
Farm
Elle

Precision Agriculture: Using technology to improve agricultural productivity and sustainability

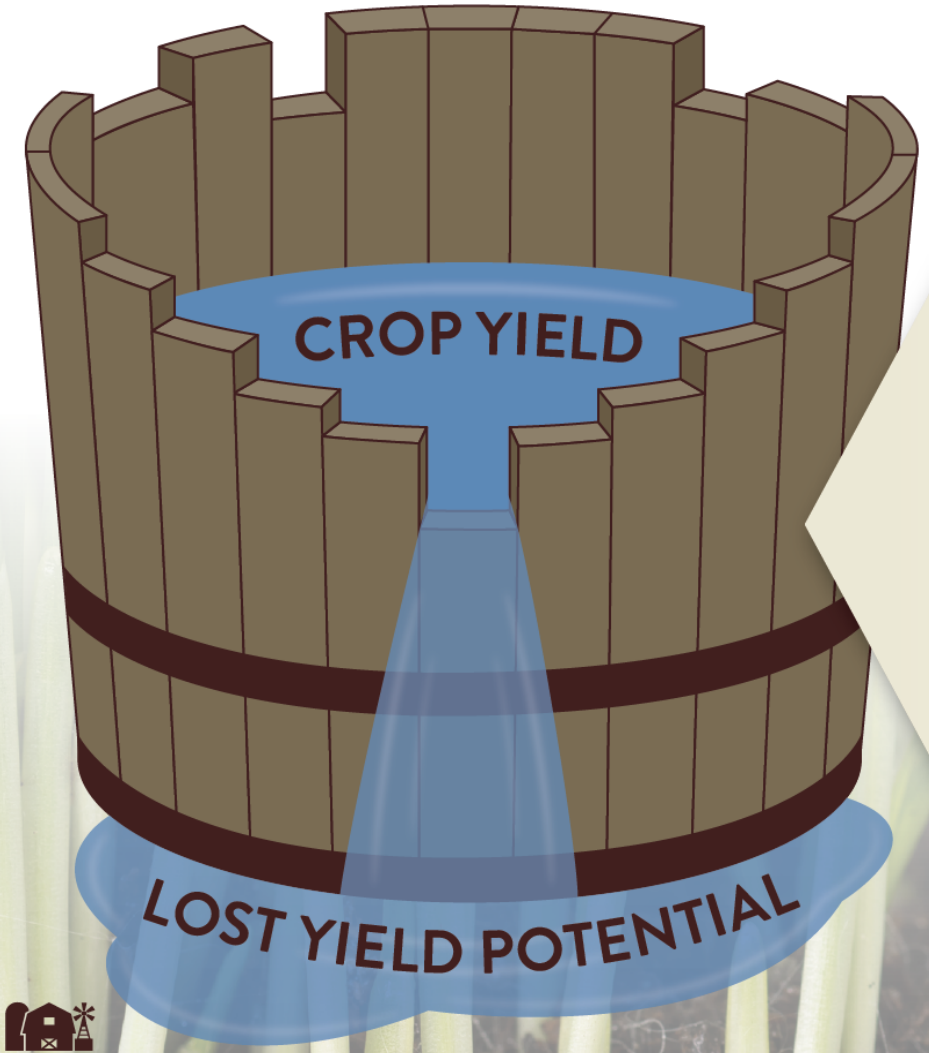
In our farming practices, we've strategically combined the Knapsack Algorithm and Liebig's Minimum Law to enhance efficiency and sustainability. Through this integrated approach, our goal is to achieve precision in resource utilization, minimizing waste while maximizing crop yield. Leveraging the computational efficiency of the Knapsack Algorithm and the strategic insights from Liebig's Minimum Law, we aim to make informed decisions that contribute to both increased agricultural productivity and environmentally conscious farming practices.

Liebig's Minimum Law

Justus von Liebig's Minimum Law, a fundamental principle in resource optimization, asserts that the growth or output of a system is determined by its scarcest resource. In other words, the limiting factor, or the resource in minimum supply, constrains overall performance. This concept, named after chemist Justus von Liebig, is crucial in various fields, emphasizing the importance of addressing and optimizing the most limiting resource to achieve maximum efficiency.

