OpenGLHF

Tinkering with the Minecraft Entity Rendering

Noah Beshiri, Emanuel Mairoll, Amelie Zähringer

Agenda

- Project description and overview
- First steps in Minecraft and OpenGL
- Custom OpenGL: Bounding Boxes and Tracers
- Minecraft Rendering System: Entity information
- Live Demo

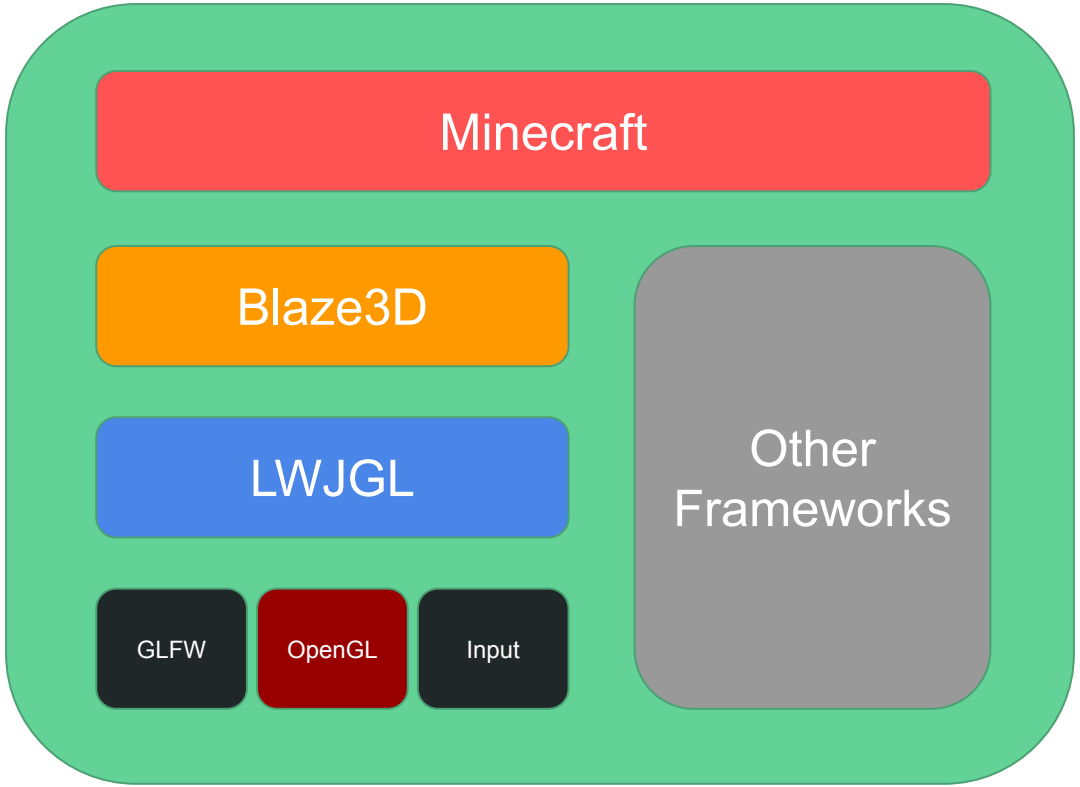
Project description and overview

- Implement a Minecraft mod
- Tinker around with OpenGL
- Helps us to defeat other players
- Goals: Bounding Boxes, Tracers, Entity information

Honorable Mention: *LearnOpenGL by Joey de Vries* (https://learnopengl.com/)

What is Minecraft? One slide introduction

Minecraft “Tech Stack”



Modding in Minecraft

- ModCoderPack → Modifies Minecraft sources directly
- Forge
- Fabric (Light)

} API Hooks

Modding in Minecraft

- Many different modding frameworks
- ModCoderPack: “dinosaur”, modifies Minecraft sources directly
- Forge: Extensive API for mod creation and compatibility
- Fabric: Lightweight, quicker update cycles than Forge
- Allow customisation by “hooking” into different parts of the game lifecycle
- For the sake of simplicity we use Fabric here

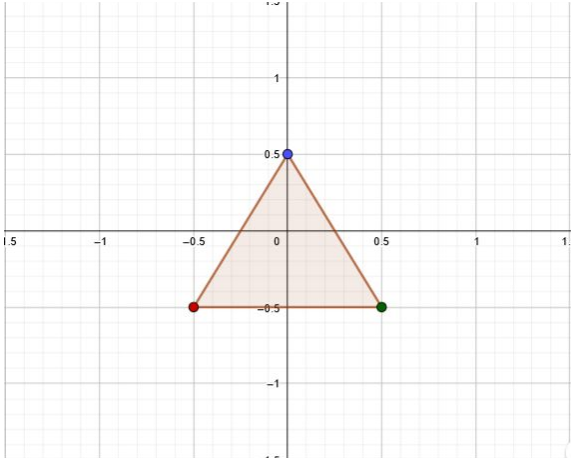
Modding in Minecraft

```java
import org.lwjgl.opengl.GL33;

public class OpenGLHFClient implements ClientModInitializer {

    @Override
    public void onInitializeClient() {
        // entry point
        // hook into minecraft update-render-loop
        WorldRenderEvents.AFTER_ENTITIES.register(this::myCustomRenderCode);
    }

    private void myCustomRenderCode(WorldRenderContext worldRenderContext) {
        // access underlying OpenGL context
        // using LWJGL GL33.gl* functions, e.g.
        GL33.glClearColor(0.2f, 0.2f, 0.3f, 1.0f);
    }
}
```

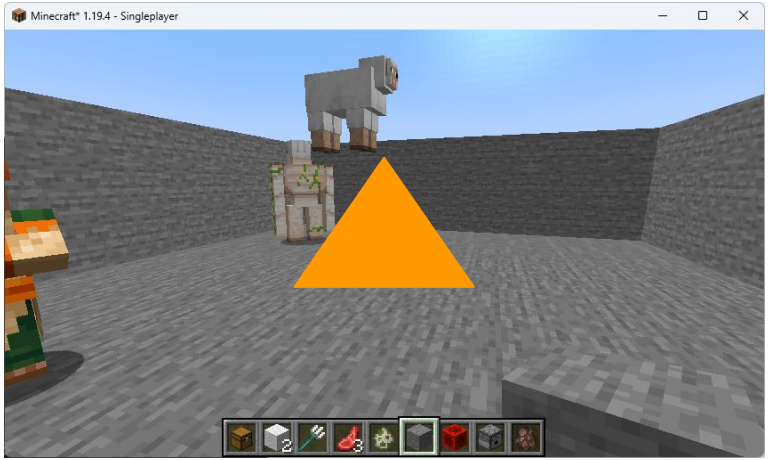
Writing custom OpenGL

First steps in Minecraft and OpenGL

Rendering a triangle:



