Computer Graphics with Clojure, LWJGL, and Fastmath

Macroexpand-Noj 2025

Jan Wedekind

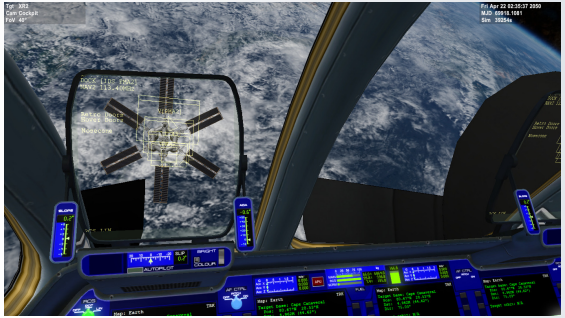
Saturday, October 18th 2025

Motivation

Develop a space flight simulator

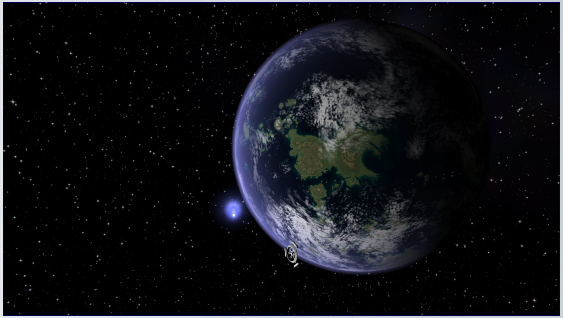
- Realistic Physics
- Free and Open Source Software
- Cross Platform
- High-Level Programming Language

Space Flight Simulator Examples 1/5



Open Orbiter Sim

former Orbiter 2016, open source since 2021, realistic, many mods



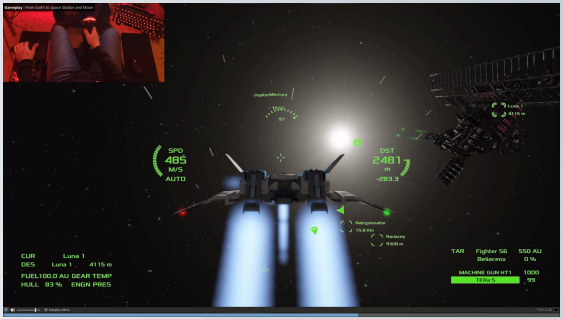
Space Nerds in Space

multiplayer star ship bridge sim, gas planets using curl noise, open source

Space Flight Simulator Examples 2/5

Flight of Nova

realistic, early access



Univoyager

real terrain with procedural detail, to be released

Space Flight Simulator Examples 3/5

Tungsten Moon

realistic, to be released

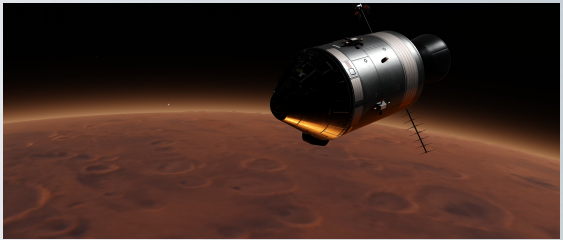
Rogue System

detailed cockpit, development on hold

Space Flight Simulator Examples 4/5

Kerbal Space Program

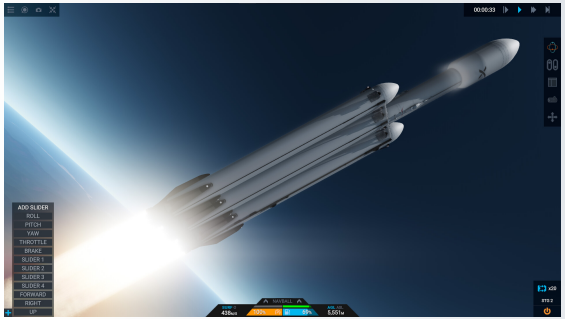
realistic, small scale planets, build spacecraft, many mods, KSP 2 team disbanded



