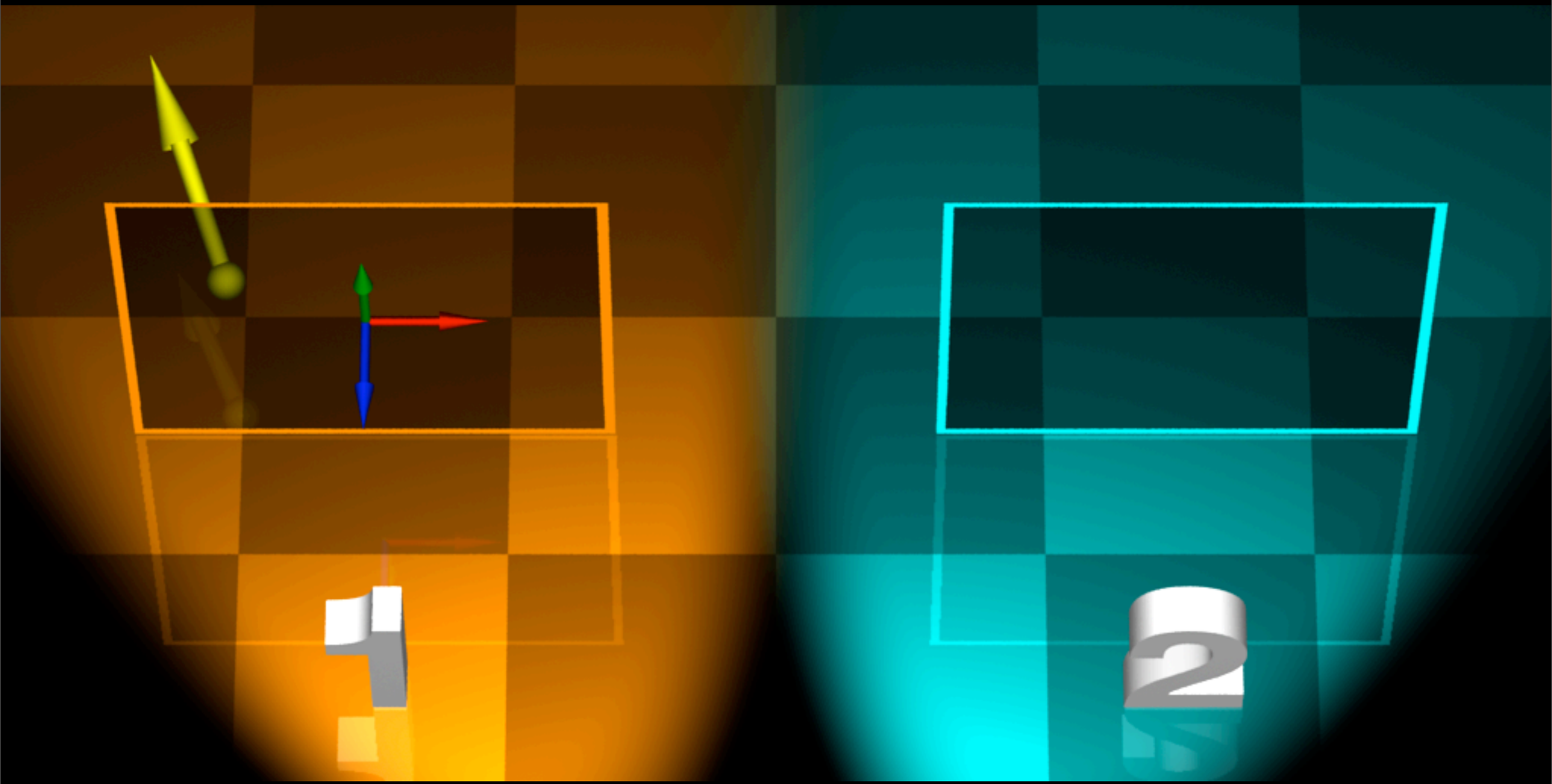
My stupid trick

- inspired by Pixar's *Presto* and Valve's *Portal*
- goal is a single-pass render allowing teleportation of sight, light and geometry
- what goes in the front of one portal comes out the front of another portal
- a portal is essentially a non-entity when considered from the back