Triangle we want to draw
(NDC / Screen Space)



Expectation

First steps in Minecraft and OpenGL

Rendering a triangle:

```
// ... setup VAO and VBO

float[] vertices = {
     0.0f,  0.5f, 0.0f,
     0.5f, -0.5f, 0.0f,
    -0.5f, -0.5f, 0.0f
};

// ... fill VBO, prepare shaders

GL33.glDrawArrays(GL33.GL_TRIANGLES, 0, 3);
```

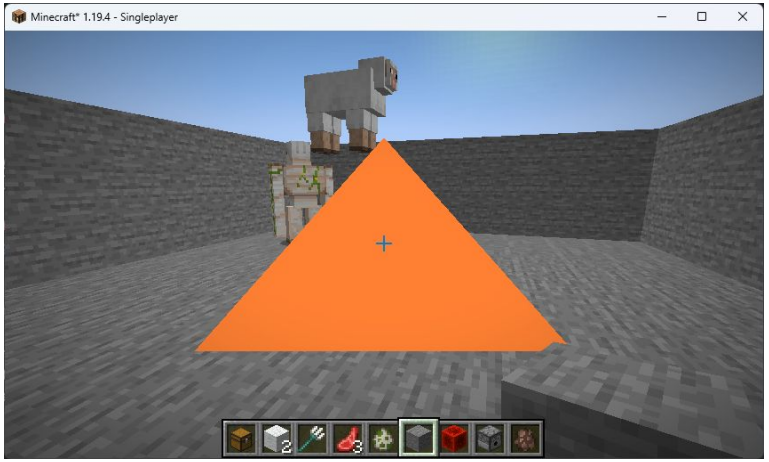
No triangle!?

➜ Backface Culling

First steps in Minecraft and OpenGL

Rendering a triangle:

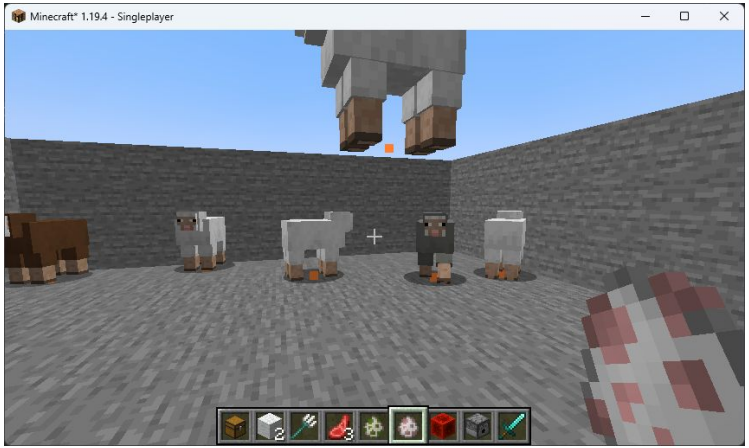
```
// ... setup VAO and VBO

float[] vertices = {
    -0.5f, -0.5f, 0.0f,
     0.5f, -0.5f, 0.0f,
     0.0f,  0.5f, 0.0f,
};

// ... fill VBO, prepare shaders

GL33.glDrawArrays(GL33.GL_TRIANGLES, 0, 3);
```



Success!

Entity Position Prototype

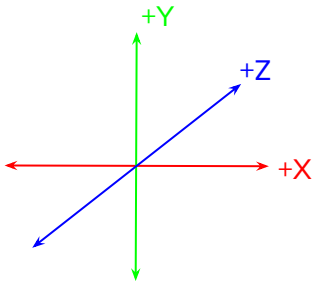
```
// ... create buffers

Gl33.glVertexAttribPointer(0, 3, GL33.GL_FLOAT, false, 0, 0);
GL33.glEnableVertexAttribArray(0);

// ... fill VBO with entity position vertices (x, y, z)

GL33.glPointSize(10);
Gl33.glDrawArrays(GL33.GL_POINTS, 0, vertices.length / 3);
```



