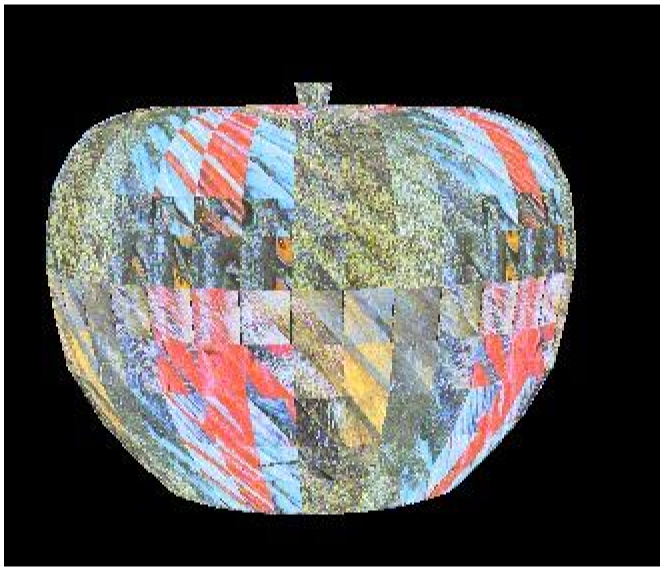
Texture Mapping: Project Report

Vishesh Agarwal, 15CS30040
S Nishok Kumar, 15CS30024



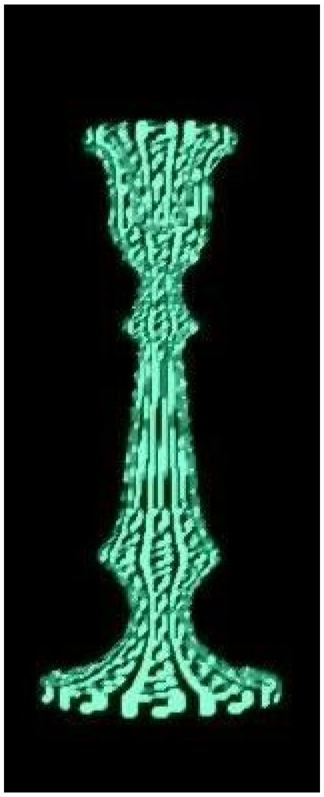
Code: https://github.com/kudimagan/Texture-Mapping

Week - 1

In the first week of work, we created our Graphics Library, which contained structures to parse object files, represent triangles and appropriate functions for ray tracing. In addition, we built structures to represent colours in RGB, HSV and perform general operations on them. We used a preliminary spherical mapping function to map texture to the object. The results were somewhat below average, checkers appearing as lumps.

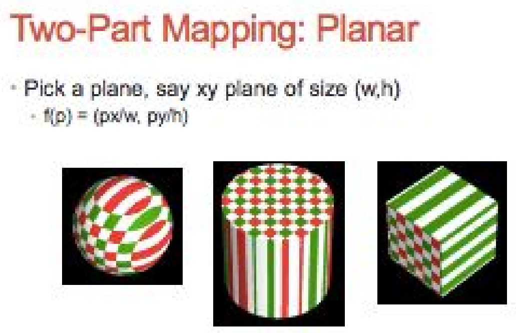
$$u = 0.5 + \frac{\arctan 2(d_z, d_x)}{2\pi}$$
$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}$$

^ Texture Mapping Equations used



Week - 2

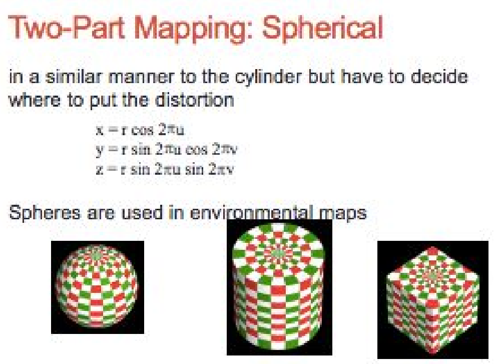
In the second week of work, we implemented more advanced texture mapping functions. (Fig 1-3). The results were better than before, but were still blurry. Furthermore there was splashing of colours, i.e. colours from one patch would spill onto differently colored patches.



Two-Part Mapping: Planar

- Pick a plane, say xy plane of size (w,h)
  - f(p) = (px/w, py/h)

Fig 1



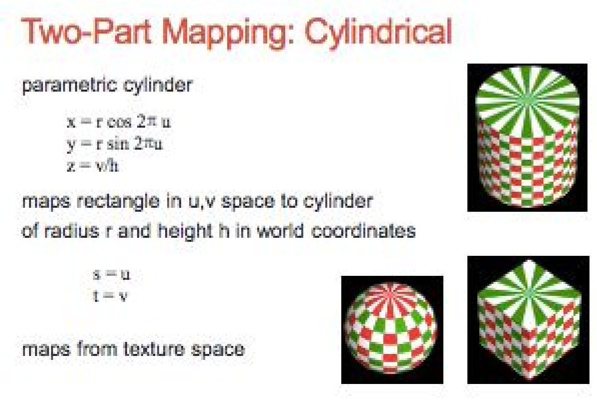
Two-Part Mapping: Spherical

in a similar manner to the cylinder but have to decide where to put the distortion

