Computer Graphics
Project Presentation

Name:- Ruchitra Kumaresan
Id:- 1304279

Topics:-

1. Vertex and Fragment program
2. Projective Texturing

Context

- Introduction
- Code
- Execution(Before & After)
- Conclusion
- References

2.Vertex and Fragment:-

- Vertex and fragment programs are fundamental components of modern graphics processing units (GPUs) that enable developers to write custom shaders for rendering realistic 3D graphics.
- Vertex programs, also known as vertex shaders, manipulate the properties of vertices (points) in 3D models, such as position, color, and texture coordinates, to achieve effects like deformation and animation.

27.Projective texturing:-

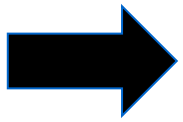
- Projective texturing is a technique used in computer graphics to enhance the realism of rendered scenes by applying textures to objects in a way that considers the perspective of the viewer.
- Traditional texturing applies textures based on the object's surface coordinates, which can lead to distortions when the object is viewed from different angles.
- Projective texturing, on the other hand, uses a projection matrix to calculate texture coordinates based on the object's position relative to the viewer, resulting in more accurate texture mapping and improved visual quality.

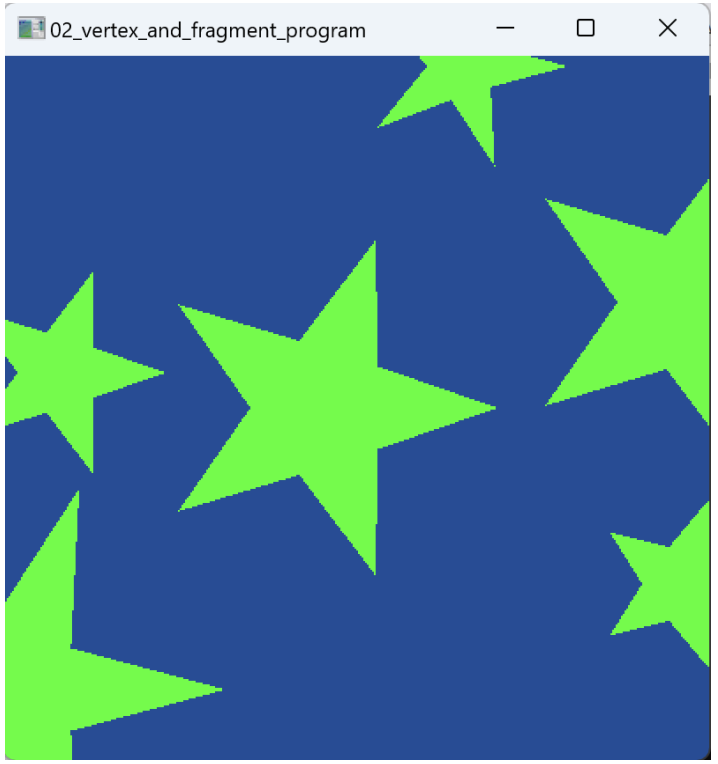
Vertex and Fragment (Before):

```
glClearColor(0.1, 0.3, 0.6, 0.0);  /* Blue background */
```

e)

background color as Blue.





Vertex and Fragment (After):-

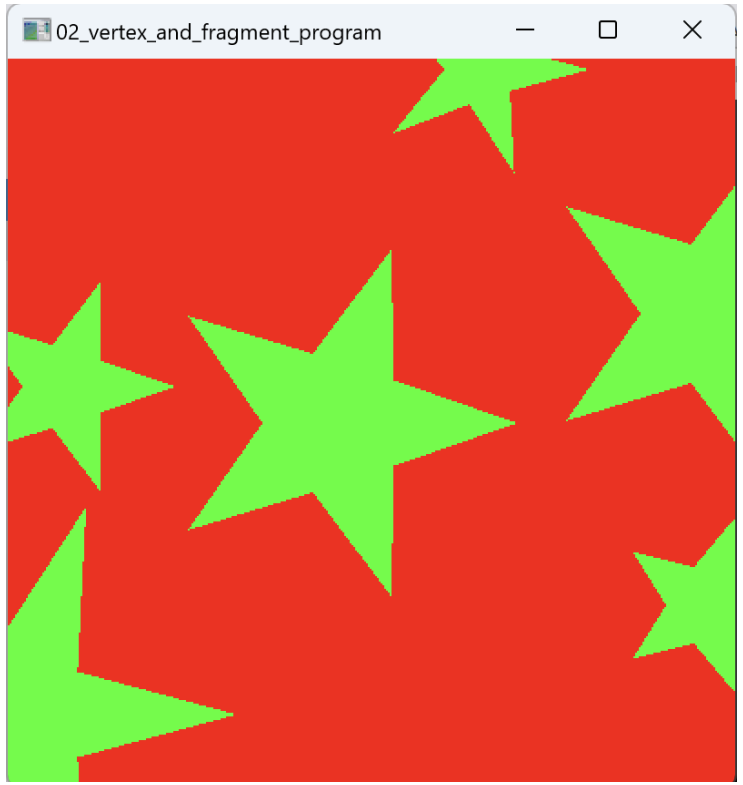
```
glClearColor(0.0, 0.0, 0.0, 1.0);  /* Black background */
```