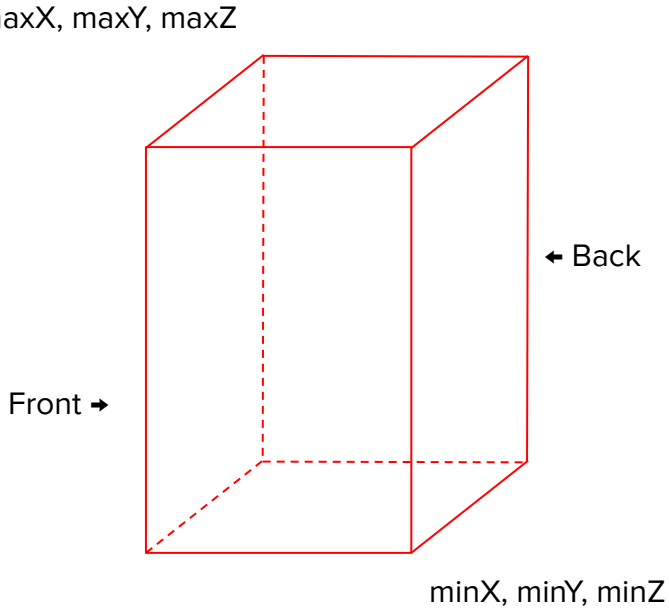
Bounding Boxes

Idea / Goal

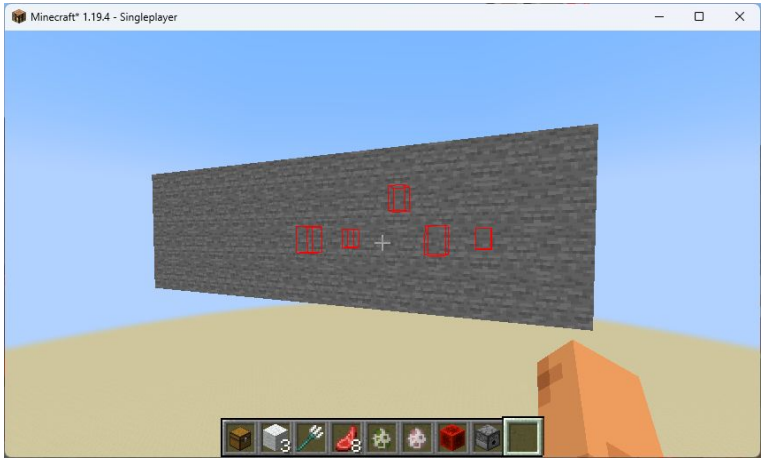
Axis Aligned Bounding Boxes

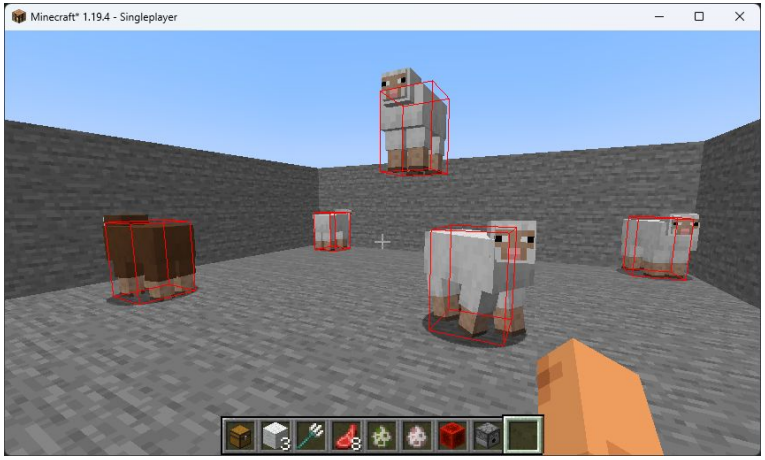


Minecraft World
Reference Frame



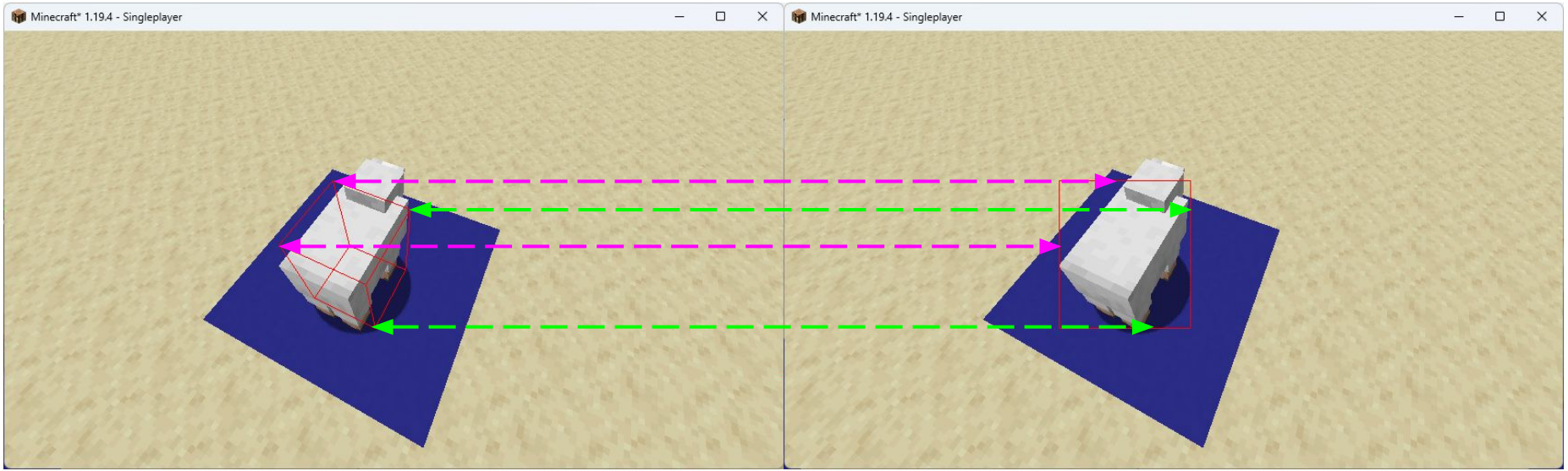
Bounding Boxes





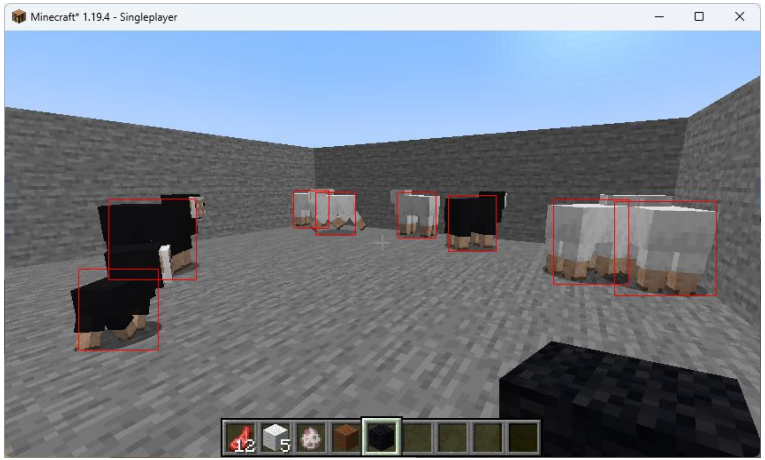
Implemented 3D Bounding Boxes

Bounding Boxes



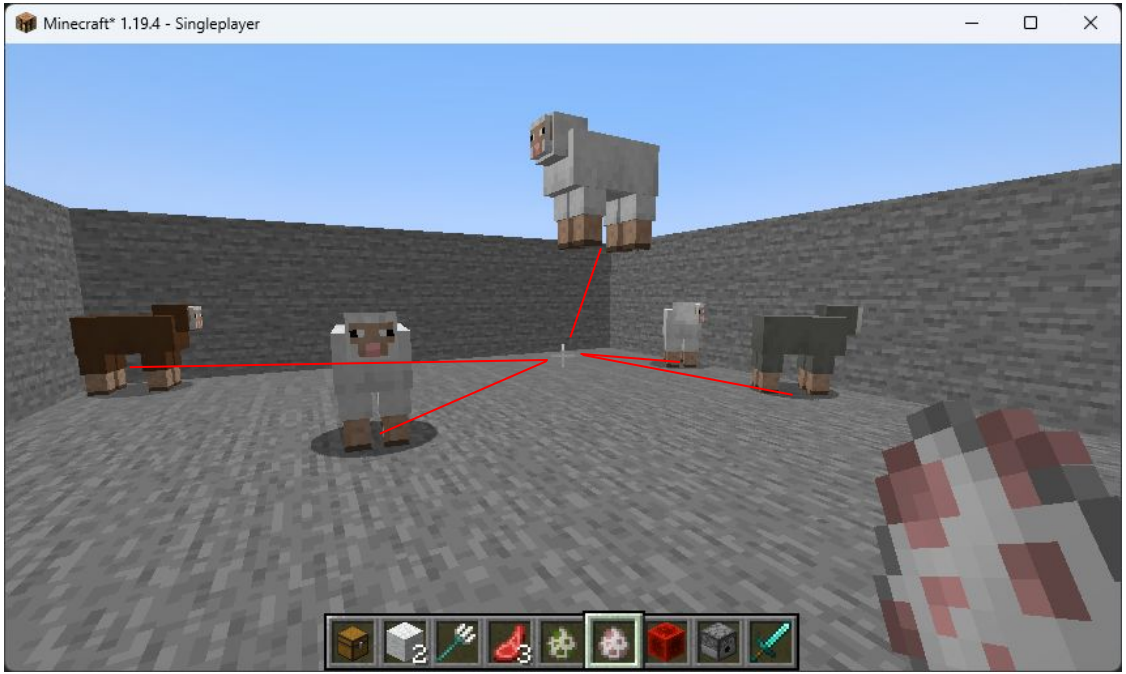
Convert 3D Bounding Box to 2D Plane (Convex Hull)