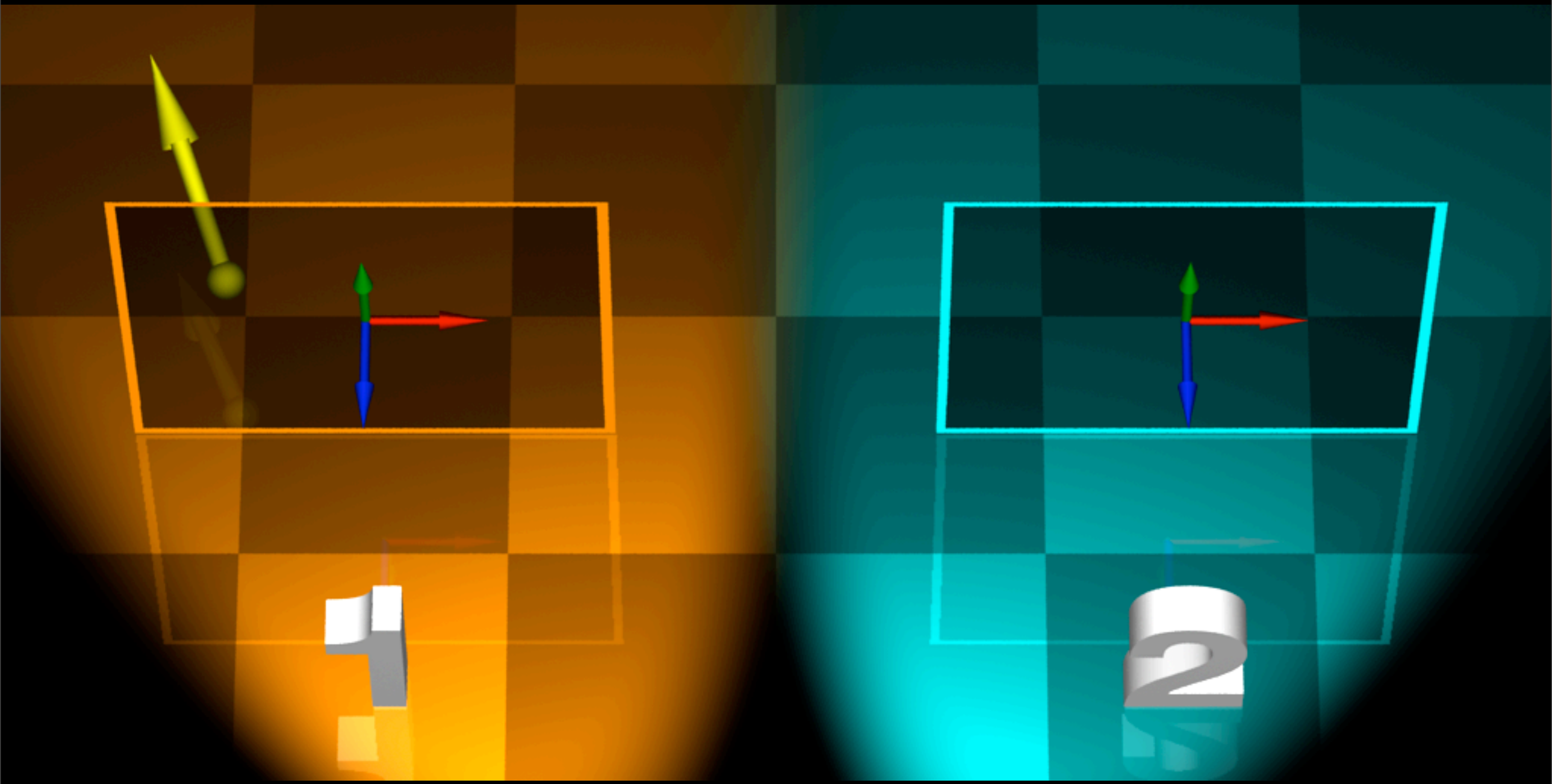
Teleportation



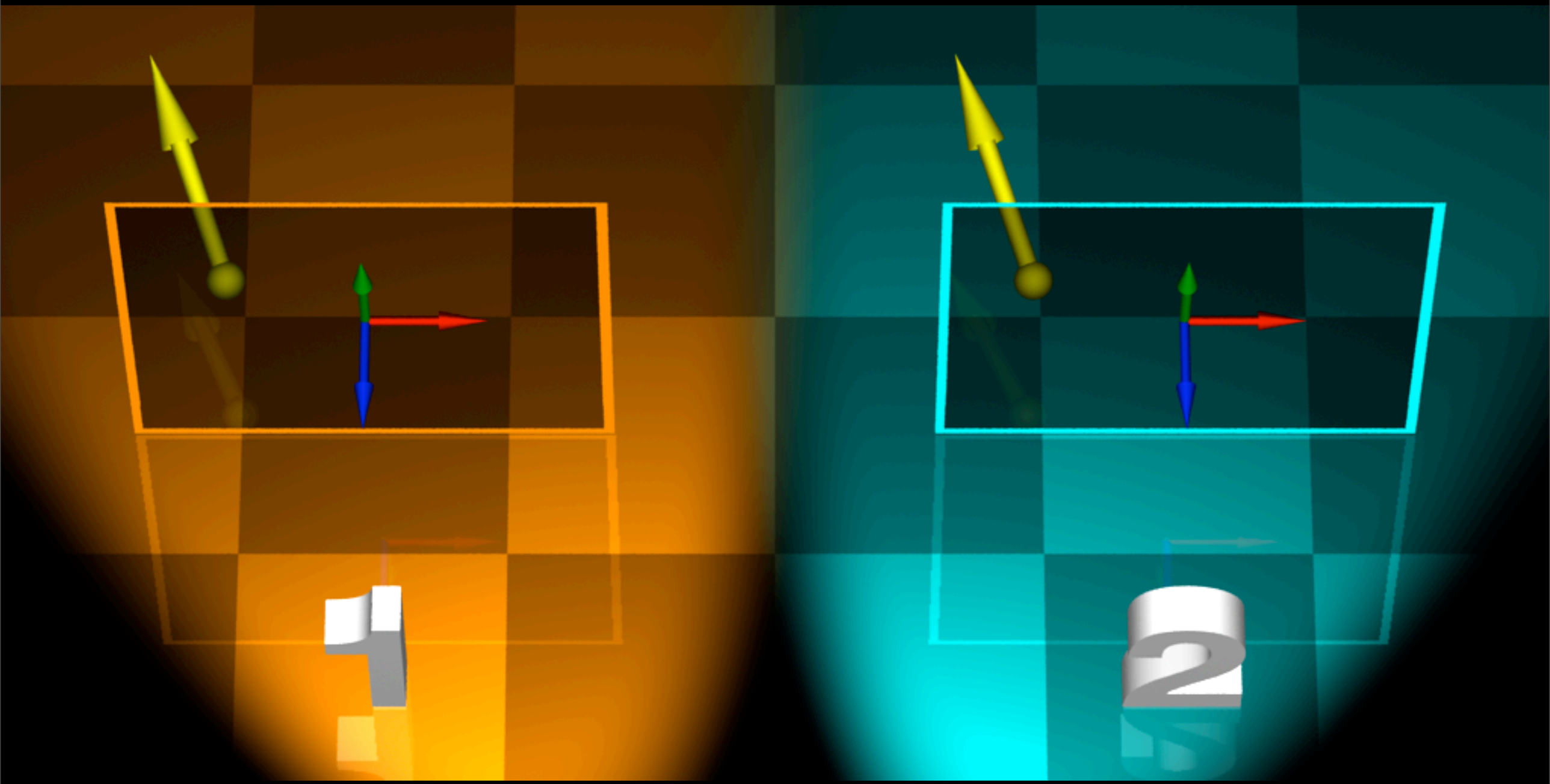
Teleportation



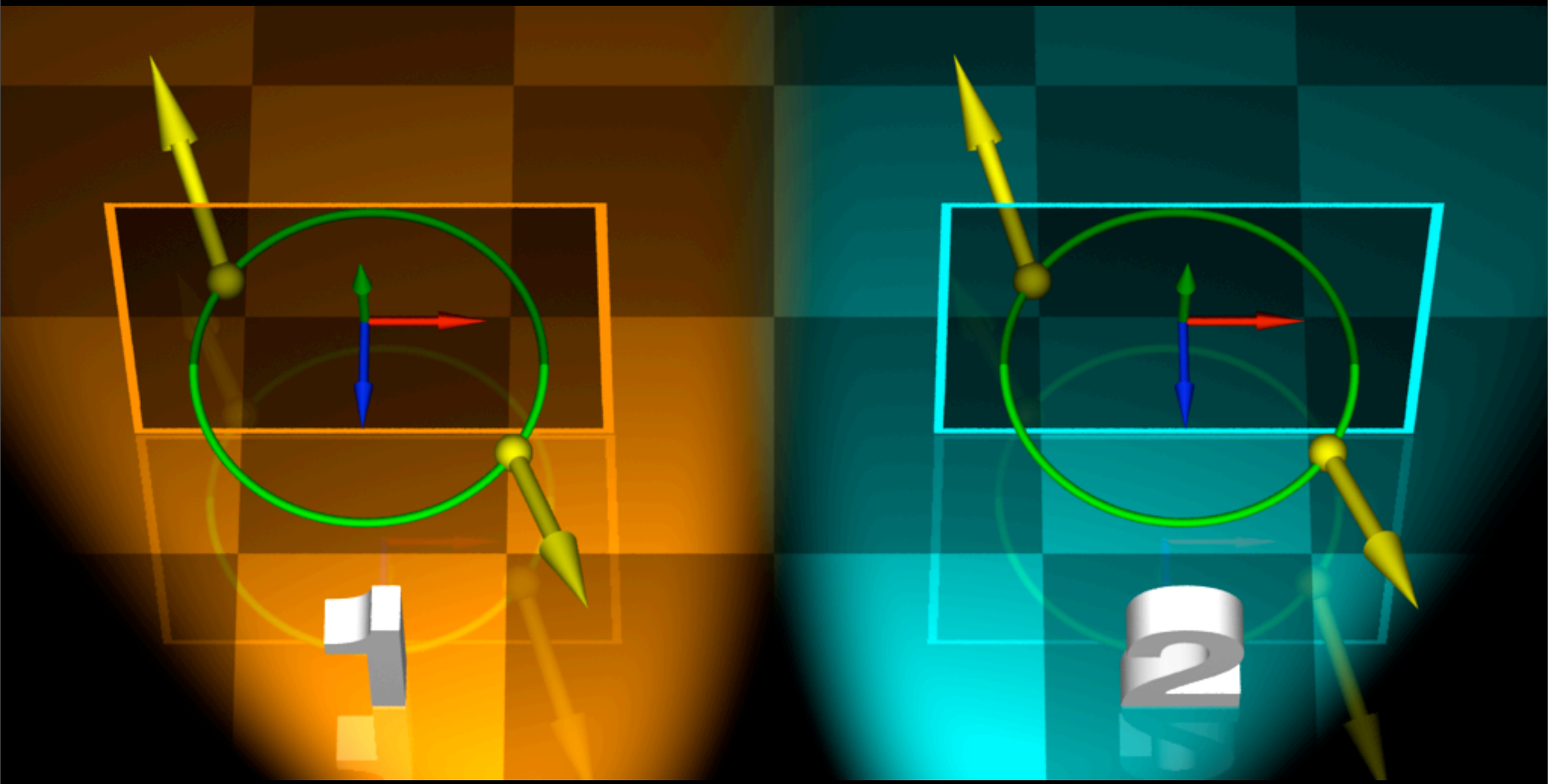
Teleportation



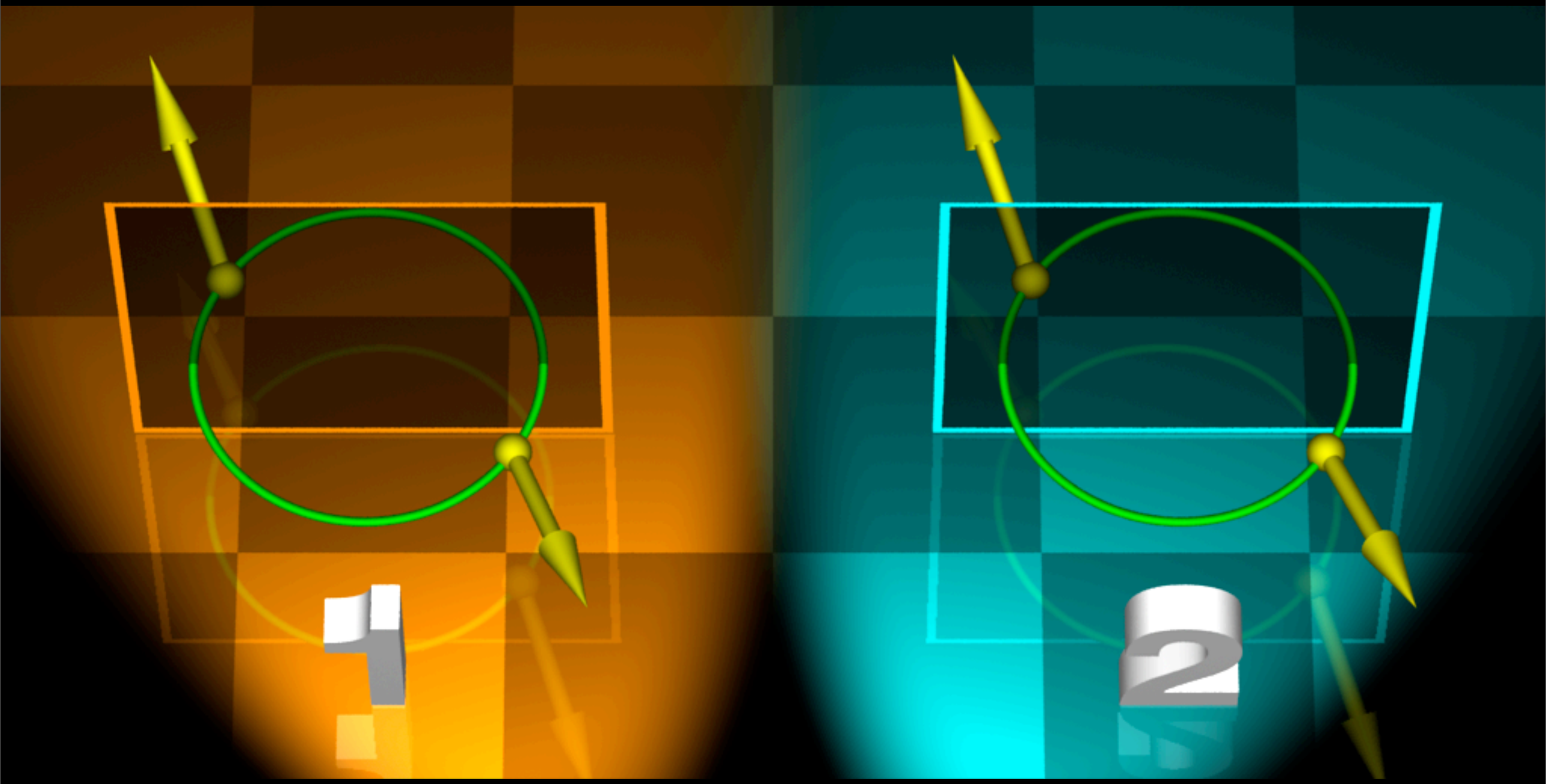
Teleportation



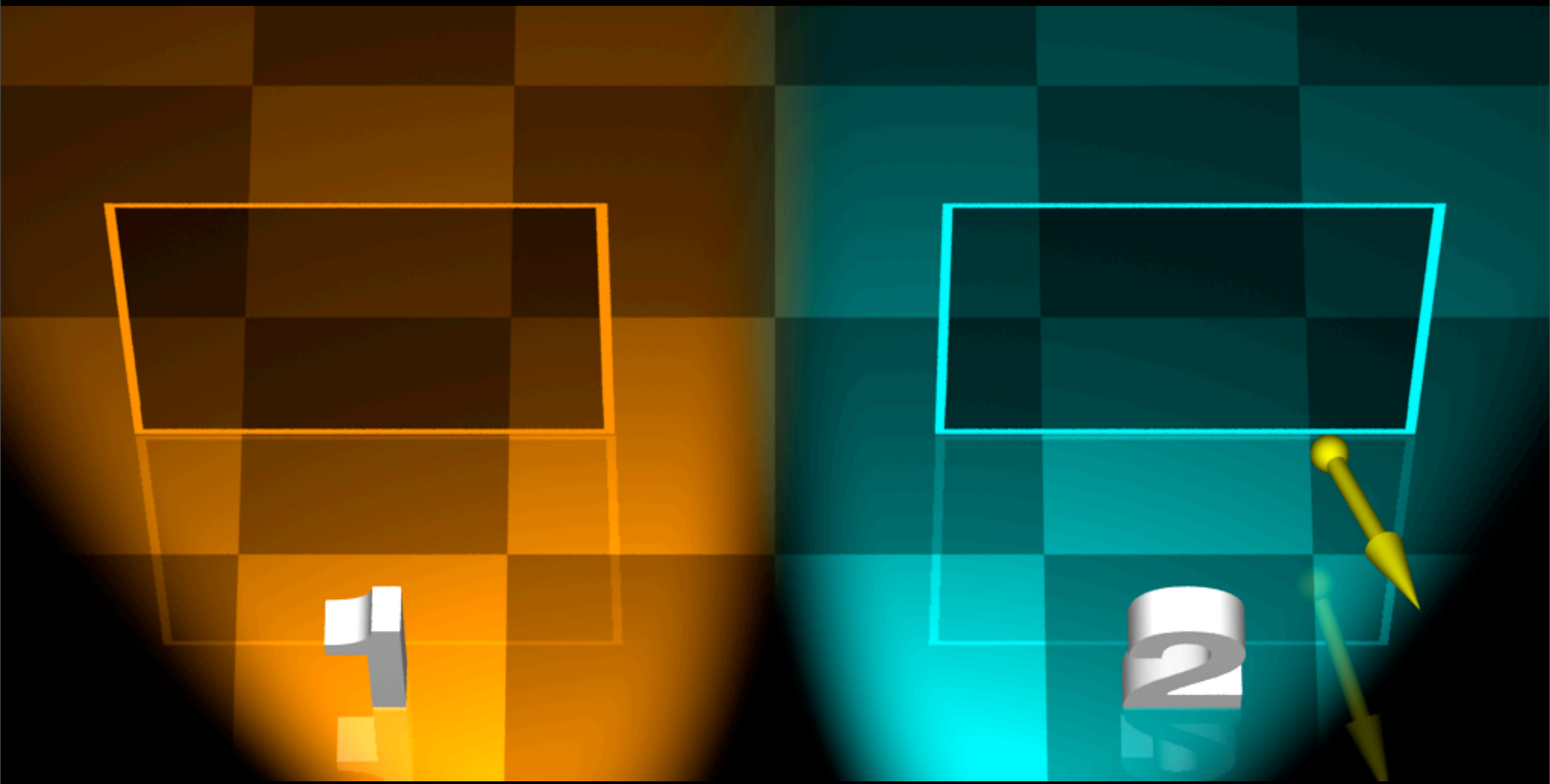
Teleportation



