

Scale

```
void main(void) {  
    // compute position  
    vec4 v = vec4(vertex, 1.0);  
    v.x = v.x * 2.0;  
    v.z = v.z * 0.5;  
    gl_Position = _mvProj * v;  
  
    uv = uv1;  
    // compute light info  
    n = normalize(_norm * normal);  
}
```

```
void main(void)  
{  
    gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);  
}
```



Wave

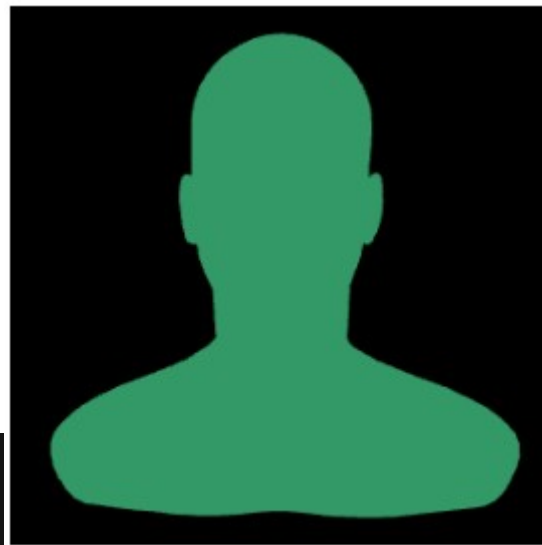
```
void main(void) {  
    // compute position  
    vec4 v = vec4(vertex, 1.0);  
    float s = 1.0 + 0.1*sin(v.s*_time)*sin(v.z*_time);  
    v.y = s * v.y;  
    gl_Position = _mvProj * v;  
  
    uv = uv1;  
    // compute light info  
    n = normalize(_norm * normal);  
}
```

```
void main(void)  
{  
    gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);  
}
```



Phong

```
void main(void) {  
    // compute position  
    vec4 v = vec4(vertex,1.0);  
    normale = _norm * normal;  
    positione = _mv * v;  
    gl_Position = _mvProj * v;  
}
```



```
void main(void)  
{  
    vec3 light = vec3(1.0,1.0,1.0);  
    vec4 lightDiffuse = vec4(1.0,1.0,1.0,1.0);  
    vec4 lightAmbient = vec4(1.0,1.0,1.0,1.0);  
    vec3 lightSpecular = vec3(1.0,1.0,1.0);  
  
    float shi = 10.0;  
    vec3 norm = normalize(normale);  
    vec3 lightv = normalize(light - positione.xyz);  
    vec3 viewv = normalize(positione.xyz);  
    vec3 halfv = normalize(lightv + viewv);  
    vec4 diffuse = max(0.0, dot(lightv, viewv))*diff*lightDiffuse;  
    vec4 ambient = amb*lightAmbient;  
    vec3 specular = pow(max(0.0, dot(norm, halfv)), shi) * spec * lightSpecular;  
    vec3 color = vec3(ambient.xyz + diffuse.xyz + specular);  
    gl_FragColor = vec4(color, 1.0);  
}
```