Computer Graphics

Project 1 - Image Processing

Name: Oscar Stark

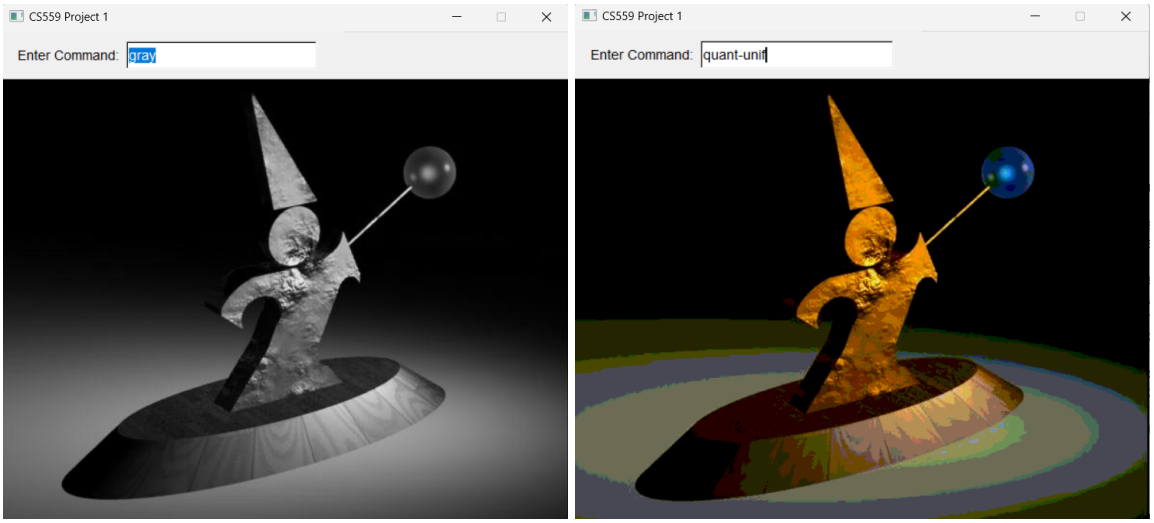
Use the function “load” to load an image in tga format. Example: “load wiz.tga”.
Image Editing Functions:

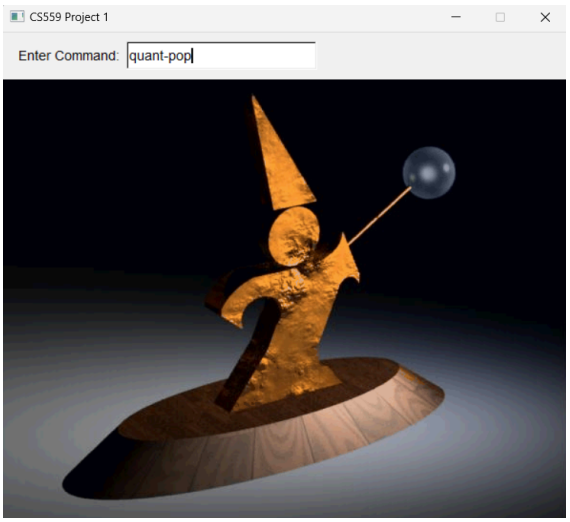
1. **gray (Color to Grayscale):** This function converts the image to grayscale using the formula I = 0.299r + 0.587g + 0.114b. The formula is like this because we perceive intensity from red, green and blue differently.
2. **quant-unif (Uniform Quantization):** This function converts the image from 24 bit color to 8 bit color. This is done using 4 shades of blue, 8 shades of red and 8 shades of green.
3. **quant-pop (Populosity):** This function converts the image from 24 bit color to 8 bit color. The 256 most popular colors from the original image are used to display the quantized image. This means that important but unpopular colors are not represented, for example the blue ball from “wiz.tga”. In the code, first a histogram with the count of each color is made, then the 256 most popular colors are chosen, and finally each pixel of the original image is changed to the closest available color.
4. **dither-thresh (Naive Dithering):** This function dithers the image by making pixels with intensity higher than 0.5 white and pixels with intensity lower than 0.5 black.
5. **dither-bright (Brightness Preserving Dithering):** This function dithers the image using a threshold that keeps average intensity the same. This is done by calculating the average intensity, then storing all intensities sorted in an array, and choosing the k-th percentile that keeps average intensity the same.
6. **dither-rand (Random Dithering):** Random noise ranging from -0.2 to 0.2 is added to all pixels and then naive dithering is applied.
7. **dither-cluster (Clustered Dithering):** The image is dithered in blocks. In the block each pixel uses a different threshold to decide if it is black or white.
8. **dither-fs (Floyd-Steinberg Dithering):** This function dithers the image by making pixels with intensity higher than 0.5 white and pixels with intensity lower than 0.5 black, after this the error is propagated to nearby pixels. The error is the difference between the original color and the final color.
9. **dither-color (Color Floyd-Steinberg Dithering):** This is like the Floyd–Steinberg dithering but with colors. The final image uses 8 shades of red, 8 shades of green and 4 shades of blue. Each pixel color channel is changed to the closest shade of color, and the error is the difference between the original value and the final value.
10. **filter-box (Box Filter):** A 5x5 filter where every element has the same weight is applied to each pixel. This operation blurs the image. The values out of range of the image (on the border of the image) are 0.
11. **filter-bartlett (Bartlett Filter):** A 5x5 Bartlett filter is applied to each pixel. This blurs the image. The values out of range of the image (on the border of the image) are 0.
12. **filter-gauss (Gaussian Filter):** A 5x5 Gaussian filter is applied to each pixel. This blurs the image. The values out of range of the image (on the border of the image) are 0.