Teleportation



Teleportation



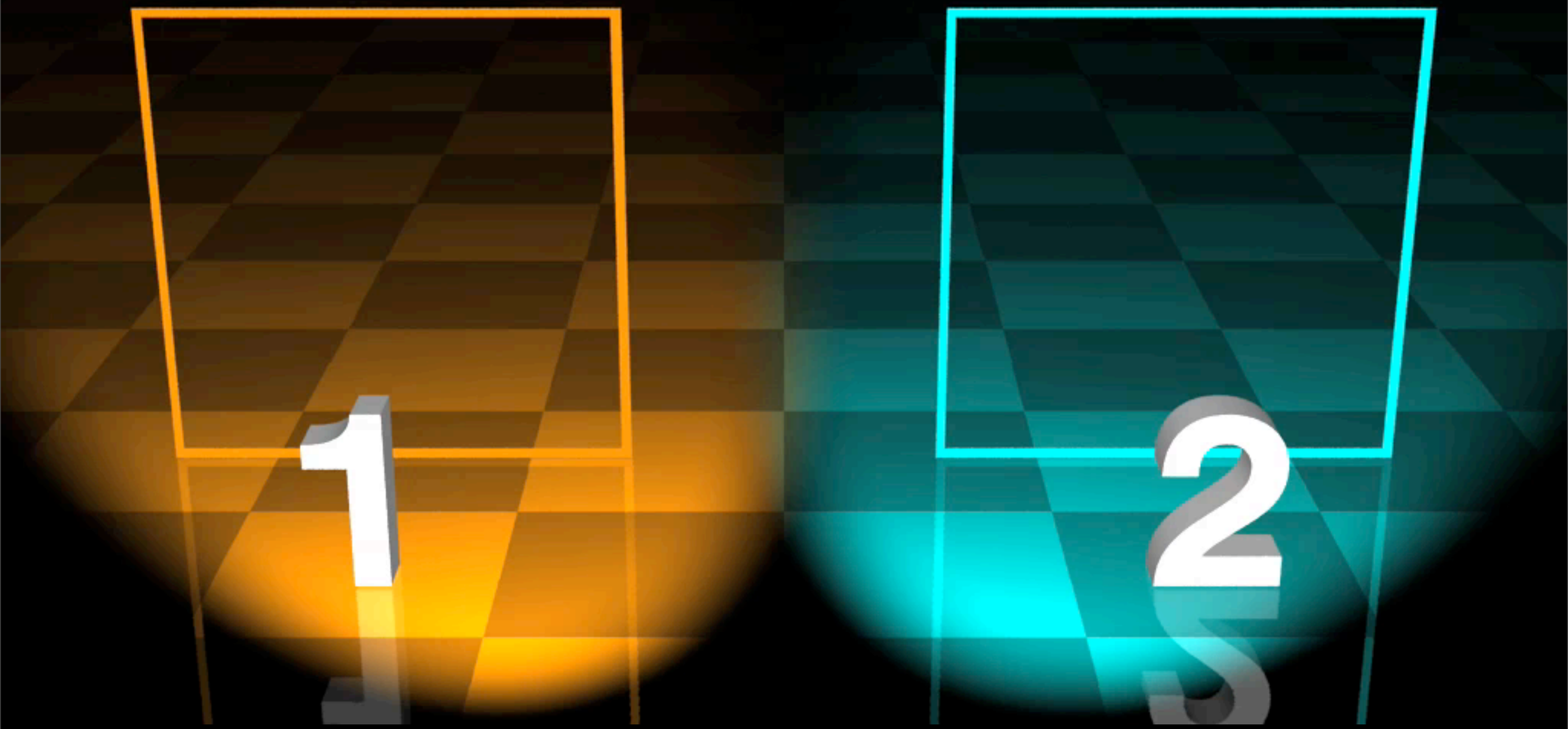
Space transformations

```
void transformThroughPortal(
   point pCur;
   string portalSpace1; string portalSpace2;
   output point pPortal1; output point pPortal2; output point pCurNew
)
{
   pPortal1 = transform( "current", portalSpace1, pCur );
   pPortal2 = rotate( pPortal1, PI, point( 0, 0, 0 ), point( 0, 1, 0 ) );
   pCurNew = transform( portalSpace2, "current", pPortal2 );
}

void vtransformThroughPortal( ... ) { ... }

void ntransformThroughPortal( ... ) { ... }
```