Bounding Boxes



Implemented 2D Bounding Boxes

Tracers

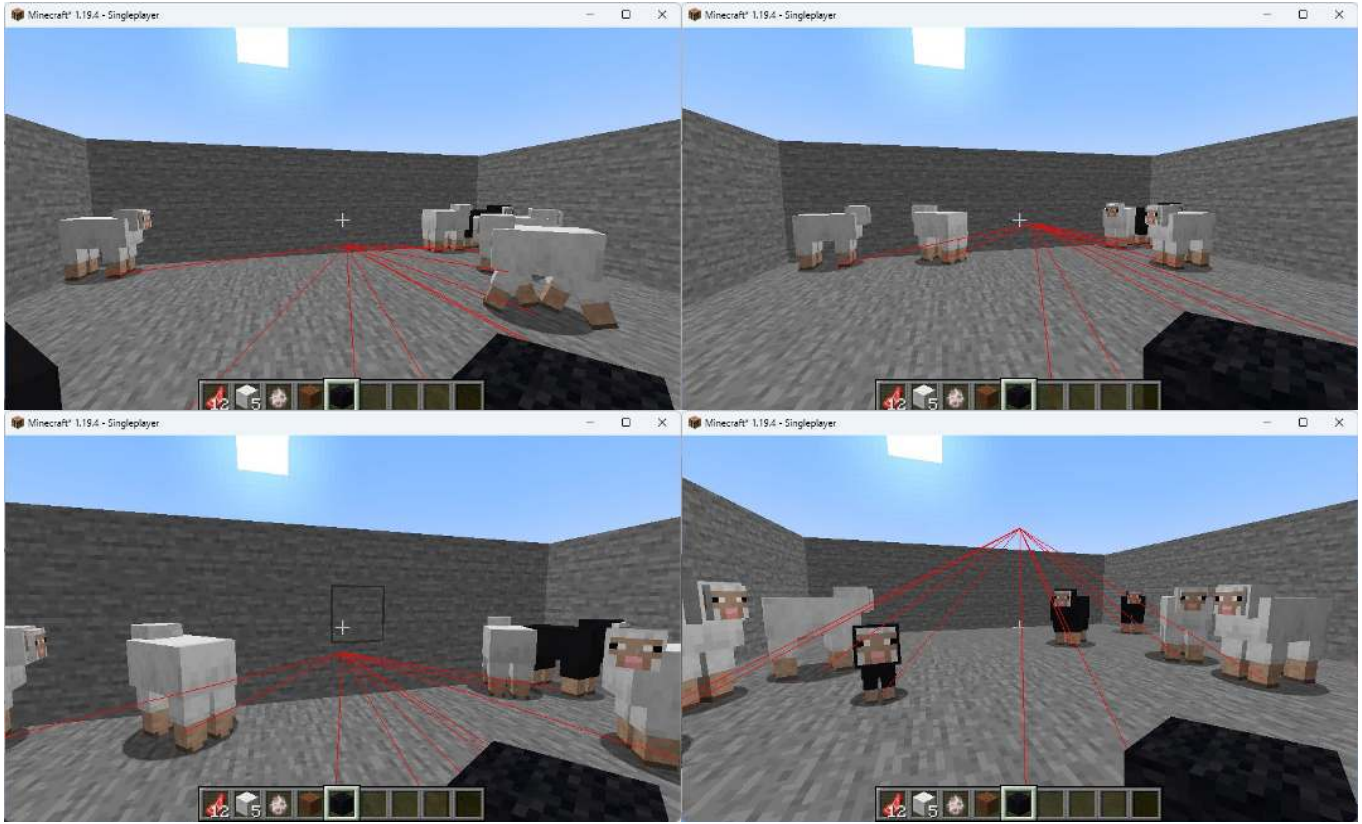


Idea / Goal

Tracers “Naive” Implementation

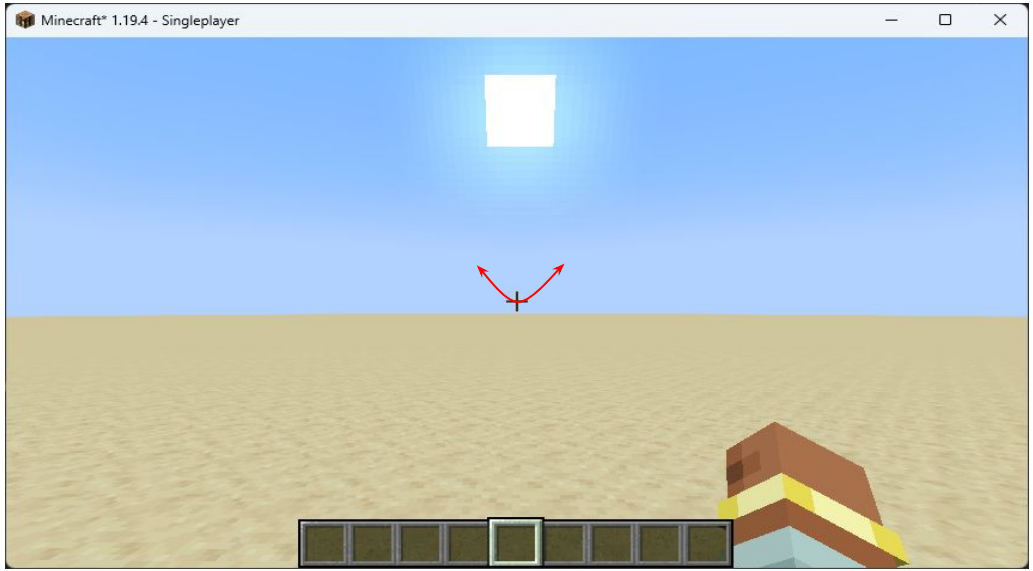
```
@Override
protected DoubleStream toVertices(Entity entity) {
    var player = MinecraftClient.getInstance().player;

    var crosshairX = player.getX() + player.getRotationVector().x;
    var crosshairY = player.getY() + player.getEyeHeight(player.getPose())
                   + player.getRotationVector().y;

    var crosshairZ = player.getZ() + player.getRotationVector().z;

    return DoubleStream.of(