$x = r \cos 2\pi u$
$y = r \sin 2\pi u \cos 2\pi v$
$z = r \sin 2\pi u \sin 2\pi v$

Spheres are used in environmental maps

Fig 2



Two-Part Mapping: Cylindrical

parametric cylinder

$x = r \cos 2\pi u$
$y = r \sin 2\pi u$
$z = v/h$

maps rectangle in u,v space to cylinder of radius r and height h in world coordinates

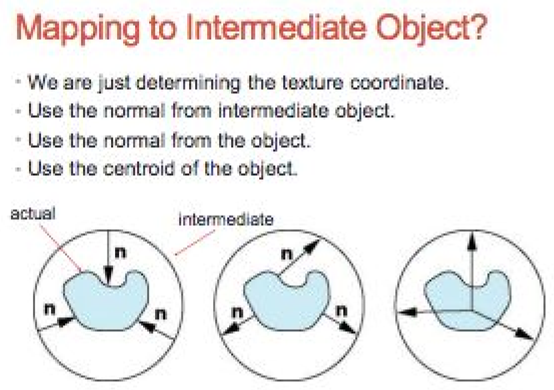
$s = u$
$t = v$

maps from texture space

Fig 3

We believed that it was due to floating point precision issues. We tried to account for the rounding error by adding thresholds and tried to use different methods for rounding. E.g. instead of rounding both (int) 4.2 and (int) 4.7 to 4, we round (int) 4.2 to 4 and (int) 4.7 to 5. However, none of these methods improved the quality of images.

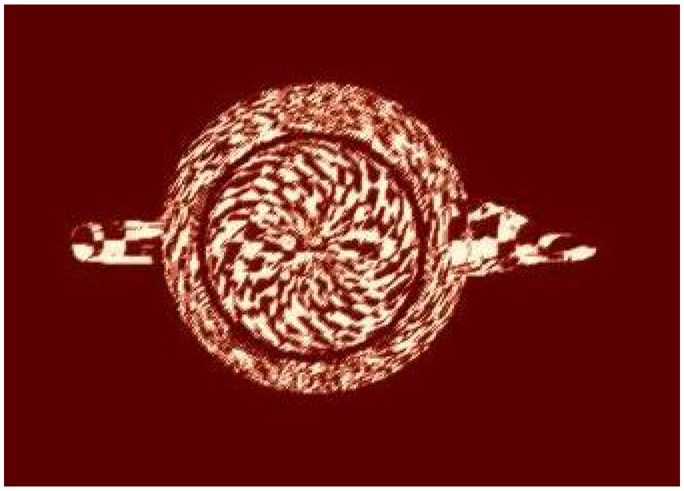
Another thing we tried in our second week was how to close in a spherical map on our non-spherical object (Fig 4). We tried using normals from the object and also tried using normals from the sphere. Projecting normals from object gave very mazy images (eg: Week 2/mug 1 and Week 2/mug 2). Projecting normals from sphere gave better results (eg: Week 2/bunny 9)



Mapping to Intermediate Object?

- We are just determining the texture coordinate.
- Use the normal from intermediate object.
- Use the normal from the object.
- Use the centroid of the object.