Kitten Space Agency

realistic, spiritual successor to KSP, to be released

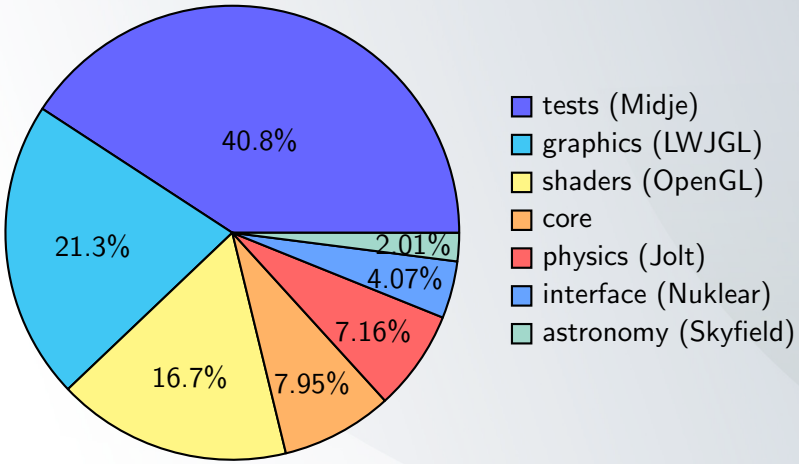
Space Flight Simulator Examples 5/5



Juno: New Origins
realistic, build spacecraft

Reentry
realistic cockpit, fixed missions, early access

Code


40.8%
21.3%
16.7%
7.95%
7.16%
4.07%
2.01%
tests (Midje)
graphics (LWJGL)
shaders (OpenGL)
core
physics (Jolt)
interface (Nuklear)
astronomy (Skyfield)

OpenGL Superbible

Pearson

informIT®

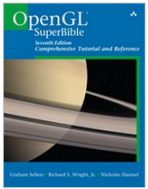
books, eBooks, and digital learning

Join | Sign In | View Your Cart

Search

Store | Formats | Deals & Promotions | Video Training | Imprints | Explore | Community

Home > Store



View Larger Image

Register your product to gain access to bonus material or receive a coupon.

OpenGL Superbible: Comprehensive Tutorial and Reference, 7th Edition

By Graham Sellers, Richard S Wright, Nicholas Haemel

Published Jul 20, 2015 by Addison-Wesley Professional. Part of the OpenGL series.

eBook

Your Price: $46.39

List Price: $57.99

Includes EPUB and PDF

About eBook Formats

Add to cart

Description | Sample Content | Updates

InformIT Promotional Mailings & Special Offers

I would like to receive exclusive offers and hear about products from InformIT and its family of brands. I can unsubscribe at any time.

Email Address

Submit

Corporate, Academic, and Employee Purchases

deps.edn

```
;; * Fastmath
;; * Lightweight Java Game Library (LWJGL)
;;   (Replace linux with windows or macos as appropriate)
{:deps {org.clojure/clojure {:mvn/version "1.12.2"}
        generateme/fastmath {:mvn/version "2.4.0"
          :exclusions [com.github.haifengl/smile-mkl org.bytedeco/openblas]}
        org.lwjgl/lwjgl {:mvn/version "3.3.6"}
        org.lwjgl/lwjgl$natives-linux {:mvn/version "3.3.6"}
        org.lwjgl/lwjgl-opengl {:mvn/version "3.3.6"}
        org.lwjgl/lwjgl-opengl$natives-linux {:mvn/version "3.3.6"}
        org.lwjgl/lwjgl-glfw {:mvn/version "3.3.6"}
        org.lwjgl/lwjgl-glfw$natives-linux {:mvn/version "3.3.6"}}
 :paths ["."]}
```

Importing Libraries

```
(require '[clojure.java.io :as io]
         '[clojure.math :refer (PI to-radians)]
         '[fastmath.vector :refer (vec3 sub add mult normalize)])
(import '[javax.imageio ImageIO]
        '[org.lwjgl BufferUtils]
        '[org.lwjgl.glfw GLFW
                         GLFWCursorPosCallbackI
                         GLFWMouseButtonCallbackI]
        '[org.lwjgl.opengl GL GL11 GL13 GL15 GL20 GL30])
```