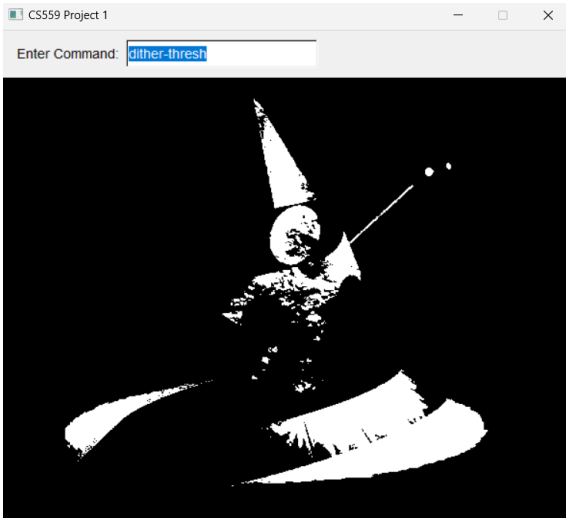
13. **filter-gauss-n (Arbitrary-Size Gaussian Filter):** Example: “filter-gauss-n 7”, this example would apply a 7x7 gauss filter to the current image. Applies a NxN Gaussian filter to each pixel. The NxN matrix has to be constructed first. We made this matrix using the exact integer values, but this makes computation unnecessarily slow. An approximation of the values could be used.
14. **filter-edge (Edge Detection):** A high pass filter is applied to each pixel to detect edges. We made this filter using the 5x5 Gaussian filter. The negative results from the high pass filter are changed to 0.
15. **filter-enhance (Edge Enhance):** The results from the edge detection are added to the original image, truncating color values at 255 (because color values go from 0 to 255).
16. **half (Half Size):** This function halves the image size. The reconstruction is done using a 3x3 Bartlett filter. We used backward mapping.
17. **double (Double Size):** This function doubles the image size. The reconstruction is done using Bartlett filters. Depending on the coordinates of the resulting pixel, a 4x4, 3x3, 3x4 or 4x3 filter is used. We used backward mapping.
18. **rotate (Arbitrary Rotation):** Example: “rotate 45”, this example would rotate the current image 45 degrees clockwise. This function rotates the image by the given angle in degrees. The reconstruction is done using Bartlett filters. We used backward mapping.