Basic setup

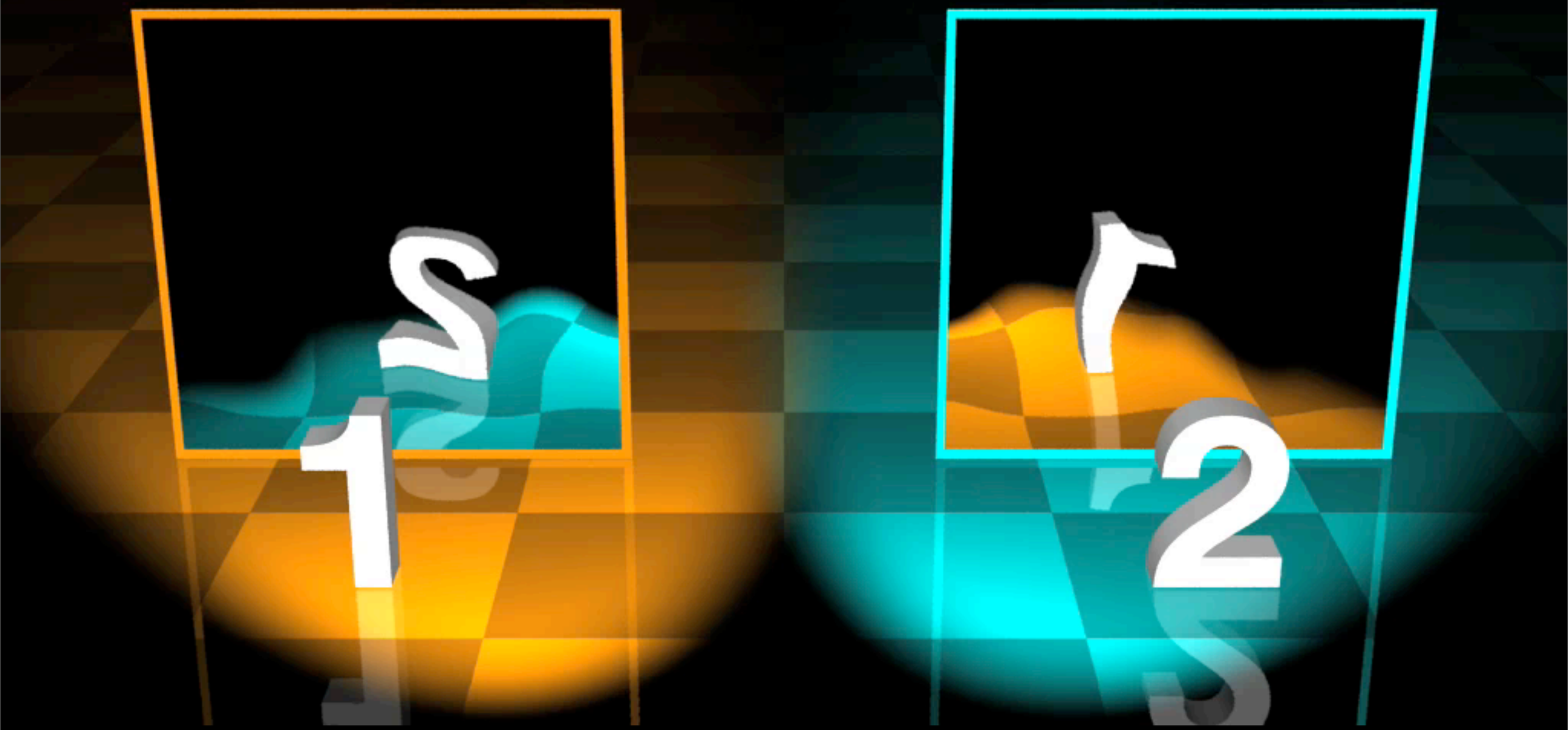


Teleporting sight

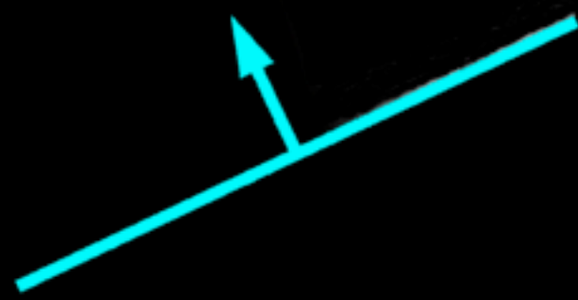
```
surface portal( string exitPortalSpace = "" )
{
   Ci = Oi = 0;
   if( N . I < 0 )
   {
      point pPortal1, pPortal2, pCurNew;
      vector iPortal1, iPortal2, iCurNew;
      transformThroughPortal( P, "object", exitPortalSpace,
            pPortal1, pPortal2, pCurNew );
      vtransformThroughPortal( I, "object", exitPortalSpace,
            iPortal1, iPortal2, iCurNew );
      Ci = trace( pCurNew, iCurNew );
      Oi = 1;
   }
}
```