It represents black (RGB values are all 0.0) with full opacity (alpha is 1.0), which converts blue background to black.

Vertex and Fragment (After):-

Line 61:-

```
glClearColor(1.0, 0.0, 0.0, 0.0);  /* Red background */
```

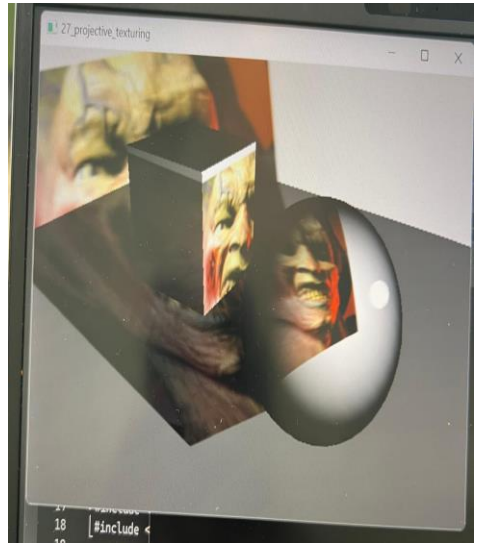


Red: 1.0 (full intensity red)
Green: 0.0 (no green)
Blue: 0.0 (no blue)
Alpha: 0.0 (fully transparent)
This combination of parameters creates a red color with full intensity and fully transparent alpha, resulting in acompletely red background that is transparent.

Projective Texturing:-

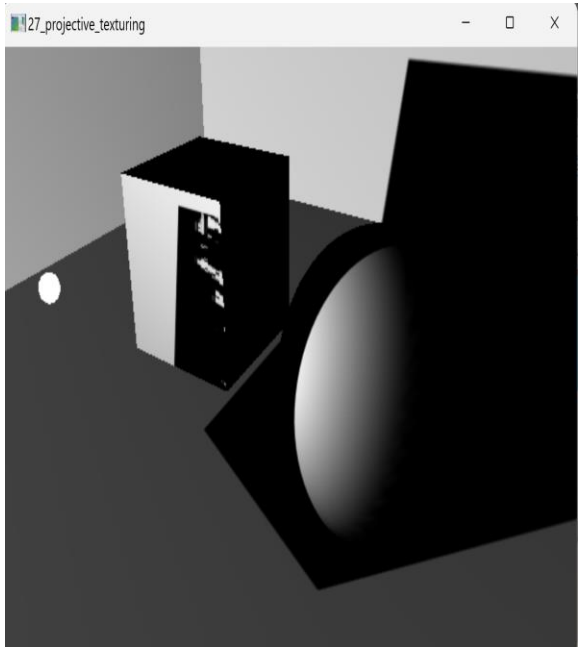
- To change the projection color using grayscale product with vector, which weight and converts to grayscale[0.2126, 0.7152, 0.0722] which effectively removes color information to black and white.

[before]



- To change the direction of projection I used textureMatrix, 4*4 matrix which transforms the input position to texture co-ordinates.

[After]