Creating a Plain Window

Plain Window - Create

```
;; Initialise Graphics Library Framework (GLFW).
;; Create a window with OpenGL context.
(GLFW/glfwInit)

(def window-width 1280)
(def window-height 720)

(GLFW/glfwDefaultWindowHints)
(def window
     (GLFW/glfwCreateWindow window-width window-height "example" 0 0))

(GLFW/glfwShowWindow window)
(GLFW/glfwMakeContextCurrent window)
(GL/createCapabilities)
```

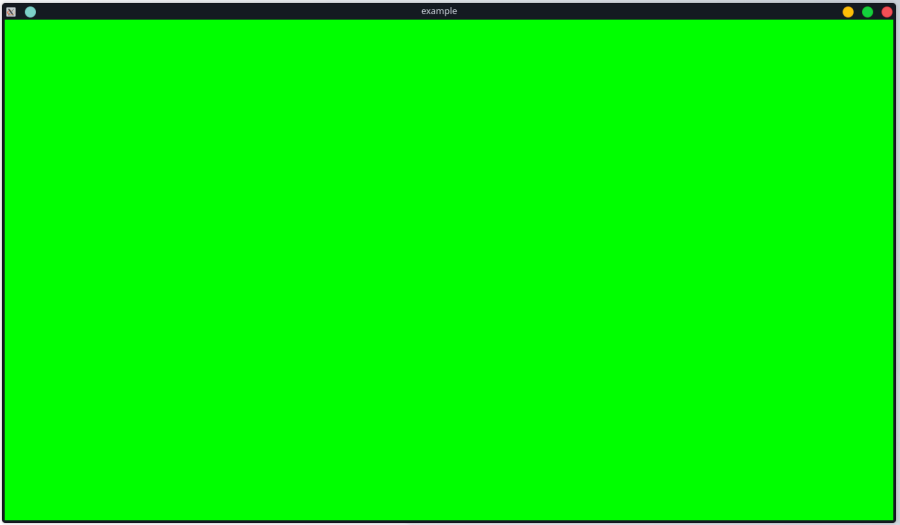
Plain Window - Event Loop

```
;; Run event loop as follows:
(while (not (GLFW/glfwWindowShouldClose window))
       (GL11/glClearColor 0.0 1.0 0.0 1.0)
       (GL11/glClear GL11/GL_COLOR_BUFFER_BIT)
       (GLFW/glfwSwapBuffers window)
       (GLFW/glfwPollEvents))
```

Plain Window - Terminate

```
;; Terminate the program like this:
(GLFW/glfwDestroyWindow window)
(GLFW/glfwTerminate)
```

Plain Window - Result



Rendering a Single Quad

Single Quad - Vertex Shader

```
#version 130
in vec3 point;
void main()
{
  gl_Position = vec4(point, 1);
}
```

```
(def vertex-source "
#version 130
in vec3 point;
void main()
{
  gl_Position = vec4(point, 1);
}")
```

Single Quad - Fragment Shader

```
#version 130
uniform vec2 iResolution;
out vec4 fragColor;
void main()
{
  vec2 uv = gl_FragCoord.xy / iResolution.xy;
  fragColor = vec4(uv, 0, 1);  // RGBA
}
```

```
(def fragment-source "
#version 130

// ...

")
```