THE LAW OF THE MINIMUM: *Liebig's Barrel Metaphor*

A plant's potential is limited if even 1 growth factor is deficient or missing



BARREL STAVES = Individual growth factors that plants need

PLANTS NEED THE CORRECT:

- Nutrition
- Light
- Water/ Humidity
- Soil Conditions
- Temperature

Macro-nutrients

7 N NITROGEN	19 K POTASSIUM	16 S SULFUR
15 P PHOSPHORUS	20 Ca CALCIUM	12 Mg MAGNESIUM

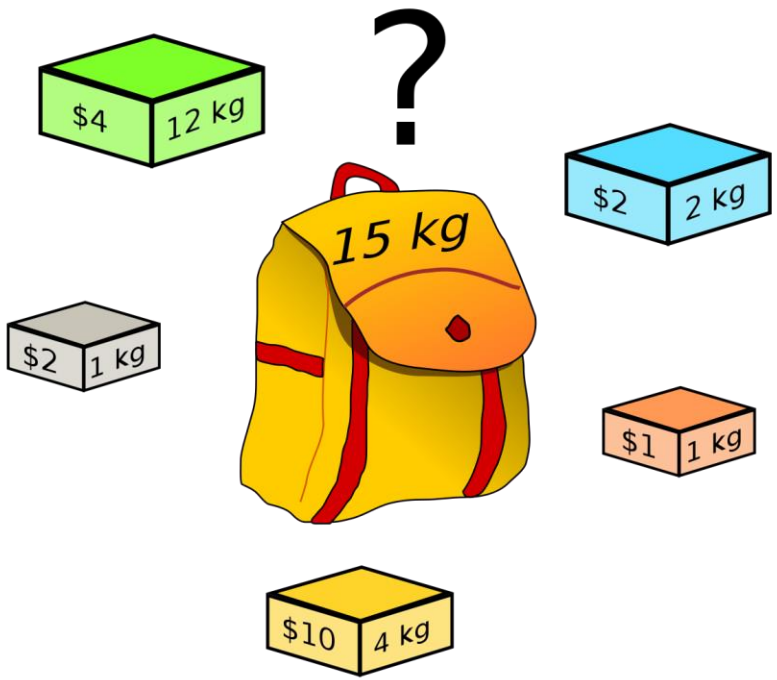
Micro-nutrients

17 Cl CHLORINE	5 B BORON	25 Mn MANGANESE	28 Ni NICKEL
26 Fe IRON	29 Cu COPPER	30 Zn ZINC	42 Mo MOLYBDENUM

REDMOND

Knapsack Algorithm

The knapsack algorithm is a dynamic programming tool used for optimizing resource allocation, specifically when items have values and weights, and there's a constraint on the total weight that can be carried. It's widely applied in scenarios like resource management.



Connection Between Knapsack and Liebig

The connection between the Knapsack Algorithm and Liebig's Minimum Law lies in their shared emphasis on optimizing resource utilization. The Knapsack Algorithm, a dynamic programming technique, addresses computational problems by efficiently allocating items with weights and values within specified constraints. In parallel, Liebig's Minimum Law asserts that the growth or output of a system is constrained by its scarcest resource, urging a focus on optimizing limiting factors for optimal overall performance.