            entity.getX(), entity.getY(), entity.getZ(),
            crosshairX, crosshairY, crosshairZ
    );
}
```

Tracers “Naive” Implementation



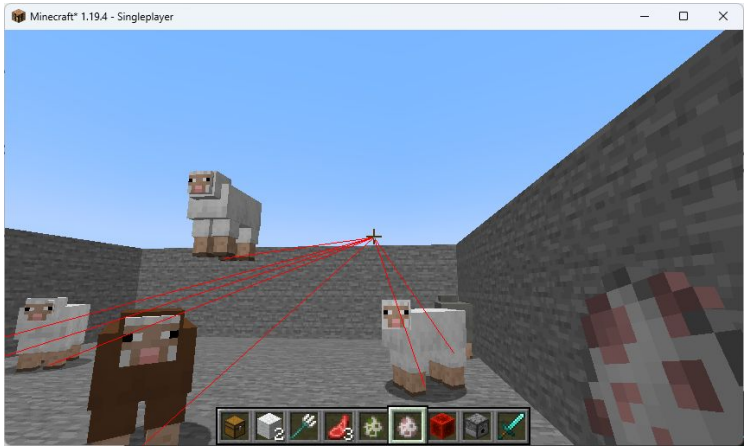
View Bobbing

- Simulates “head shaking” while walking



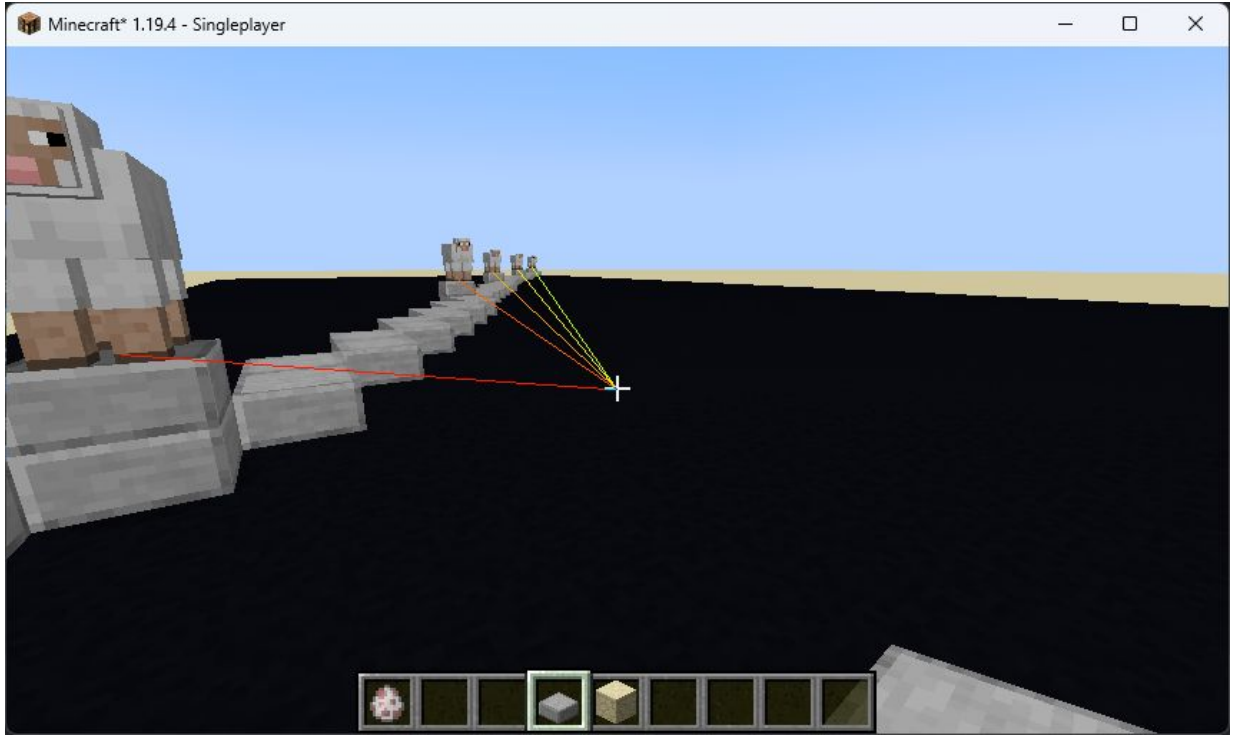
Tracers

```
layout (points) in;
layout (line_strip, max_vertices = 2) out;

uniform mat4 ModelViewMat;
uniform mat4 ProjMat;

void main() {
    gl_Position = ProjMat * ModelViewMat * gl_in[0].gl_Position;
    EmitVertex();
    gl_Position = vec4(0, 0, 0, 1);
    EmitVertex();
}
```