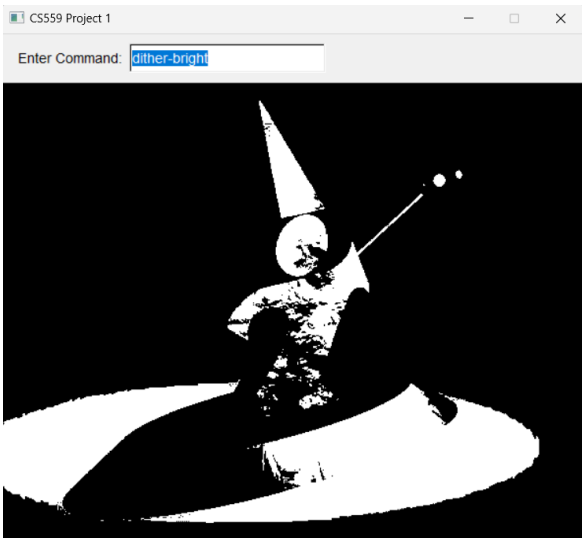
Annex:

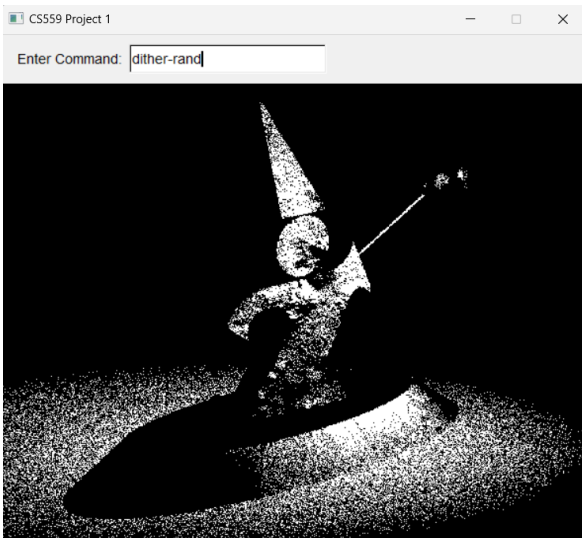
The following images show results of the functions in order.