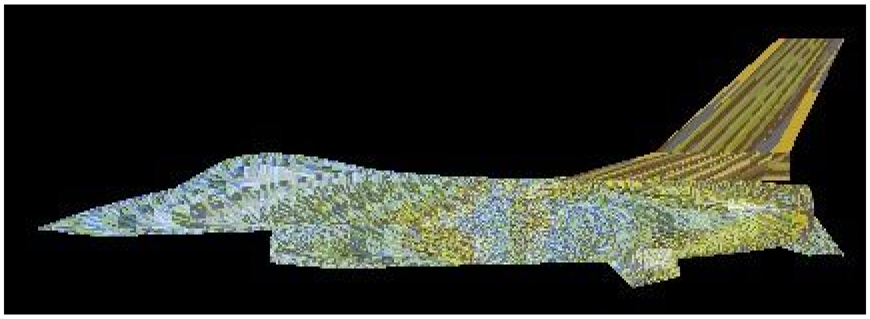
Fig 4



Week - 3

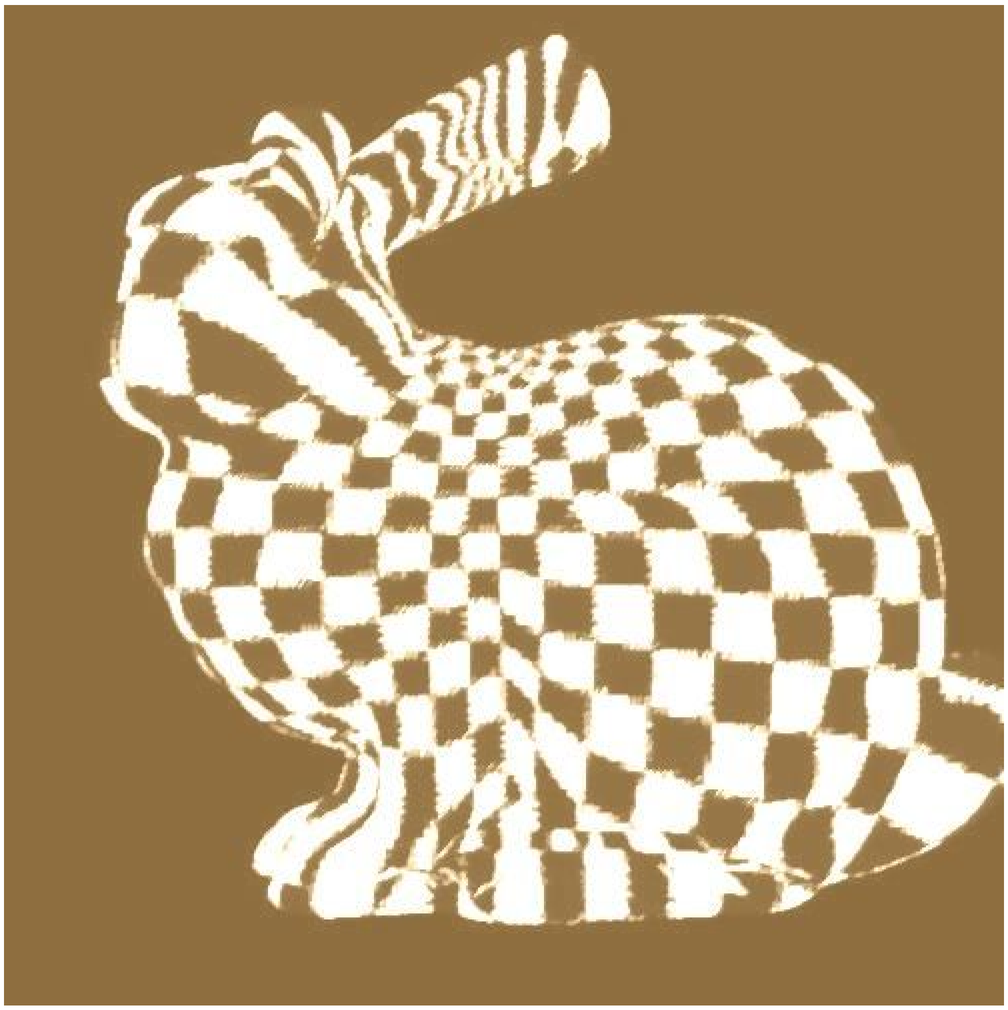
In the third week of work, we shifted our focus from checkerboard textures that we were generating within the code itself, and worked on applying textures from image files. Working on the texture files showed promising results. It also strengthened our belief that the issue in checkerboard textures was that of floating point precision only.

We also tried to work on the floating point precision problem using interpolation from triangle vertices to the interior points. The splashing problem was somewhat reduced but not completely. However, the images got somewhat blurry (compare Week 2/bunny 11 with Week 3/sphere 0)



Week - 4

In the final week of our work, we had to get around the problem of colour spilling. We bettered our fixed point interpolation by using smoothening. Thus, instead of looking at the individual pixel where a ray strikes, we look at the patch around it to find the interpolated colour for that pixel. This smoothening gave us demonstrable results.
The final higher resolution images are kept in Final Images folder and the images generated during the development phase have been kept in the Development Images folder.



References

We looked into the following papers for texture mapping references. However, the concept of using quantized minimization of energy of cross fields over surface, which leads to the best smoothest global seamless parameterization was too overwhelming for us to implement.

1. **Single patch, fixed boundary**: these algorithm can parametrize a disk-like part of the surface given fixed 2D positions for its boundary. These algorithms are efficient and simple, but they usually produce high-distortion maps due to the fixed boundary.
2. **Single patch, free boundary:** these algorithms let the boundary deform freely, greatly reducing the map distortion. Care should be taken to prevent the border to self-intersect.
3. **Global parametrization**: these algorithms work on meshes with arbitrary genus. They initially cut the mesh in multiple patches that can be separately parametrized. The generated maps are discontinuous on the cuts (often referred as *seams*).
4. **Global seamless parametrization**: these are global parametrization algorithm that hides the seams, making the parametrization “continuous”, under specific assumptions that we will discuss later.

Papers:

1. Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, Werner Stuetzle. Multiresolution Analysis of Arbitrary Meshes, 2005. ↩
2. Bruno Lévy, Sylvain Petitjean, Nicolas Ray, Jérome Maillot. Least Squares Conformal Maps, for Automatic Texture Atlas Generation,, 2002. ↩
3. Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, Steven J. Gortler. A Local/Global Approach to Mesh Parameterization, 2008. ↩
4. David Bommes, Henrik Zimmer, Leif Kobbelt. Mixed–integer quadrangulation, 2009. ↩
5. Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, Mirela Ben–Chen. Directional Field Synthesis, Design, and Processing, 2016 ↩