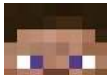
Implemented Tracers

Tracers



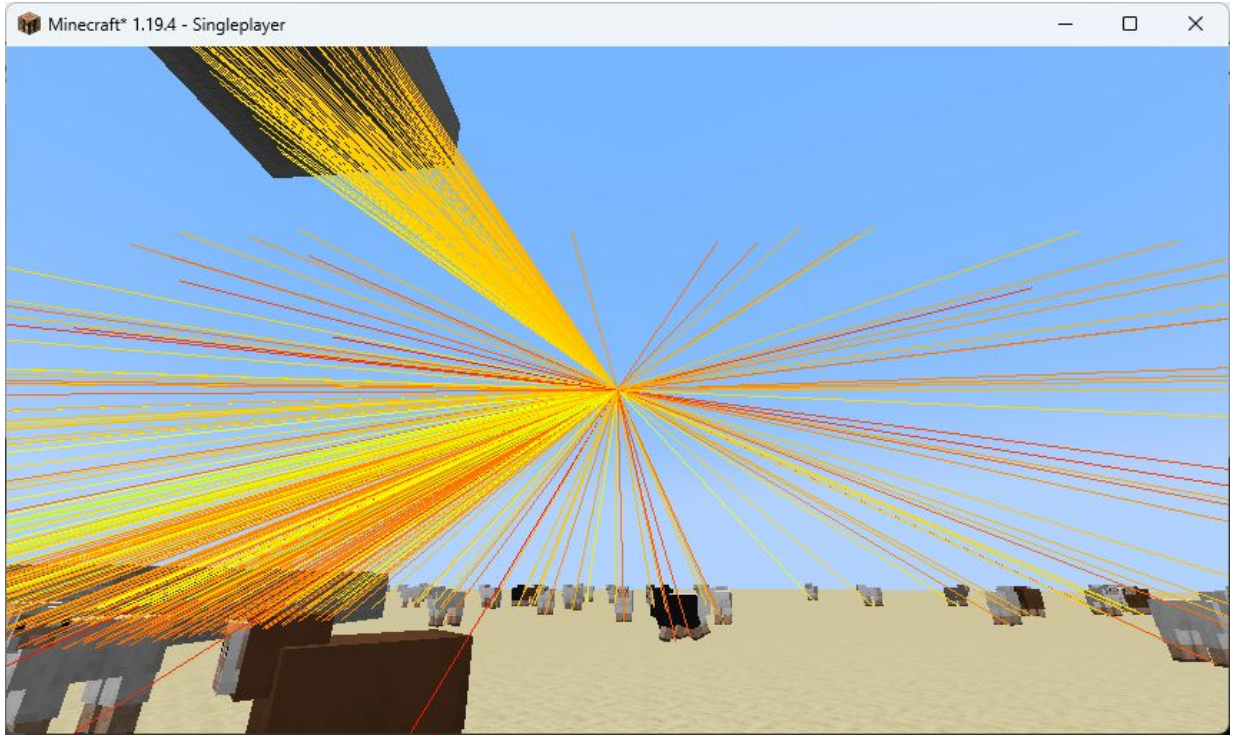
Tracers change color based on distance

Tracers



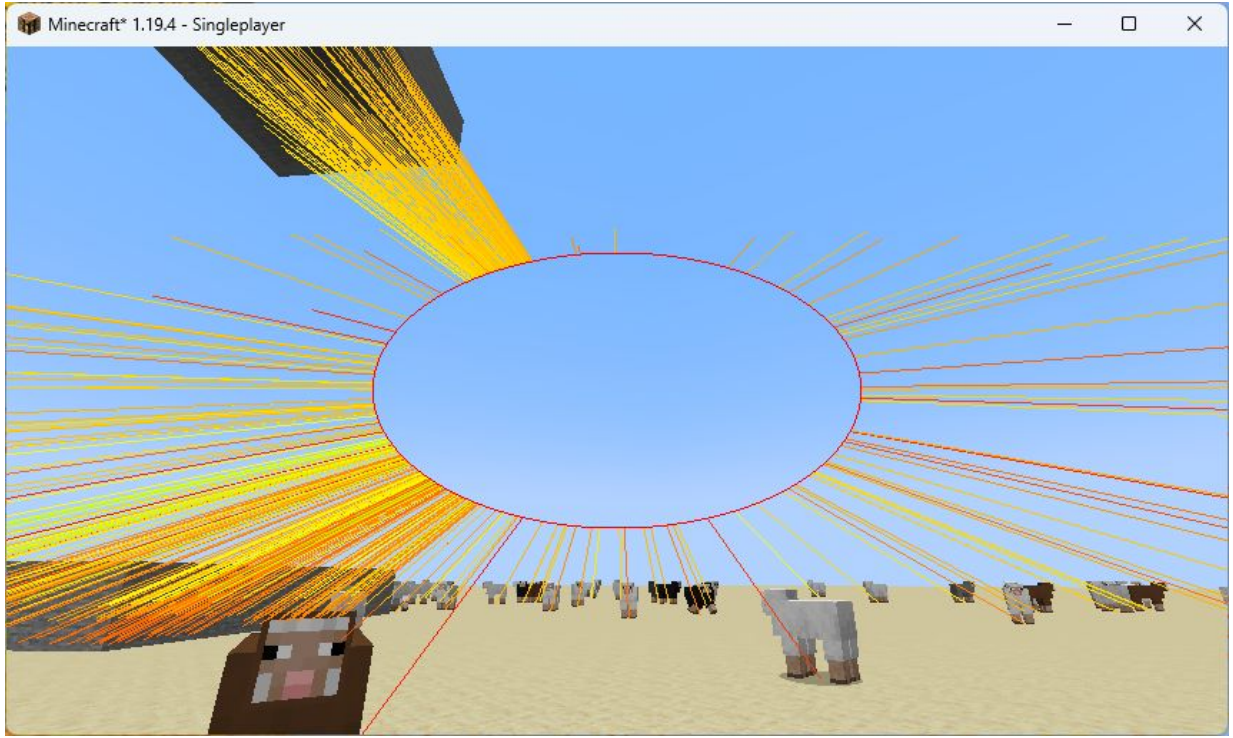
```
out vec4 FragColor;

in float distanceGeom;

uniform float maxDistance;

void main()
{

    float distanceClamped = clamp(maxDistance, 0.0, distanceGeom);
    float x = distanceClamped / maxDistance;

    float r = 2.0 * (1.0 - x);
    float g = 2.0 * x;

    FragColor = vec4(r, g, 0.0, 1.0);

}
```