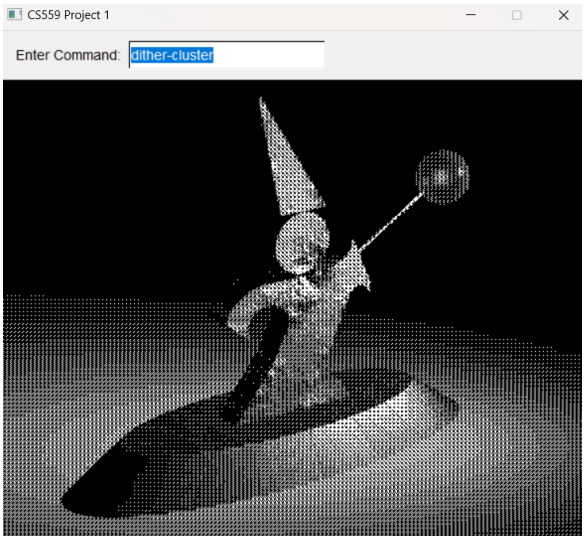


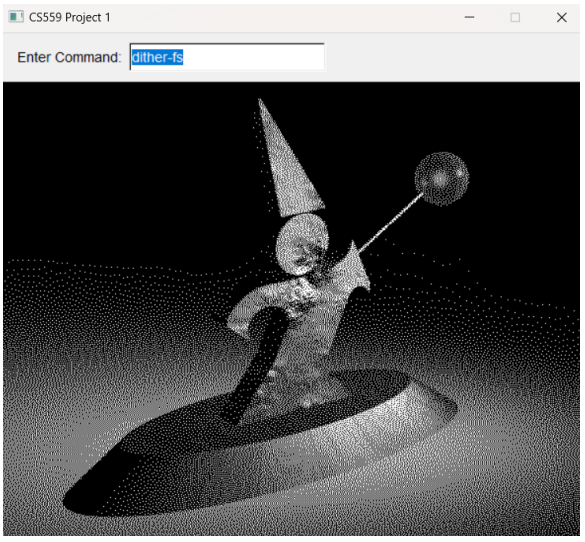

CS559 Project 1
Enter Command: quant-pop

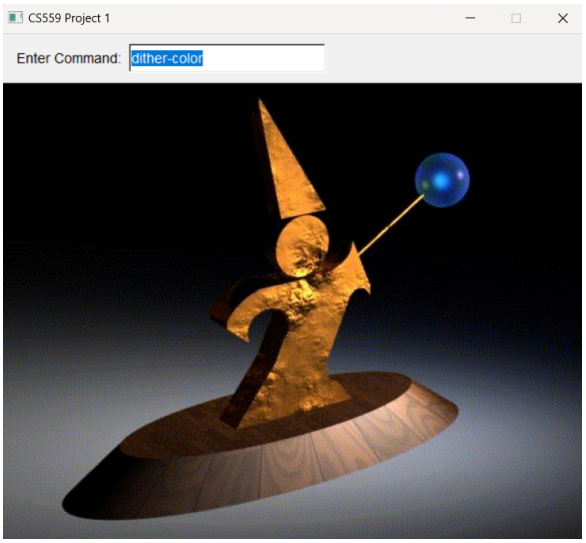

CS559 Project 1
Enter Command: dither-thresh

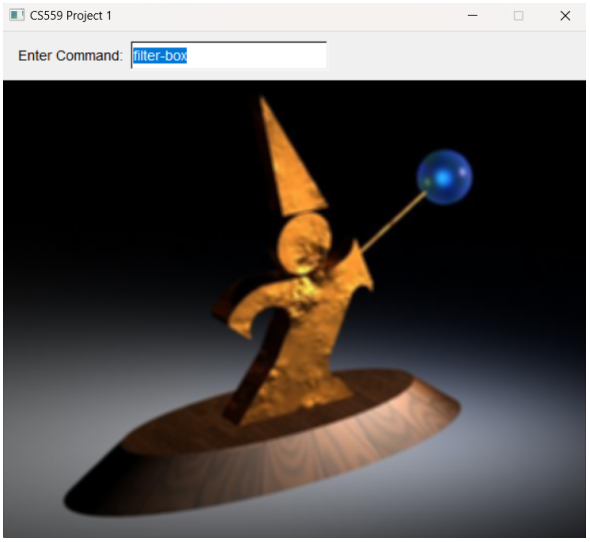

CS559 Project 1
Enter Command: dither-bright

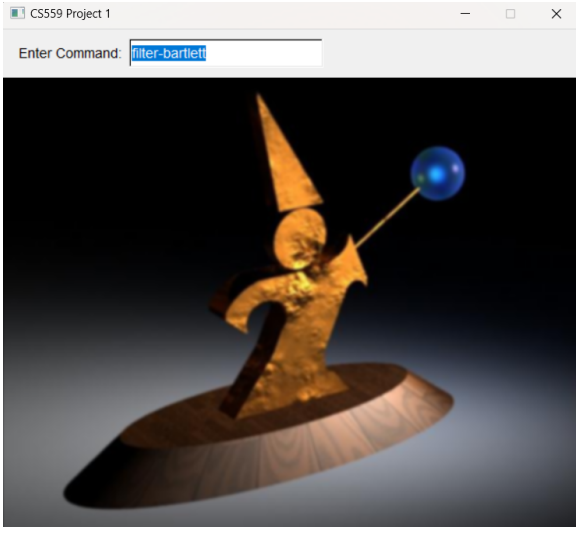

CS559 Project 1
Enter Command: dither-rand