Teleporting sight

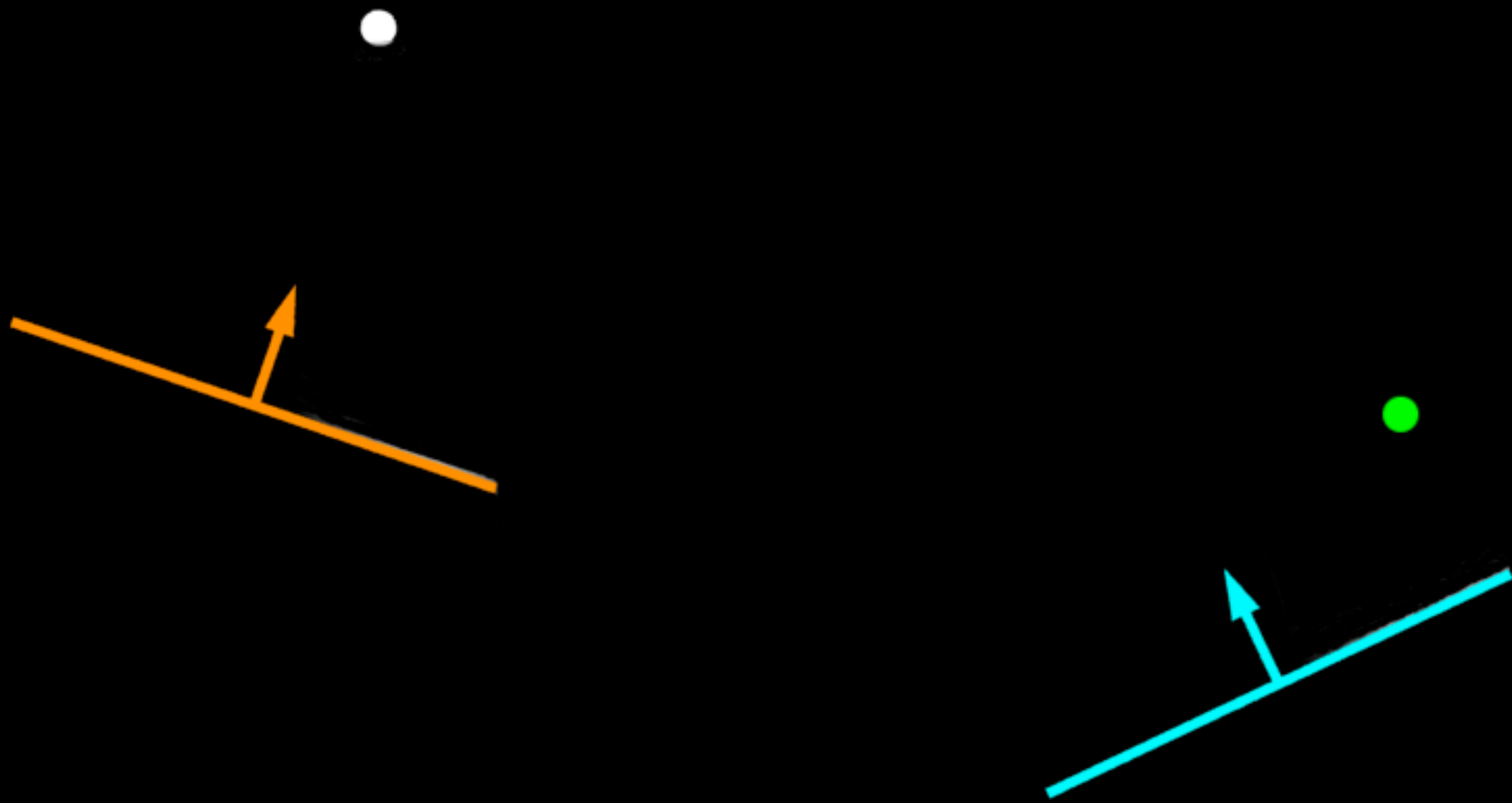
Fun with ray tracing



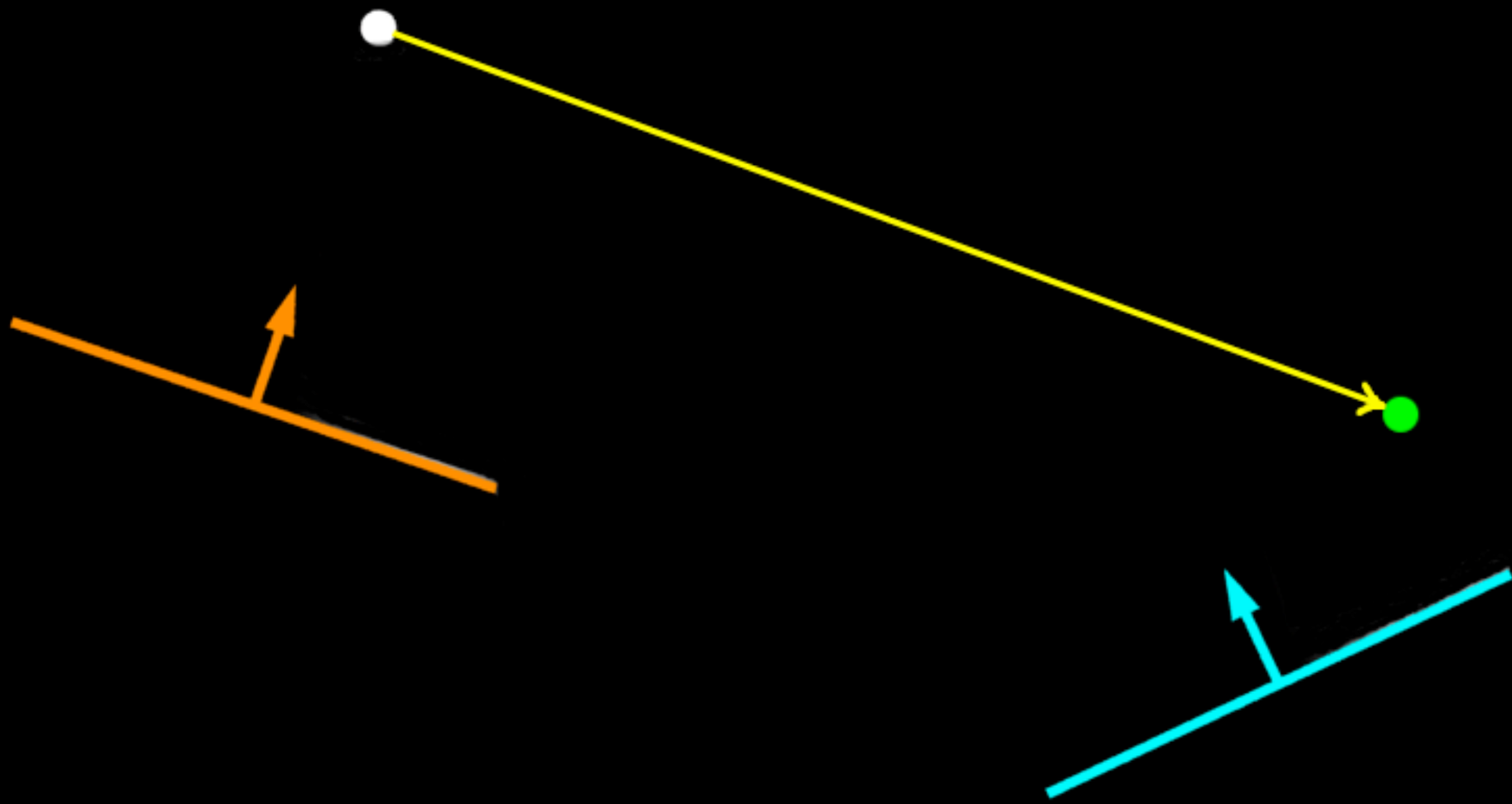
Teleporting light



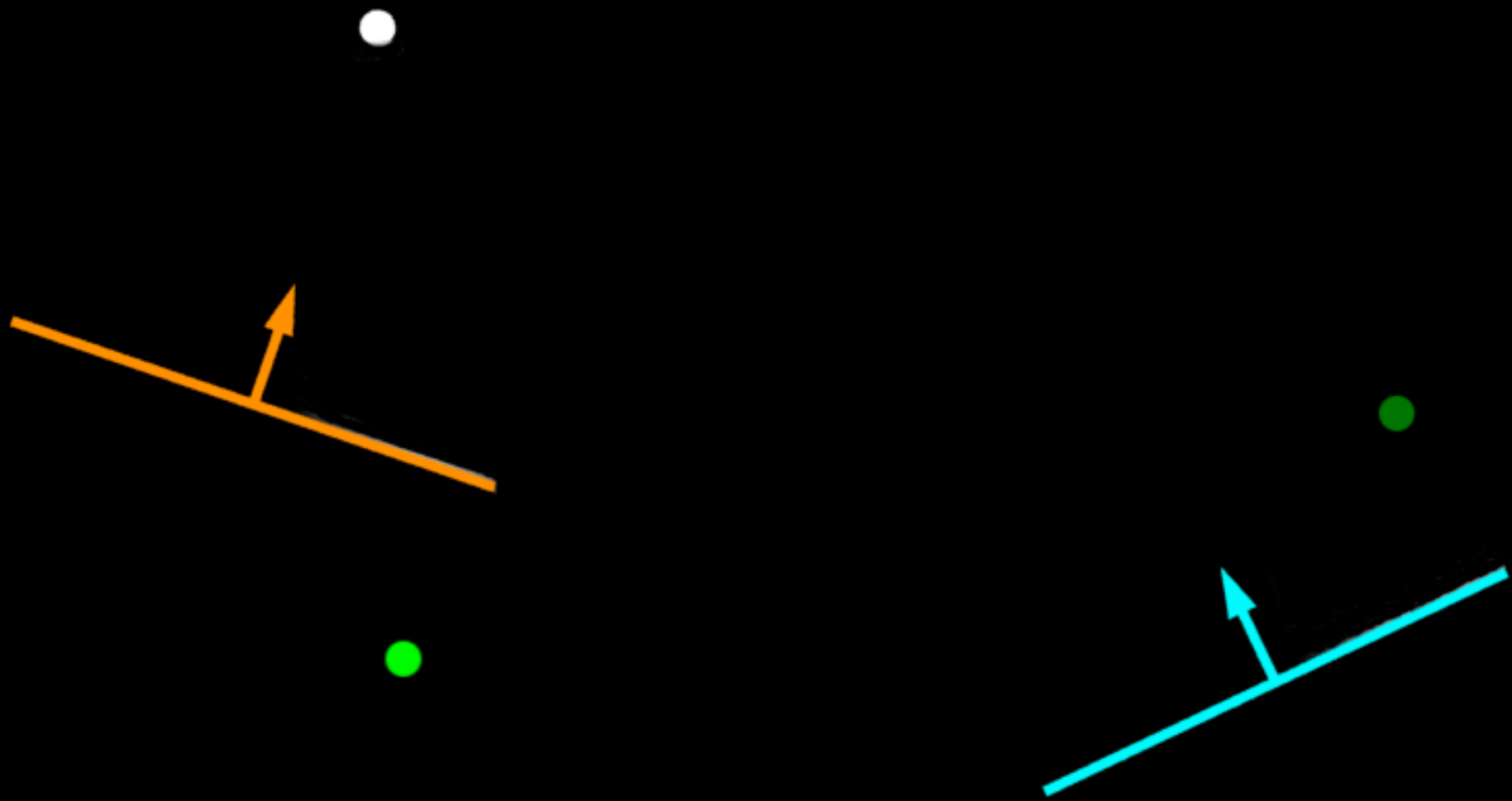
Teleporting light



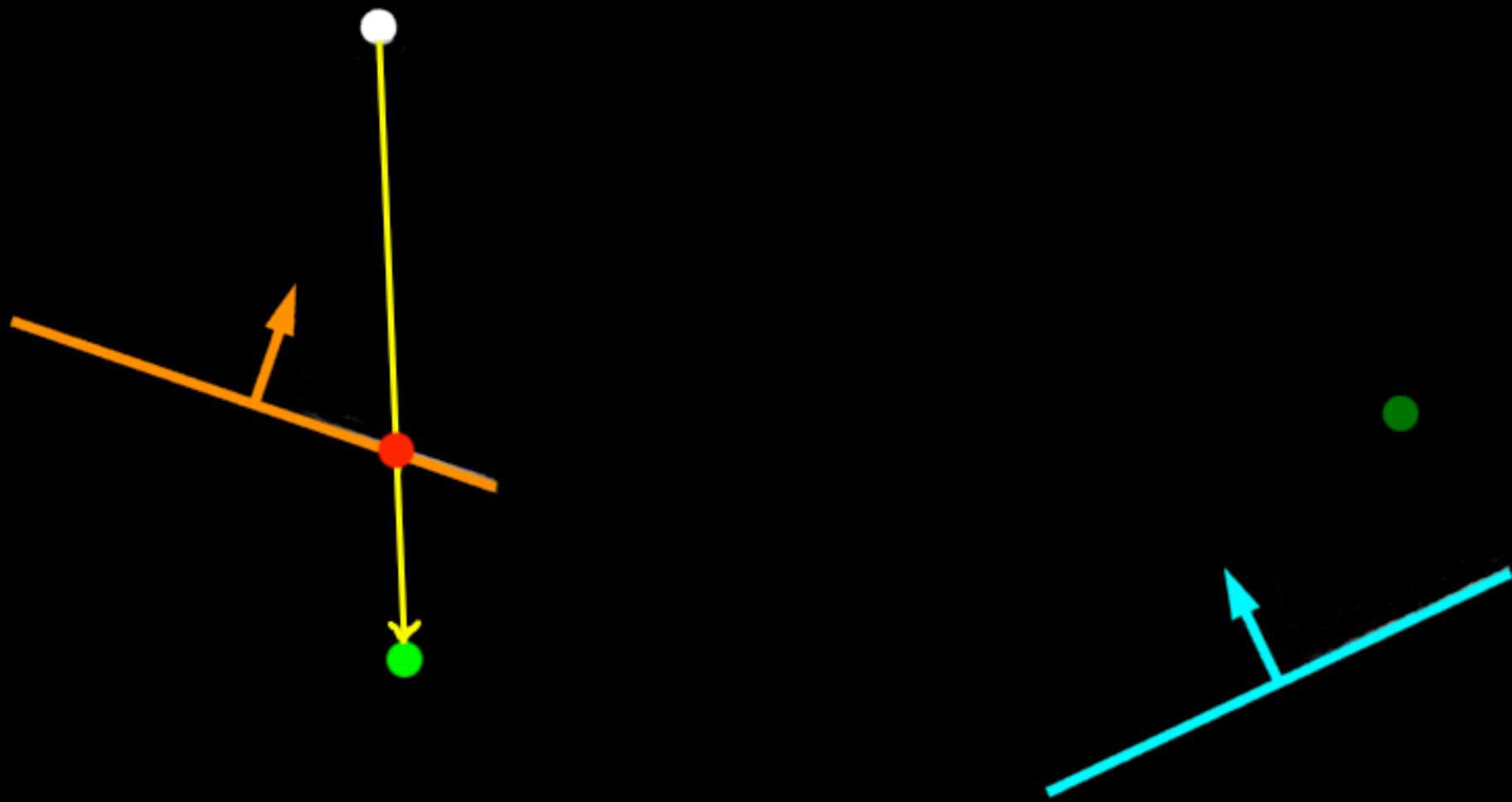
Teleporting light



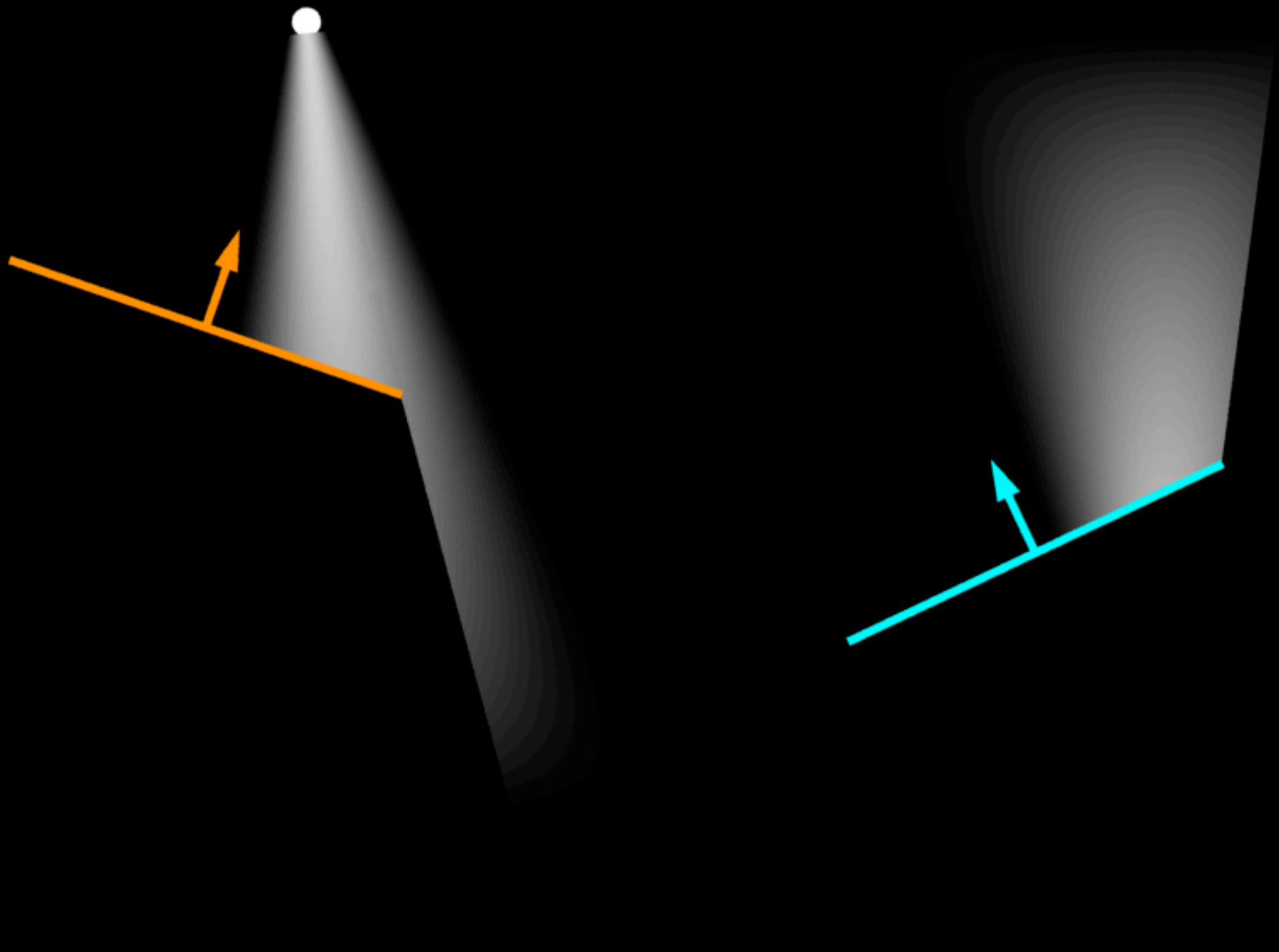
Teleporting light



Teleporting light



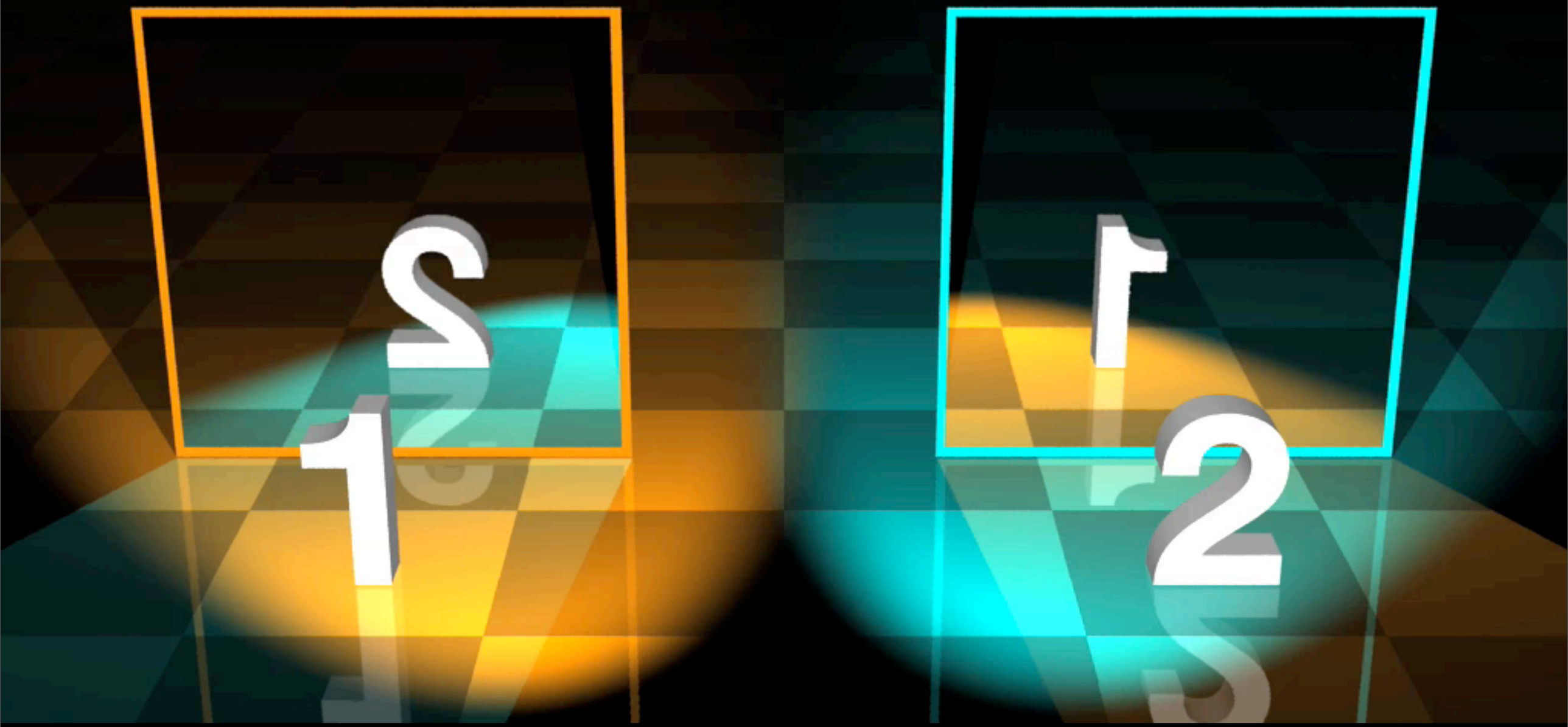
Teleporting light



Teleporting light

```
float isLightTeleported( point pCur; vector lCur; string sendingPortalSpace )
{
   point pPortal = transform( "current", sendingPortalSpace, pCur );
   if( pPortal[ 2 ] < 0 )
   {
      vector lPortal = vtransform( "current", sendingPortalSpace, lCur );
      point pAtPortal = pPortal - lPortal * ( pPortal[ 2 ] / lPortal[ 2 ] );
      if( abs( pAtPortal[ 0 ] ) < 0.5 && abs( pAtPortal[ 1 ] ) < 0.5 )
      {
         return 1;
      }
   }
   return 0;
}
```