Single Quad - Compiling Shaders

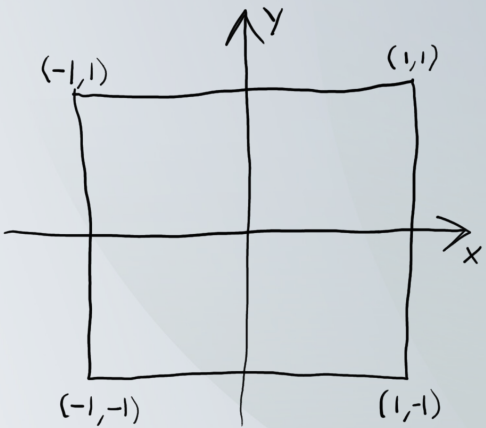
```
(defn make-shader [source shader-type]
  (let [shader (GL20/glCreateShader shader-type)]
    (GL20/glShaderSource shader source)
    (GL20/glCompileShader shader)
    (when (zero? (GL20/glGetShaderi shader GL20/GL_COMPILE_STATUS))
      (throw (Exception. (GL20/glGetShaderInfoLog shader 1024))))
    shader))

(def vertex-shader (make-shader vertex-source GL20/GL_VERTEX_SHADER))
(def fragment-shader (make-shader fragment-source GL20/GL_FRAGMENT_SHADER))
```

Single Quad - Linking Shader Program

```
(defn make-program [& shaders]
  (let [program (GL20/glCreateProgram)]
    (doseq [shader shaders]
           (GL20/glAttachShader program shader)
           (GL20/glDeleteShader shader))
    (GL20/glLinkProgram program)
    (when (zero? (GL20/glGetProgrami program GL20/GL_LINK_STATUS))
      (throw (Exception. (GL20/glGetProgramInfoLog program 1024))))
    program))

(def program (make-program vertex-shader fragment-shader))
```

Single Quad - Vertices and Indices

```
(def vertices  ;   x    y   z
  (float-array [-1.0 -1.0 0.0,
                 1.0 -1.0 0.0,
                -1.0  1.0 0.0,
                 1.0  1.0 0.0]))

(def indices
  (int-array [0 1 3 2]))
```



Single Quad - Buffers

```
(defmacro def-make-buffer [method create-buffer]
  `(defn ~method [data#]
     (let [buffer# (~create-buffer (count data#))]
       (.put buffer# data#)
       (.flip buffer#)
       buffer#)))

(def-make-buffer make-float-buffer BufferUtils/createFloatBuffer)
(def-make-buffer make-int-buffer BufferUtils/createIntBuffer)
(def-make-buffer make-byte-buffer BufferUtils/createByteBuffer)
```

Single Quad - Vertex Array Object

```
(defn setup-vao [vertices indices]
  (let [vao (GL30/glGenVertexArrays)  ; vertex array object (context)
        vbo (GL15/glGenBuffers)       ; vertex buffer object (vertices)
        ibo (GL15/glGenBuffers)]      ; index buffer object (indices)
    (GL30/glBindVertexArray vao)
    (GL15/glBindBuffer GL15/GL_ARRAY_BUFFER vbo)
    (GL15/glBufferData GL15/GL_ARRAY_BUFFER
                       (make-float-buffer vertices) GL15/GL_STATIC_DRAW)
    (GL15/glBindBuffer GL15/GL_ELEMENT_ARRAY_BUFFER ibo)
    (GL15/glBufferData GL15/GL_ELEMENT_ARRAY_BUFFER
                       (make-int-buffer indices) GL15/GL_STATIC_DRAW)
    {:vao vao :vbo vbo :ibo ibo}))

(def vao (setup-vao vertices indices))
```

Single Quad - Vertex Attributes Layout

```
(def point (GL20/glGetAttribLocation program "point"))
; void glVertexAttribPointer(index, size, type, normalized,
;                            stride, pointer);
(GL20/glVertexAttribPointer point 3 GL11/GL_FLOAT false
                            (* 3 Float/BYTES) (* 0 Float/BYTES))
(GL20/glEnableVertexAttribArray point)
```

Single Quad - Uniform Variables

```
(GL20/glUseProgram program)
(GL20/glUniform2f (GL20/glGetUniformLocation program "iResolution")
                  window-width window-height)
```