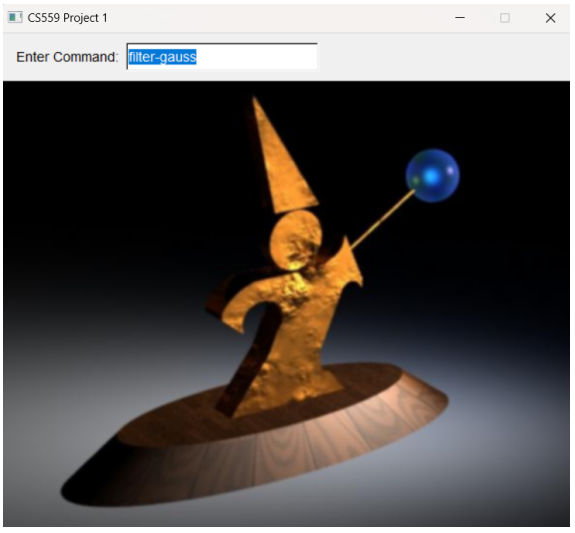

CS559 Project 1
Enter Command: dither-cluster

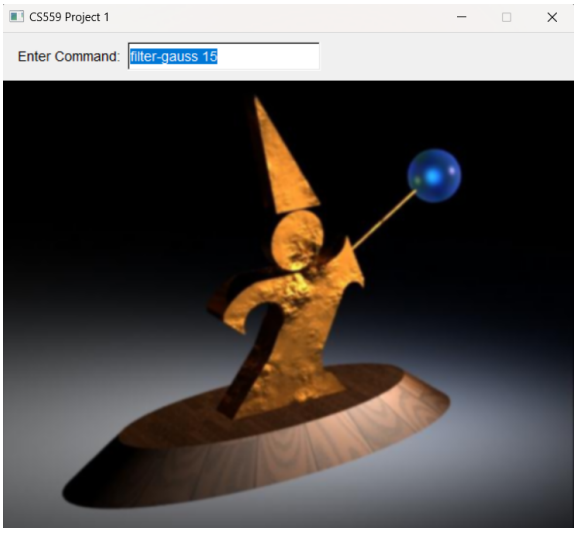

CS559 Project 1
Enter Command: dither-fs

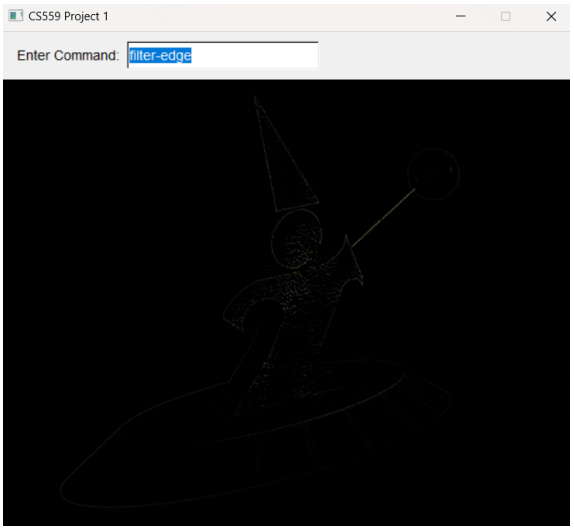

CS559 Project 1
Enter Command: dither-color

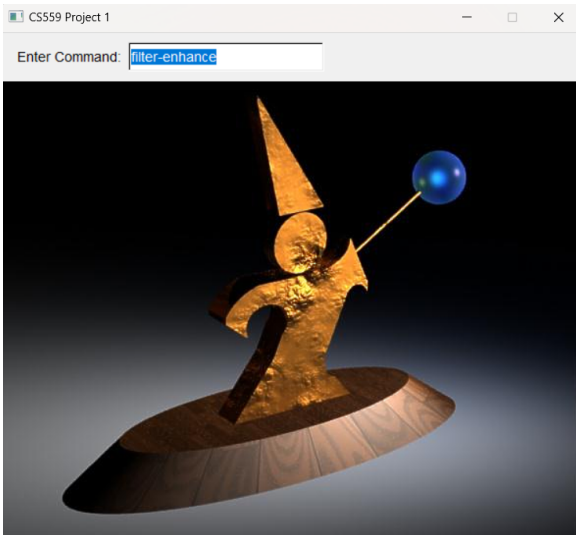

CS559 Project 1
Enter Command: filter-box

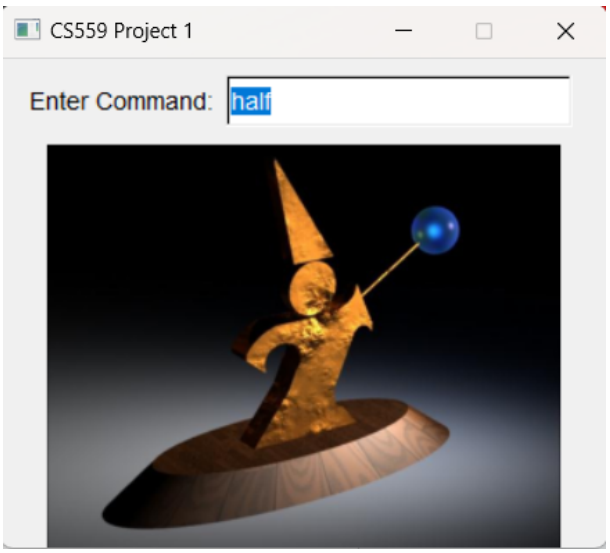

CS559 Project 1
Enter Command: filter-bartlett