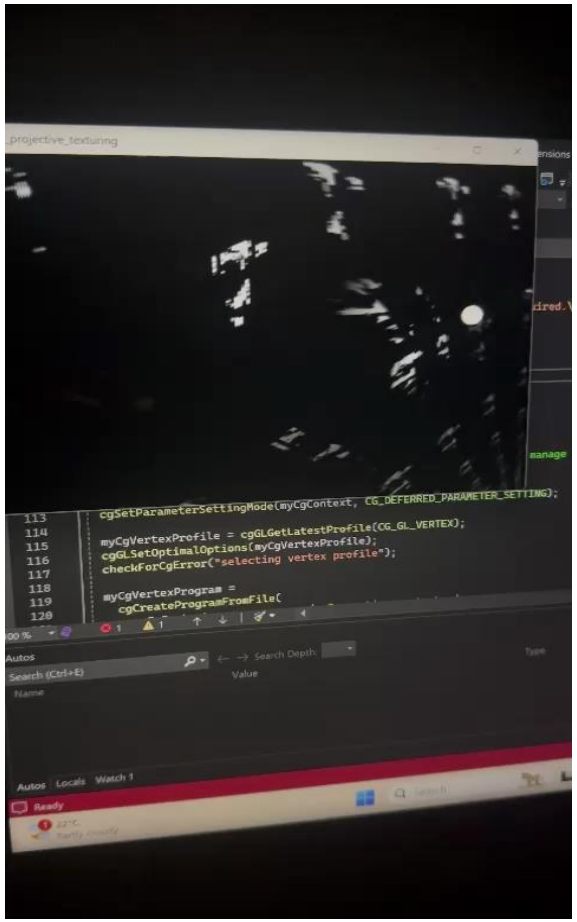


Projective Texturing(Before):-

Here, Textute Matrix is used for text co-ordinate transformation and its also commonly used for various transformation.

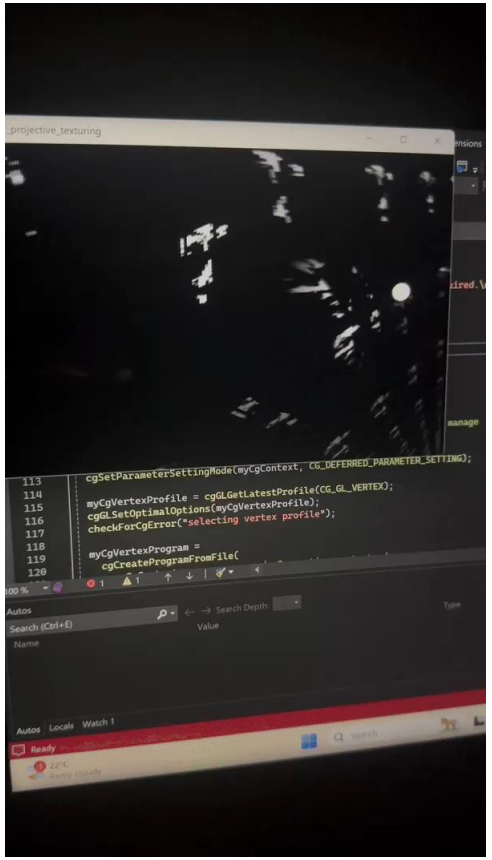
```
void motion(int x, int y)
{
  const float heightMax = 10,
              heightMin = -1.5;

  if (moving) {
    eyeAngle += 0.005*(beginx - x);
    eyeHeight += 0.03*(y - beginy);
    if (eyeHeight > heightMax) {
      eyeHeight = heightMax;
    }
    if (eyeHeight < heightMin) {
      eyeHeight = heightMin;
    }
    beginx = x;
    beginy = y;
    glutPostRedisplay();
  }
  if (movingLight) {
    lightAngle += 0.005*(x - xLightBegin);
    lightHeight += 0.03*(yLightBegin - y);
    xLightBegin = x;
    yLightBegin = y;
    glutPostRedisplay();
  }
}
```



- Projective Texturing(After):-

```
void motion(int x, int y)
{
  const float heightMax = 10,
              heightMin = -1.5;
  if (moving) {
    eyeAngle += 0.001*(beginx - x); //slower reaction
    eyeHeight += 0.01*(y - beginy); //slower vertical reaction
    if (eyeHeight > heightMax) {
      eyeHeight = heightMax;
    }
    if (eyeHeight < heightMin) {
      eyeHeight = heightMin;
    }
    beginx = x;
    beginy = y;
    glutPostRedisplay();
  }
  if (movingLight) {
    lightAngle += 0.001*(x - xLightBegin);
    lightHeight += 0.01*(yLightBegin - y); //slower vertical movement
    xLightBegin = x;
    yLightBegin = y;
    glutPostRedisplay();
  }
}
```



Outcomes

- **Projective Texturing:** Enhances realism by simulating shadows, reflections, and decals on objects, improving visual quality and enabling creative effects.
- **Vertex and Fragment Shaders:** Enable real-time rendering of effects like reflections and dynamic lighting, allow customization of materials and textures, and optimize performance by offloading calculations to the GPU.
- **Overall Impact:** These techniques collectively improve visual quality, realism, and the ability to create custom effects in real-time rendering applications.

Thank You..!