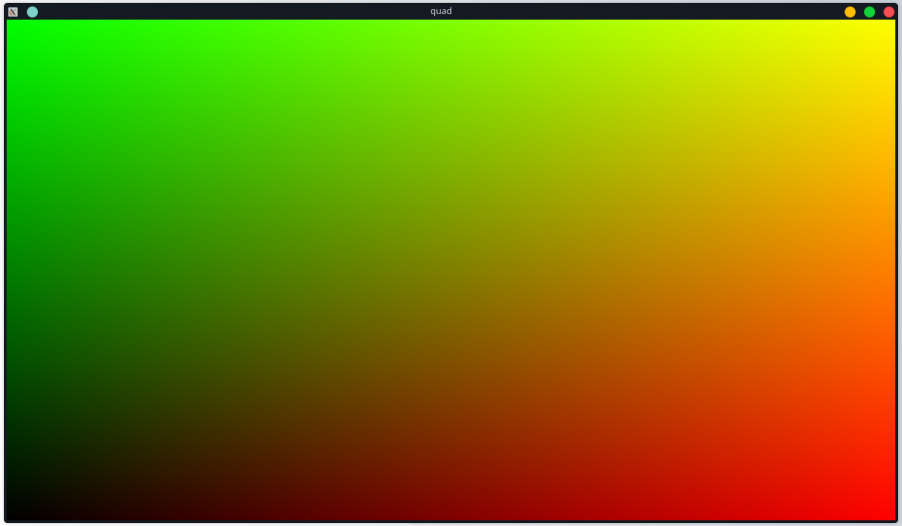
Single Quad - Event Loop

```
(while (not (GLFW/glfwWindowShouldClose window))
       (GL11/glClearColor 0.0 0.0 0.0 1.0)
       (GL11/glClear GL11/GL_COLOR_BUFFER_BIT)
       (GL11/glDrawElements GL11/GL_QUADS (count indices)
                            GL11/GL_UNSIGNED_INT 0)
       (GLFW/glfwSwapBuffers window)
       (GLFW/glfwPollEvents))
```

Single Quad - Terminate

```
(defn teardown-vao [{:keys [vao vbo ibo]}]
  (GL15/glBindBuffer GL15/GL_ELEMENT_ARRAY_BUFFER 0)
  (GL15/glDeleteBuffers ibo)
  (GL15/glBindBuffer GL15/GL_ARRAY_BUFFER 0)
  (GL15/glDeleteBuffers vbo)
  (GL30/glBindVertexArray 0)
  (GL15/glDeleteBuffers vao))

(teardown-vao vao)

(GL20/glDeleteProgram program)
```

Single Quad - Result



Rendering a Texture

Texture - Load Image

```
;; NASA CGI Moon Kit https://svs.gsfc.nasa.gov/4720/
;; Lunar Reconnaissance Orbiter Camera (LROC)
(def color (ImageIO/read (io/file "lroc_color_poles_2k.tif")))
(def color-raster (.getRaster color))
(def color-width (.getWidth color-raster))
(def color-height (.getHeight color-raster))
(def color-channels (.getNumBands color-raster))
(assert (= color-channels 3))
;; reading bytes into int-array for brevity here
(def color-pixels (int-array (* color-width color-height color-channels)))
(.getPixels color-raster 0 0 color-width color-height color-pixels)
```

Texture - Create Texture

```
(def buffer (make-byte-buffer (byte-array (map unchecked-byte color-pixels)))
(def texture-color (GL11/glGenTextures))
(GL11/glBindTexture GL11/GL_TEXTURE_2D texture-color)
(GL11/glTexImage2D GL11/GL_TEXTURE_2D 0 GL11/GL_RGBA
                   color-width color-height 0
                   GL11/GL_RGB GL11/GL_UNSIGNED_BYTE
                   buffer)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_MIN_FILTER
                      GL11/GL_LINEAR)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_MAG_FILTER
                      GL11/GL_LINEAR)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_WRAP_S
                      GL11/GL_REPEAT)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_WRAP_T
                      GL11/GL_REPEAT)
(GL11/glBindTexture GL11/GL_TEXTURE_2D 0)
```

Texture - Color Lookup

```
#version 130

uniform sampler2D moon;

vec3 color(vec2 uv)
{
  return texture(moon, vec2(uv.x, 1.0 - uv.y)).rgb;
}
```

Texture - Fragment Shader

```
#version 130

uniform vec2 iResolution;
out vec4 fragColor;

vec3 color(vec2 uv);

void main()
{
  fragColor = vec4(color(gl_FragCoord.xy / iResolution.xy), 1.0);
}
```