Tracers



Too many entities are very bothersome

Tracers



Solution: Ellipse Cutout

Tracers

```
out vec4 FragColor;

uniform float majorAxis;
uniform float minorAxis;
uniform ivec2 viewportDimensions;

bool isInEllipse(vec2 point)
{

    float rx = majorAxis / 2;
    float ry = minorAxis / 2;

    float left = pow(point.x - 0.5, 2.0) * pow(ry, 2.0) + pow(point.y - 0.5, 2.0) * pow(rx, 2.0);
    float right = pow(rx, 2.0) * pow(ry, 2.0);
    return left <= right;
}

void main()
{

    vec2 FragCoord = gl_FragCoord.xy / viewportDimensions;

    if (isInEllipse(FragCoord)) discard;

    // ... tracer color change based on distance code (r, g)

    FragColor = vec4(r, g, 0.0, 1.0);

}
```

Using Blaze3D

One slide of theory about Blaze3D

Mojang's proprietary 3D graphics engine (intr. 2019 in 1.15)

Order of rendering:

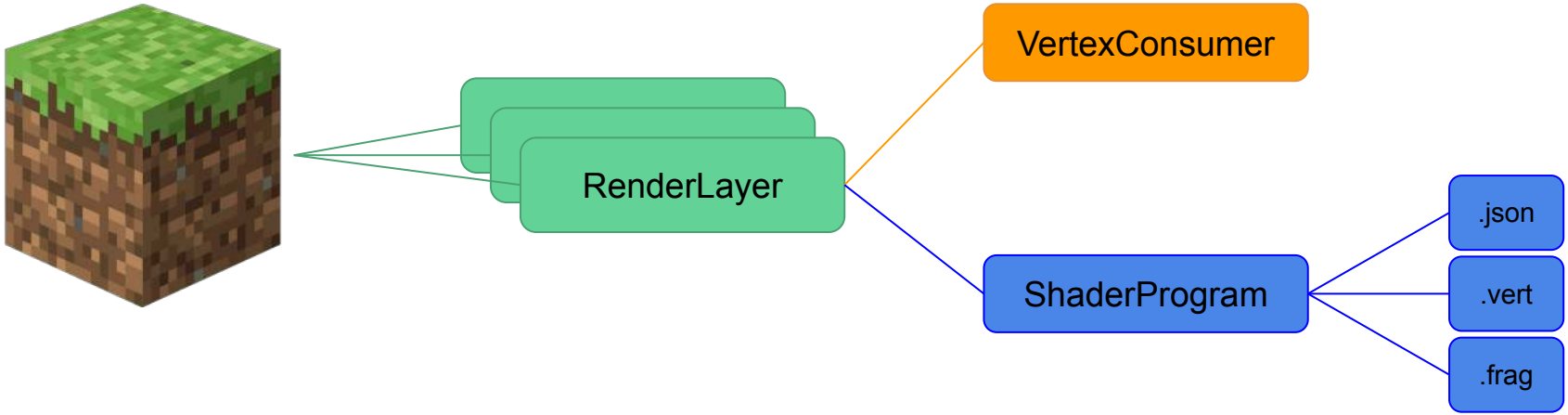
- Opaque, full blocks are rendered first
- Entities and non-full blocks follow
- Transparent objects are rendered in back-to-front order
- The HUD and GUI overlays are rendered last

