









Project #3













In this project, you need to implement an interactive chopper that can shoot (glowing) bullets. Also, a terrain should be rendered based on a texture image.

1 Requirements

Controlling the chooper The chopper needs to be controlled by the keyboard as follows.

- : Rotate left
- : Rotate right
- : Move forward
- : Move backward
-  or : Move upward
-  or : Move downward

Controlling the view The view (camera) needs to be controlled as follows. Refer to the demo video.

-  +  /  + : Rotate the view around the z-axis of the world coordinates system.
-  +  /  + : Tilt down/up the whole world.
-  or : Zoom in
-  or : Zoom out
- Note that the z-axis is always pointing upward in the screen.

The terrain Render the terrain based on the image attached.

- On the host side, you need to generate an $N \times N$ quad grid with each quad is split into two triangles. ([reference figure](#))
- Each vertex of the grid should have ONLY ONE attribute, the texture coordinates distributed evenly in $[0, 1] \times [0, 1]$. You SHOULD NOT use position/normal attributes. They should be computed in the vertex shader based on the texture. (Will be explained soon.).

The minimum number of texture units in vertex shaders is 16. The actual number can be checked by calling `gl.getParameter(gl.MAX_VERTEX_TEXTURE_IMAGE_UNITS)`. Or, you can go to [WebGL Report](#) and check the value of “Max Vertex Texture Image Units” in the “Vertex Shader” box.

- You also need to generate an index buffer for indexed rendering.
- When rendering the terrain, you need to bind the texture generated from the attached image.
- In the vertex shader, based on the texture coordinate attribute, you need to compute x , y , z coordinates as follows.

- The x , y coordinates can be computed by simply scaling and translating (to center the terrain in the world) of the texture coordinates. For example, if we want the lengths of the terrain along the x - and y -axes be L_x and L_y , respectively, then

$$x(s, t) = L_x \cdot s - (L_x/2)$$

$$y(s, t) = L_y \cdot t - (L_y/2)$$

- The z coordinate needs to be obtained by fetching the texture. Scale it (by S_z) appropriately if needed.:

$$z(s, t) = S_z \cdot \text{texture}(s, t)$$

- The tricky part is **to compute the normal vector**. (reference: [Unit normal vector of a surface](#))
 1. The normal vector at a point is orthogonal to the tangent plane at that point. So we first need to find the tangent plane at a point on the surface.
 2. For a **smooth parametric surface**

$$f(s, t) = \begin{bmatrix} x(s, t) \\ y(s, t) \\ z(s, t) \end{bmatrix},$$

two tangent vectors at $f(s_0, t_0)$ can be computed as

$$\begin{cases} \frac{\partial f}{\partial s}(s_0, t_0) & \text{(along the parameter } s) \\ \frac{\partial f}{\partial t}(s_0, t_0) & \text{(along the parameter } t) \end{cases}$$

where

$$\frac{\partial f}{\partial s} := \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right) \quad \text{and} \quad \frac{\partial f}{\partial t} := \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t} \right).$$

In this example,

$$x(s, t) = L_x \cdot s - (L_x/2)$$

$$y(s, t) = L_y \cdot t - (L_y/2)$$

$$z(s, t) = S_z \cdot \text{texture}(s, t)$$

* Examples of smooth parametric surfaces:

- **Bézier surface** ([demo1](#), [demo2](#))
- **NURBS surface**: supported by Blender

3. Obviously, we can easily differentiate $x(s, t)$ and $y(s, t)$. But, we cannot differentiate $z(s, t)$ since the texture image is a discrete function. Instead we can approximate it with finite differences, e.g., **central difference**.:

$$\frac{dg}{du}(u) = \frac{g(u + \delta) - g(u - \delta)}{2\delta}$$

where δ is a small positive number. Note that, to make this work well, you need to set the `gl.TEXTURE_MIN_FILTER` texture (or sampler) parameter to `gl.LINEAR`.

4. Then we can obtain the normal vector by taking **cross product** of the two tangent vectors. (followed by normalization)

* **Note that GLSL provides the `cross` function.**

Lighting/Shading You need to apply lighting effects both for the chooper (body + rotor) and the terrain.

- Put a fixed (directional or point) light (with appropriate properties) in the scene.
- You can implement either Phong reflection model or Blinn-Phong reflection model.
- You can choose either Phong interpolation or Gouraud interpolation.
- You should compute lighting for all the (active) multiple lights in the scene. (including the glowing bullets below.)

Glowing bullets Your chopper has a weapon. If you press the `Spacebar`, a glowing bullet is shot forward.

- Each bullet acts as a point light source and the chopper and the terrain should be lighted accordingly.
- Each bullet traverses the scene being affected by the gravity. Let the initial position and velocity of the bullet is (x_0, y_0, z_0) and (v_{x0}, v_{y0}, v_{z0}) , respectively. If we ignore all the complicated air resistance and everything, our differential equations are

$$\begin{aligned}\frac{d^2x}{dt^2}(t) &= 0 \\ \frac{d^2y}{dt^2}(t) &= 0 \\ \frac{d^2z}{dt^2}(t) &= -g\end{aligned}$$

where g is the gravitational constant. (You can set g to any value you want to make the animation look plausible.) In other words, the velocity doesn't change along the x - and y -axes, but changes along the z -axis due to the gravity. To make the bullets animate, you can use a simple numerical method, e.g., the [explicit Euler method](#). (Details will be explained in the lecture.)

- To make things simple, limit the number of maximum active bullets in the scene. (9 in the demo video)
- The bullet should “die” if its z value is too low. (e.g., if $z < 0$)

2 NOTE

- Compress all the required files (including the `readme.txt` file) into one ZIP file and upload it.
- Again, please press the “Submit” button after uploading your file.
- For test, I will execute a local web server.
- **Please extend the app to make it look awesome!!!**