Ryan Dang

05/20/19

CS443

SnapMat Project

1. **Project Statement**

My application is called “SnapMat” and the reason for this is because it allows people to take a picture of anything and turns them into textures used by many 3D programs such as Blender, Unity, and Unreal Engine. The “Mat” comes from “Material” which is a common term for the result of all of these textures coming together. This app lets people create these textures easier since it can be difficult to find textures that people would want, and this will mostly benefit people that use these types of textures daily such as 3D modelers. This is similar to a desktop app named “Materialize” which can take an image and turn it into a heightmap, a normal map, metallic map, and many other types of material textures. To use this app you need to give permissions to read/write and the camera. Since it’s use is supposed to be for taking pictures when on the go it’s mainly for phones.

1. **Application Design**

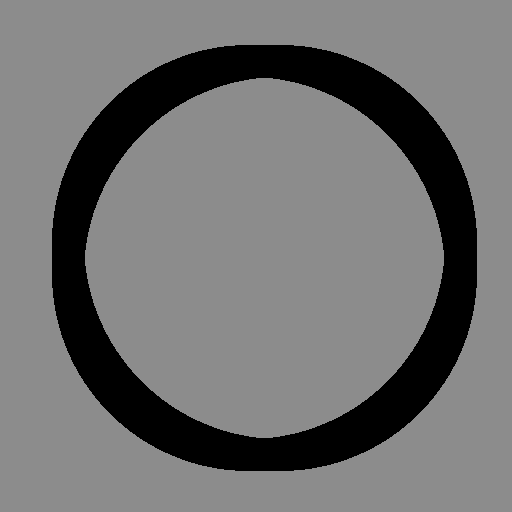
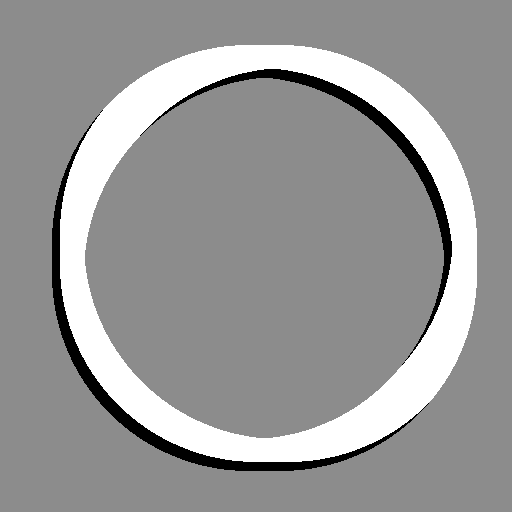
The app has a single activity that opens the camera or gallery activity to get an image to be used in the main activity. The camera and gallery will return URI which the main activity will take and use for turning the image into a bitmap and will also resize it. This bitmap will be used for modifying by creating a BitmapDrawable which allows the bitmap’s pixels to be modified. Then for image processing it will iterate each and every pixel modifying it by turning the image into grayscale using V = (pixel.red \* 0.3 + pixel.green \* 0.6 + pixel.blue \* 0.1). This is a weighted average to give better contrast in colors and it’s based on how we perceive more green and reds than blues.

This is how it looks when the colors are averaged for the grayscale.

This is how it looks when they colors have a weighted average, notice the woman’s shirt having much more contrast

Images From: <http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/>

This main activity has seek bar listeners that wait for the changes to the seek bars. These seek bars are used as inputs for changing the values for the heightmap. The levels algorithm for changing the brightness is NV = Min(255, Max(0, (V - L) \* 255 / (H - L))) taken from <https://stackoverflow.com/questions/39172741/gimp-colour-level-algorithm>.

The heightmap -> normal map algorithm takes the surrounding pixels and subtracts the left and right, and the above and below pixels from each other.  
 

Example of the process above.

1. **Implementation and Evaluation**

The app works fine, but is a bit slow this is because it is doing each calculation per pixel of the image. I suppose a faster method would be using the surface view. During initial testing errors popped up when the image had a large resolution so the image so to remedy this the application would scale the image down to 512x512.

1. **References**

Grayscale: <http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/>

Levels: <https://stackoverflow.com/questions/39172741/gimp-colour-level-algorithm>

Normal Map: <https://www.gamedev.net/forums/topic/475213-generate-normal-map-from-heightmap-algorithm/>

1. **Experiences and Thoughts**

I felt like improving the speed of the program would make it much more useful, and having a 3D model to exactly view how the textures would change the model would be very nice features.