Minecraft Blaze Entity

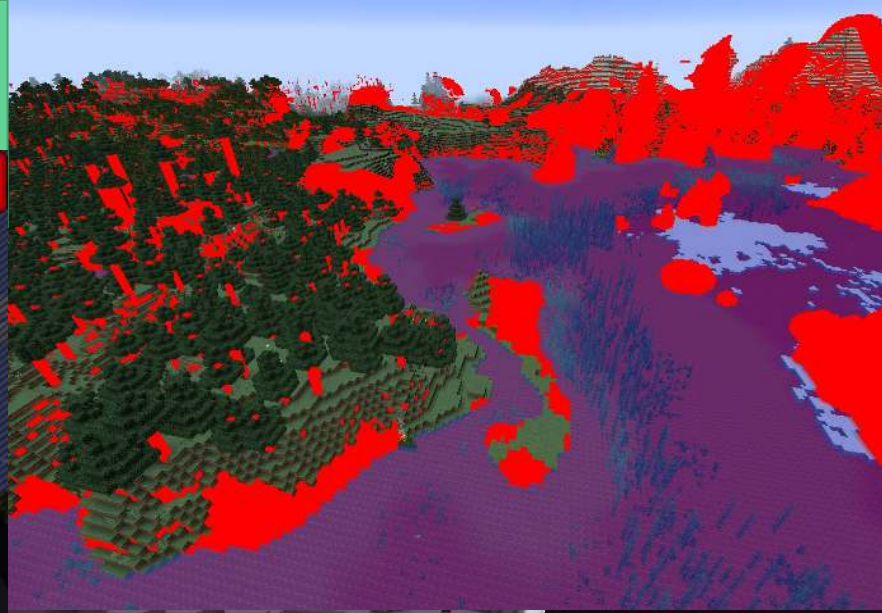
Ok... two slides

Core Concepts of Minecraft-Rendering

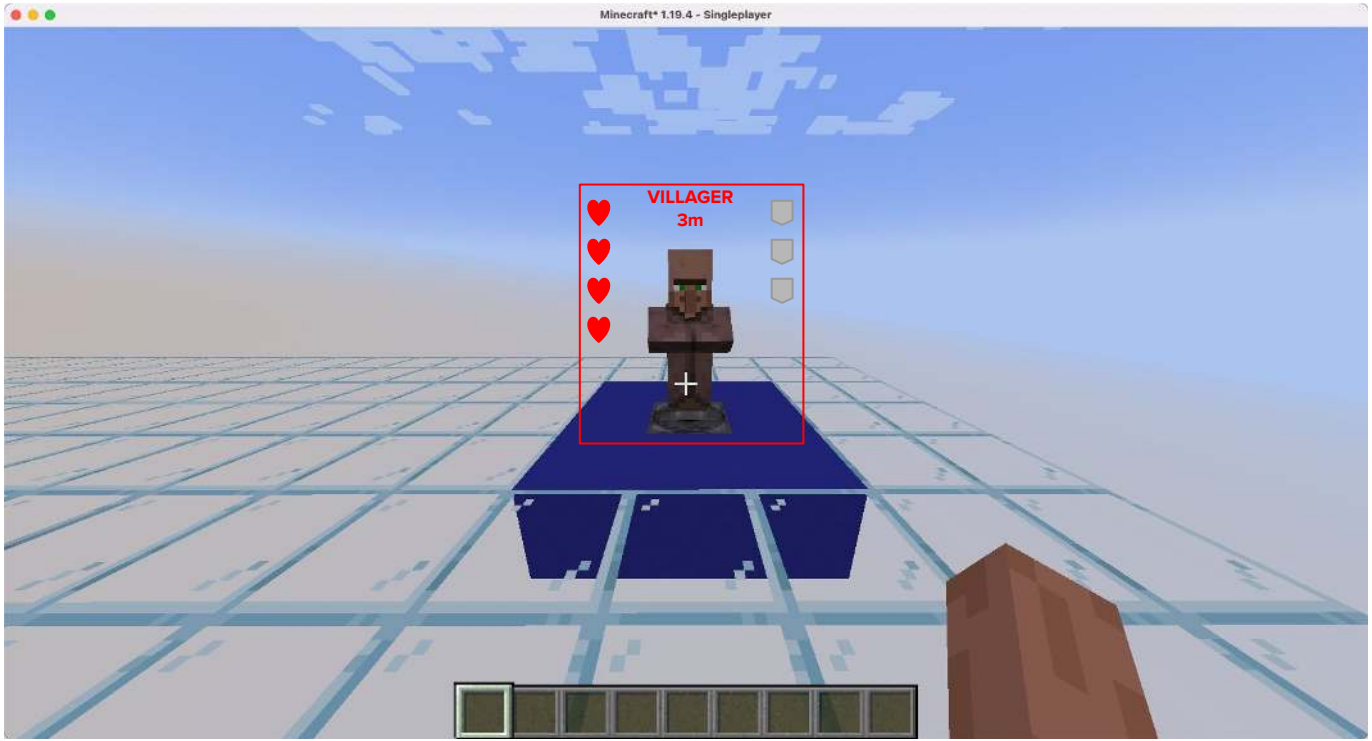
- **RenderLayer**: Groups elements for rendering based on characteristics.
- **VertexConsumer**: Interface for sending vertices data to GPU.
- **ShaderProgram**: Controlling vertex and fragment shaders.



Core Shader Examples



Entity Information Panel



Idea / Goal

Rendering the backplate

```
private static void renderBackplate(Rect2i dim, VertexConsumerProvider vertexConsumers, Matrix4f matrix4f, int light) {
    float rectAlpha = MinecraftClient.getInstance().options.getTextBackgroundOpacity(0.25F);

    VertexConsumer vertexConsumer = vertexConsumers.getBuffer(RenderLayer.getTextBackground());

    var rectMinX = dim.getX();
    var rectMaxX = dim.getX() + dim.getWidth();
    var rectMinY = dim.getY();
    var rectMaxY = dim.getY() + dim.getHeight();

    vertexConsumer.vertex(matrix4f, rectMinX, rectMaxY, 0.01f).color(0, 0, 0, rectAlpha).light(light).next();
    vertexConsumer.vertex(matrix4f, rectMaxX, rectMaxY, 0.01f).color(0, 0, 0, rectAlpha).light(light).next();
    vertexConsumer.vertex(matrix4f, rectMaxX, rectMinY, 0.01f).color(0, 0, 0, rectAlpha).light(light).next();
    vertexConsumer.vertex(matrix4f, rectMinX, rectMinY, 0.01f).color(0, 0, 0, rectAlpha).light(light).next();
}
```

Entity Information Panel

Backplate done!

Text Rendering

```java
private static void renderText(Text text, int line, Rect2i dim, VertexConsumerProvider vertexConsumers,
                               Matrix4f matrix4f, int light) {
    TextRenderer textRenderer = MinecraftClient.getInstance().textRenderer;
    float x = (float) textRenderer.getWidth(text) / 2;
    textRenderer.draw(text, -x, dim.getY() + 3 + (line * 10), /* ... other arguments ...*/);
}


// later within the Minecraft TextRenderer and GlyphRenderer
void drawGlyph(GlyphRenderer glyphRenderer, boolean bold, boolean italic, float weight, float x, float y,
               Matrix4f matrix, float red, float green, float blue, float alpha, int light) {
    VertexConsumer vertexConsumer = this.vertexConsumers.getBuffer(glyphRenderer.getLayer(this.layerType));

    // ... modify alignment constraints

    vertexConsumer.vertex(matrix, f + m, k, 0.0F).color(red, green, blue, alpha).texture(this.minU, this.minV).light(light).next();
    vertexConsumer.vertex(matrix, f + n, l, 0.0F).color(red, green, blue, alpha).texture(this.minU, this.maxV).light(light).next();
    vertexConsumer.vertex(matrix, g + n, l, 0.0F).color(red, green, blue, alpha).texture(this.maxU, this.maxV).light(light).next();
    vertexConsumer.vertex(matrix, g + m, k, 0.0F).color(red, green, blue, alpha).texture(this.maxU, this.minV).light(light).next();
}
```

Minecraft Default Font PNG

Entity Information Panel

Text Rendering also done!

Custom Texture Rendering

```
private static void drawTexture(MatrixStack matrices, int x, int y, float z, float u, float v,
                                int width, int height, int textureWidth, int textureHeight) {
    int xMax = x + width;
    int yMax = y + height;

    float uMin = u / (float) textureWidth;
    float uMax = (u + (float) width) / (float) textureWidth;

    float vMin = (v + 0.0F) / (float) textureHeight;
    float vMax = (v + (float) height) / (float) textureHeight;

    Matrix4f matrix = matrices.peek().getPositionMatrix();
    RenderSystem.setShader(GameRenderer::getPositionTexProgram);
    BufferBuilder bufferBuilder = Tessellator.getInstance().getBuffer();
    bufferBuilder.begin(VertexFormat.DrawMode.QUADS, VertexFormats.POSITION_TEXTURE);
    bufferBuilder.vertex(matrix, (float) x, (float) y, z).texture(uMin, vMin).next();
    bufferBuilder.vertex(matrix, (float) x, (float) yMax, z).texture(uMin, vMax).next();
    bufferBuilder.vertex(matrix, (float) xMax, (float) yMax, z).texture(uMax, vMax).next();
    bufferBuilder.vertex(matrix, (float) xMax, (float) y, z).texture(uMax, vMin).next();
    BufferRenderer.drawWithGlobalProgram(bufferBuilder.end());
}
```

Entity Information Panel

Finished Implementation

LIVE DEMO

Thank you for
your attention!

BONUS: Bloopers and Outtakes