SHADERS

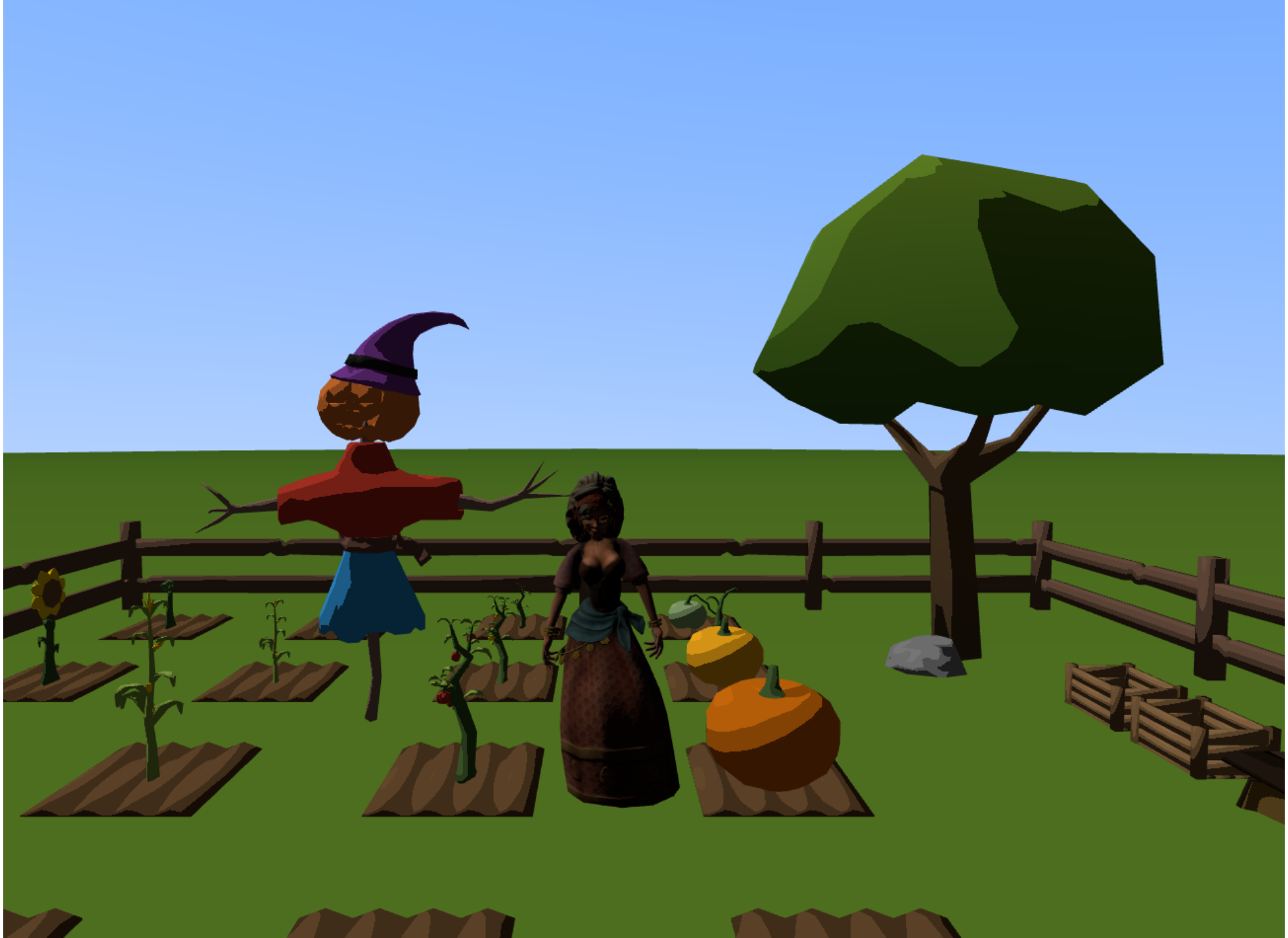
- Phong Shading
- Toon Shading

```
const PhongVertexShader =`

    out vec3 v_position;
    out vec3 v_normal;
    out vec2 v_texCoord;
    out vec3 v_surfaceToLight;
    uniform vec3 spotLightPosition;

    void main() {

        v_position = position;
        v_normal = normalize((transpose(inverse(modelMatrix)) * vec4(normal, 1.0)).xyz);
        v_texCoord = uv;
        vec3 pos = (modelMatrix * vec4(position, 1.0)).xyz;

        v_surfaceToLight = spotLightPosition - pos;

        gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
    }
`
```