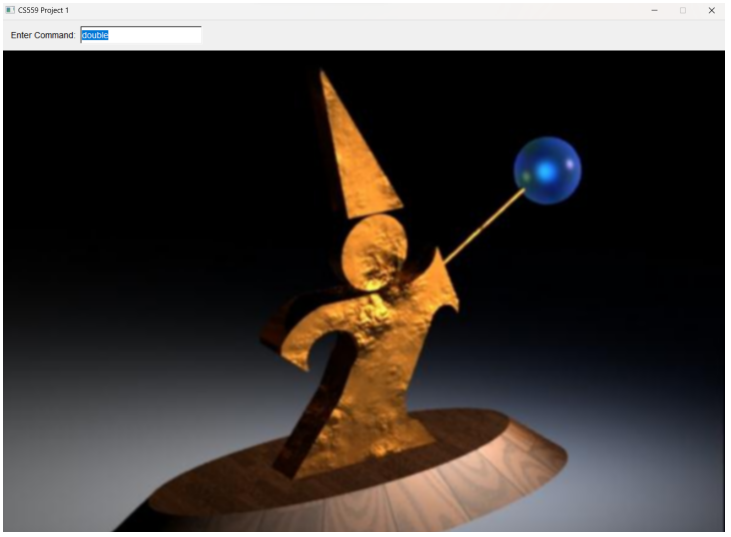

CS559 Project 1
Enter Command: filter-gauss

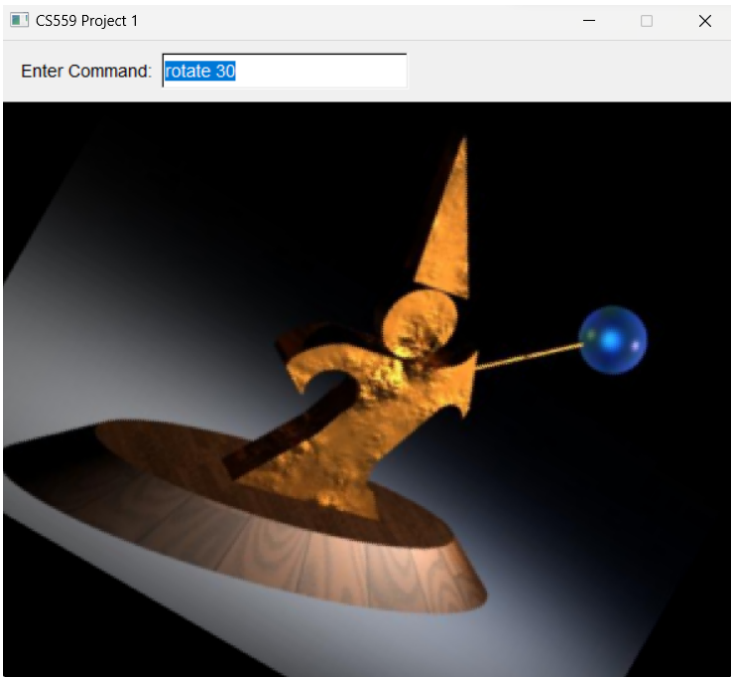

CS559 Project 1
Enter Command: filter-gauss 15


CS559 Project 1
Enter Command: filter-edge


CS559 Project 1
Enter Command: filter-enhance


CS559 Project 1
Enter Command: half


CS559 Project 1
Enter Command: double


CS559 Project 1
Enter Command: rotate 30