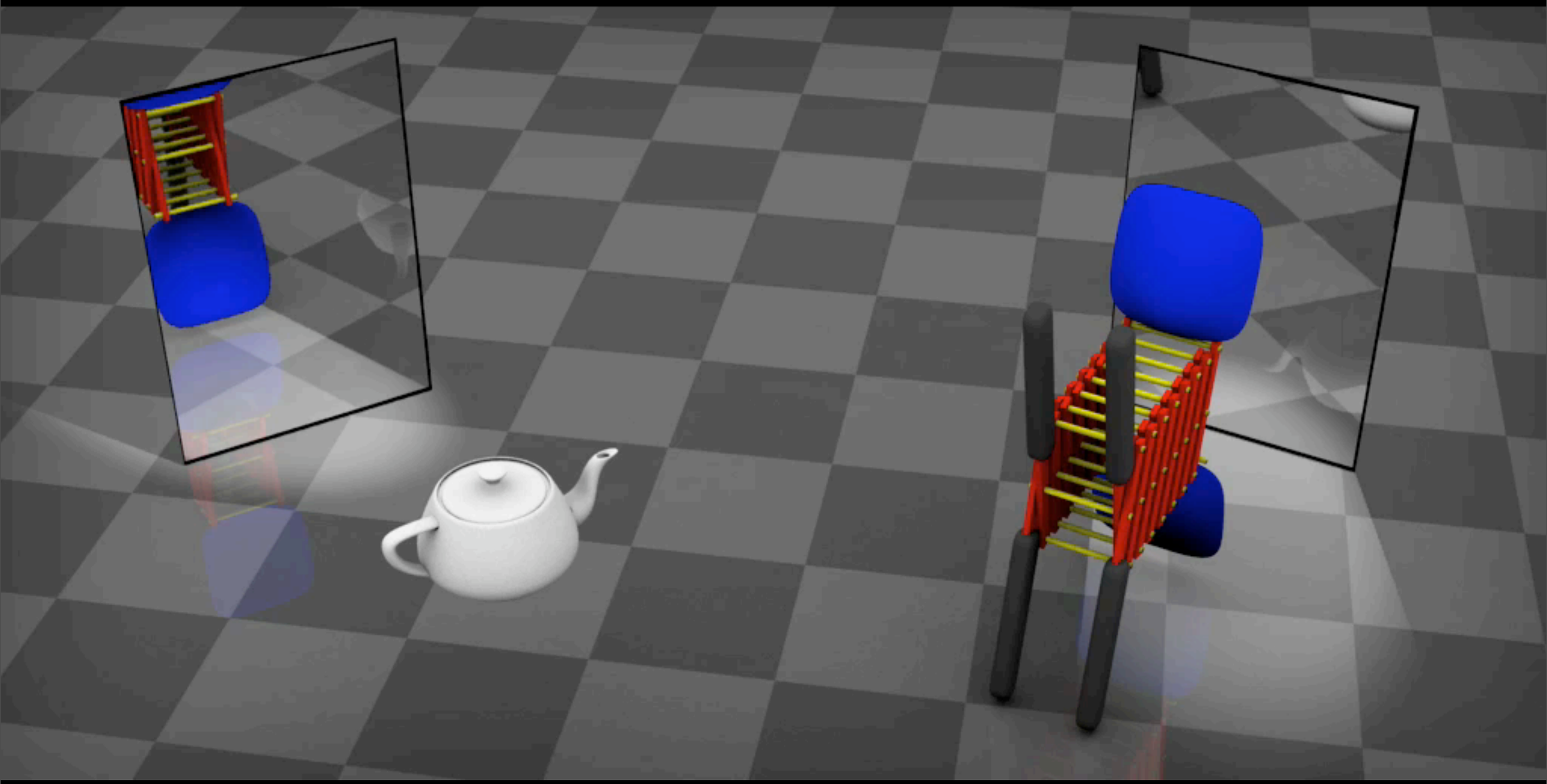
Teleporting light



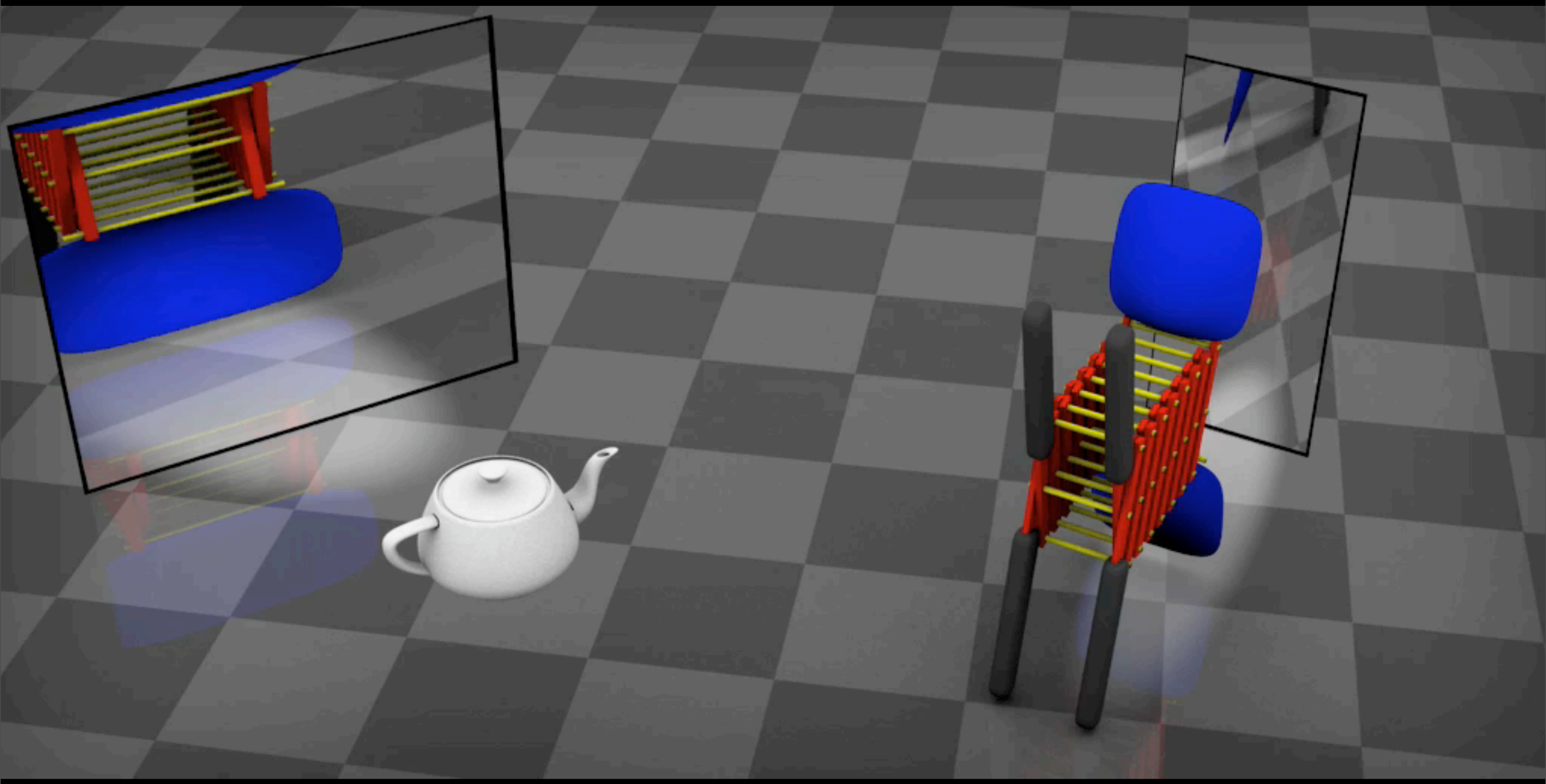
Teleporting geometry

```
displacement geometryTeleport(
   string sendingPortalSpace = ""; string receivingPortalSpace = ""
)
{
   point pPortal1, pPortal2, pCurNew;
   normal nPortal1, nPortal2, nCurNew;
   transformThroughPortal( P, sendingPortalSpace, receivingPortalSpace,
         pPortal1, pPortal2, pCurNew );
   ntransformThroughPortal( N, sendingPortalSpace, receivingPortalSpace,
         nPortal1, nPortal2, nCurNew );
   if( abs( pPortal1[ 0 ] ) < 0.5 && abs( pPortal1[ 1 ] ) < 0.5 && pPortal1[ 2 ] < 0 )
   {
      P = pCurNew;
      N = nCurNew;
   }
}
```

Teleporting geometry



Stretching portals



Shader objects

```
class geometryTeleport(
   string portalSpace1 = ""; string portalSpace2 = ""; float portalThickness = 0.01
)
{
   varying float inTransitionRegion;
   public void begin() { inTransitionRegion = 0; }
   public void displacement( output varying point P; output varying normal N )
   {
      ...
      if( ... && abs( pPortal1[ 2 ] ) < portalThickness )
      {
         inTransitionRegion = 1;
      }
   }
   public void surface( output varying color Ci, Oi )
   {
      Ci = Oi = color ( 1 - inTransitionRegion );
   }
}
```

Using co-shaders

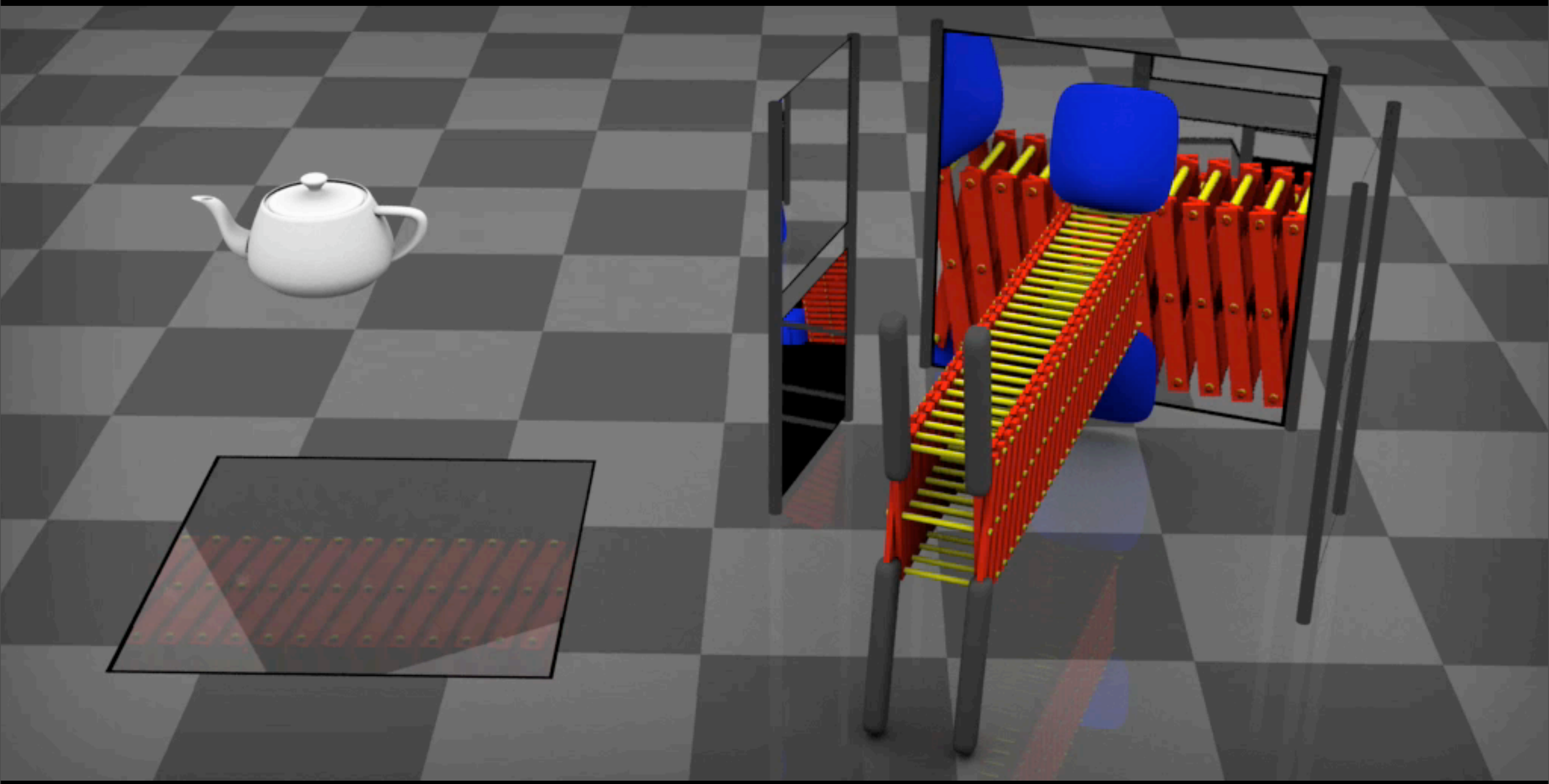
```
class teapot()
{
   shader geoTeleport;
   shader lgtTeleport;
   public void begin()
   {
      geoTeleport = getshader( "geoTeleport" );
      lgtTeleport = getshader( "lgtTeleport" );
   }
   public void displacement(
      output point P; output normal N
   )
   {
      if( geoTeleport != null )
      {
         geoTeleport->displacement( P, N );
      }
   }
```

```
   public void surface(
      output color Ci, Oi
   )
   {
      if( geoTeleport != null )
      {
         geoTeleport->surface( Ci, Oi );
      }
      color lit = 0;
      if( lgtTeleport != null )
      {
         lgtTeleport->surface( lit, Oi );
      }
      color facing = color
               normalize( N ) . normalize( -I );
      Ci *= Cs * mix( lit, facing, 0.5 );
   }
}
```

Multi-teleportation

```
class geometryTeleport( string portalSpaces[] = {}; float portalThickness = 0.01 )
{
   ...
   public void displacement( ... )
   {
      uniform float i;
      for( i = 0; i < arraylength( portalSpaces ); i += 1 )
      {
         // (0,1), (1,0), (2,3), (3,2), ...
         string portalSpace1 = portalSpaces[ i ];
         string portalSpace2 = portalSpaces[ i + 1 - 2 * mod( i, 2 ) ];
         ...
      }
   }
   ...
}
```

Multiple portals



“Awesome!” / “Huh?!?”

Thanks to everyone at MPC for their support
(we’re hiring, by the way)

Full code and test scenes available

robpieke@gmail.com