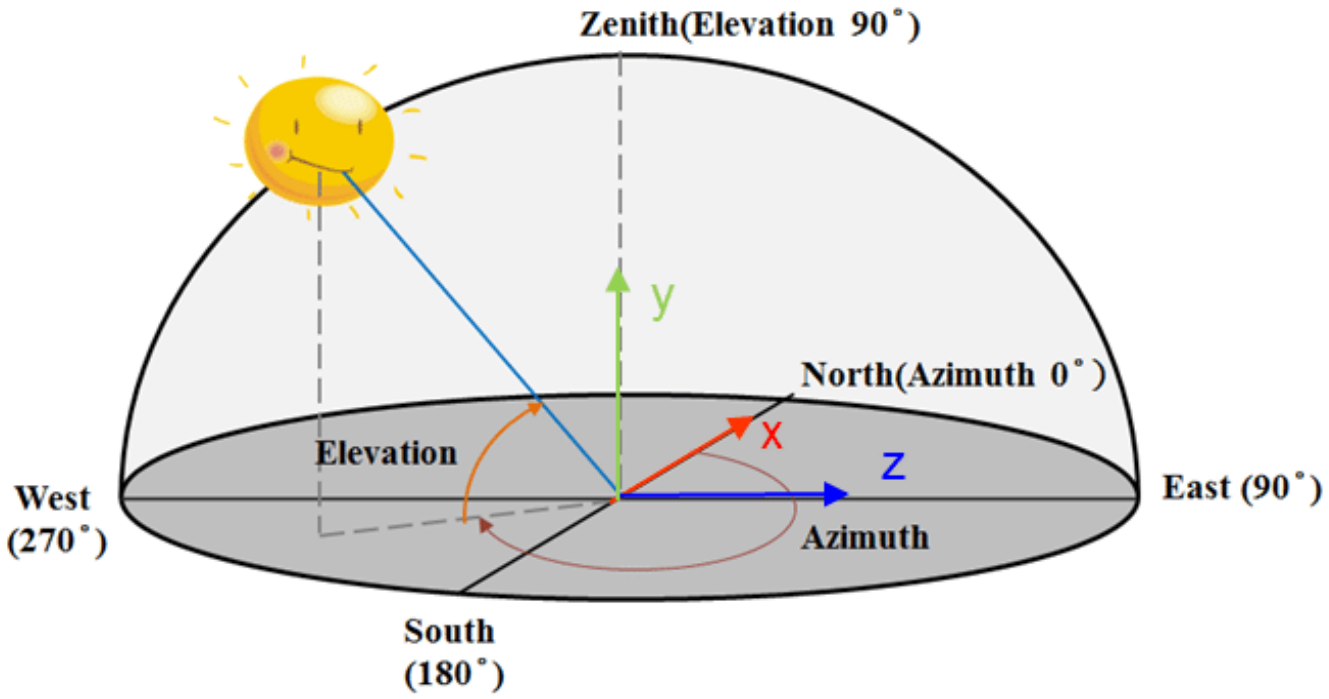
```
const PhongFragmentShader =`
    ...
    void main() {

        vec3 light_direction1 = normalize(light1_position - v_position);


        vec3 surfaceToLightDirection = normalize(v_surfaceToLight);
        float df2 = 0.0;
        float dotFromDirection = dot(surfaceToLightDirection,-u_lightDirection);
        if (dotFromDirection >= 0.9) {
            df2 = dot(v_normal, surfaceToLightDirection)*2.0*spotLightSwitch;
          }


        // Calculate the light diffusion factor for both lights
        float df1 = diffuseFactor(v_normal, light_direction1);

        // Combine diffuse contributions from both lights
        float totalDiffuse = df1+df2;
        // Sample texture
        vec3 textureColor = texture2D(u_texture, v_texCoord).xyz;

        // Calculate the final surface color
        vec3 surface_color = totalDiffuse * textureColor * light1_intensity;

        // Fragment shader output
        gl_FragColor = vec4(surface_color, 1.0);
        #include <colorspace_fragment>
    }
`
```

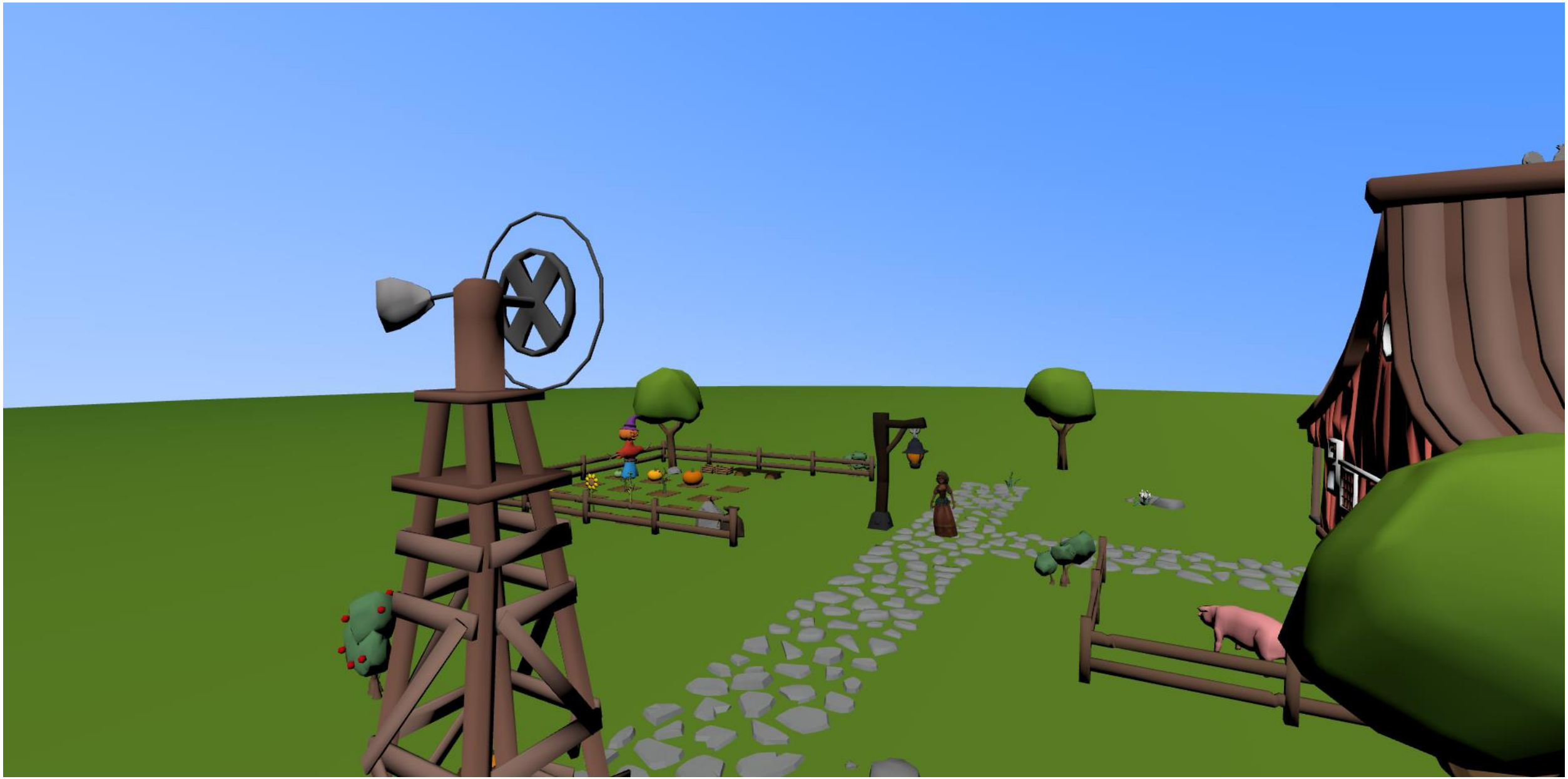
```
const toonVertexShader =`

out vec3 v_position;
out vec3 v_normal;
out vec2 v_texCoord;
out vec3 v_surfaceToLight;
uniform vec3 spotLightPosition;

void main() {
    // Save the varyings
    v_position = position;
    v_normal = normalize((transpose(inverse(modelMatrix)) * vec4(normal, 1.0)).xyz);
    v_texCoord = uv;

    vec3 pos = (modelMatrix * vec4(position, 1.0)).xyz;

    v_surfaceToLight = spotLightPosition - pos;

    // Vertex shader output
    gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
                }

`
```

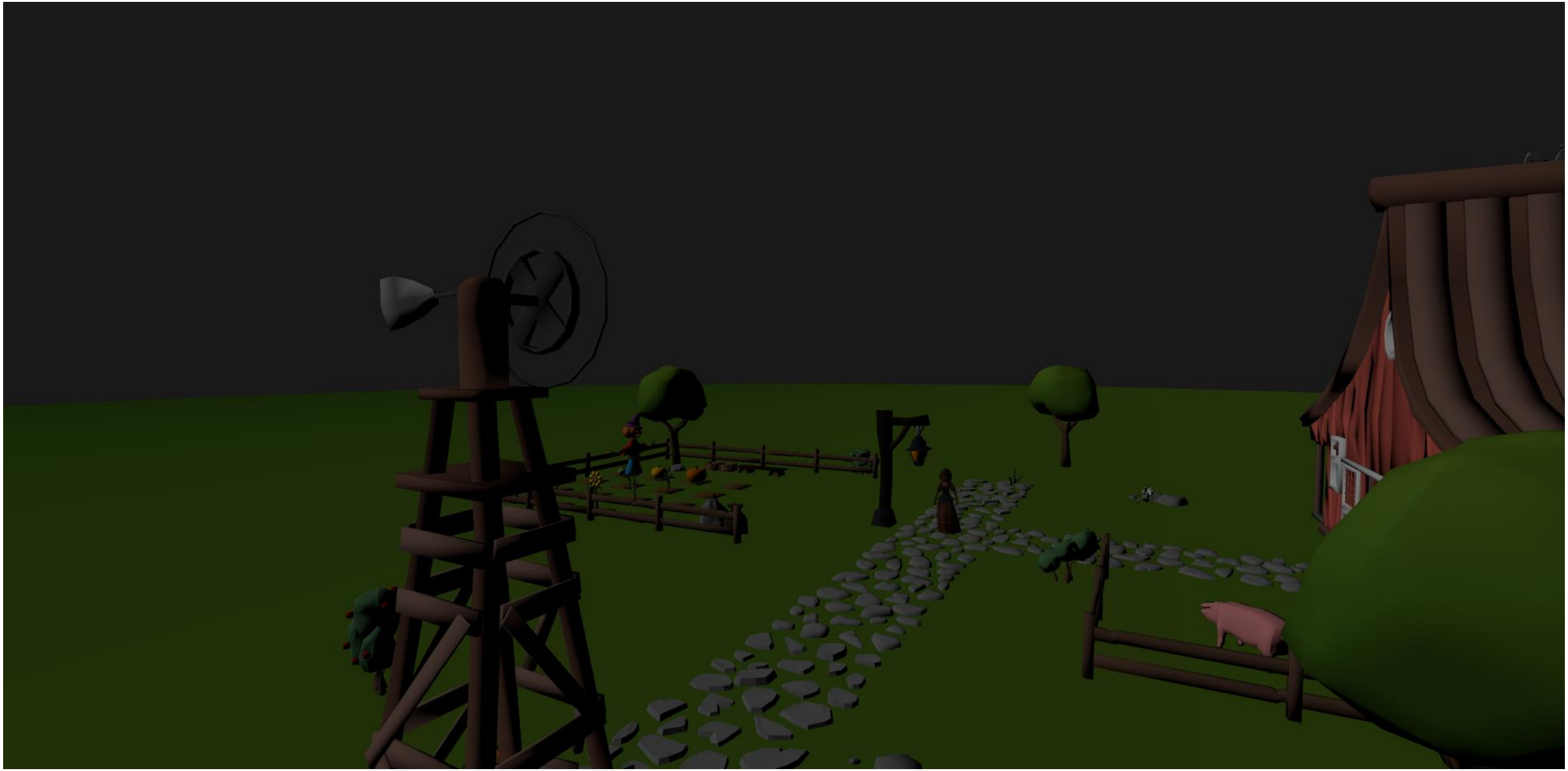
```
const toonFragmentShader =`
    ...
    void main() {

        vec3 light_direction1 = normalize(light1_position - v_position);
        float df1 = diffuseFactor(v_normal, light_direction1);
        vec3 surfaceToLightDirection = normalize(v_surfaceToLight);
        float df2 = 0.0;
        float dotFromDirection = dot(surfaceToLightDirection,-u_lightDirection);
        if (dotFromDirection >= 0.9) {
            df2 = (dot(v_normal, surfaceToLightDirection)+0.3)*spotLightSwitch;
          }

        float totalDiffuse = df1;

        // Sample texture
        vec3 textureColor = texture2D(u_texture, v_texCoord).xyz;

        // Define the toon shading steps based on the total diffuse

        float step;
        if (totalDiffuse > 0.95)
            step = 0.8;
        else if (totalDiffuse > 0.66)
            step = 0.66;
        else if (totalDiffuse > 0.33)
            step = 0.33;
        else
            step = 0.05;

        step += df2;
        vec3 surface_color = step * textureColor * light1_intensity;

        // Fragment shader output
        gl_FragColor = vec4(surface_color, 1.0);
        #include <colorspace_fragment>
    }
`
```



Implementation of Day and Night Cycle







Translate mode

Scale mode

Rotate mode

Project Team

Alperen Akça 2200356035
Murat Günay 2200356040
Murat Eren Aytekin 2200356026
Yusuf Karataş 2200356022

Thank you!