Texture - Uniform Variables

```
(GL20/glUniform1i (GL20/glGetUniformLocation program "moon") 0)
(GL13/glActiveTexture GL13/GL_TEXTURE0)
(GL11/glBindTexture GL11/GL_TEXTURE_2D texture-color)
```

Texture - Terminate

```
(GL11/glDeleteTextures texture-color)
```

Texture - Result

Rendering a Cube

Cube - Mouse Callbacks

```
(def mouse-pos (atom [0.0 0.0]))
(def mouse-button (atom false))

;; Callbacks invoked by (GLFW/glfwPollEvents)
(GLFW/glfwSetCursorPosCallback
  window
  (reify GLFWCursorPosCallbackI
    (invoke [_this _window xpos ypos]
            (reset! mouse-pos [xpos ypos]))))

(GLFW/glfwSetMouseButtonCallback
  window
  (reify GLFWMouseButtonCallbackI
    (invoke [_this _window _button action _mods]
            (reset! mouse-button (= action GLFW/GLFW_PRESS)))))
```

Cube - Vertex Shader 1/2

```
#version 130
#define M_PI 3.1415926535897932384626433832795
uniform float fov;  // field of view
uniform float distance;
uniform vec2 iResolution;
uniform vec2 iMouse;
in vec3 point;
out vec3 vpoint;
void main()
{
  float alpha = iMouse.x / iResolution.x * M_PI * 2.0 + M_PI;
  float beta = (iMouse.y / iResolution.y - 0.5) * M_PI * 2.0;
  // ...
}
```

Cube - Vertex Shader 2/2

```
{ // ...
  mat3 rot_y = mat3(vec3(cos(alpha), 0, sin(alpha)),
                    vec3(0, 1, 0),
                    vec3(-sin(alpha), 0, cos(alpha)));
  mat3 rot_x = mat3(vec3(1, 0, 0),
                    vec3(0, cos(beta), -sin(beta)),
                    vec3(0, sin(beta), cos(beta)));
  vec3 p = rot_x * rot_y * point + vec3(0, 0, distance);
  float f = 1.0 / tan(fov / 2.0);  // focal length
  float aspect = iResolution.x / iResolution.y;
  float proj_x = p.x / p.z * f;
  float proj_y = p.y / p.z * f * aspect;
  float proj_z = p.z / (2.0 * distance);
  gl_Position = vec4(proj_x, proj_y, proj_z, 1);
  vpoint = point;
}
```

Cube - Spherical Texture Mapping

```
#version 130

#define PI 3.14159265358979323846264338327795

vec2 uv(vec3 p)
{
  float lon = atan(p.x, -p.z);
  float lat = atan(p.y, length(p.xz));
  float u = lon / (2.0 * PI) + 0.5;
  float v = 0.5 + lat / PI;
  return vec2(u, v);
}
```

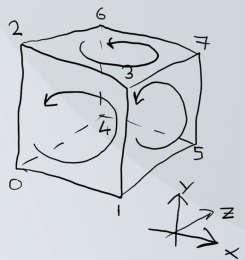
Cube - Fragment Shader

```
#version 130

in vec3 vpoint;
out vec4 fragColor;

vec2 uv(vec3 p);
vec3 color(vec2 uv);

void main()
{
  fragColor = vec4(color(uv(vpoint)), 1);
}
```

Cube - Vertices and Indices

```
(def vertices  ;   x    y    z
  (float-array [-1.0 -1.0 -1.0,
                 1.0 -1.0 -1.0,
                -1.0  1.0 -1.0,
                 1.0  1.0 -1.0,
                -1.0 -1.0  1.0,
                 1.0 -1.0  1.0,
                -1.0  1.0  1.0,
                 1.0  1.0  1.0]))
```

```
(def indices
  (int-array [0 1 3 2,
              6 7 5 4,
              0 2 6 4,
              5 7 3 1,
              2 3 7 6,
              4 5 1 0]))
```



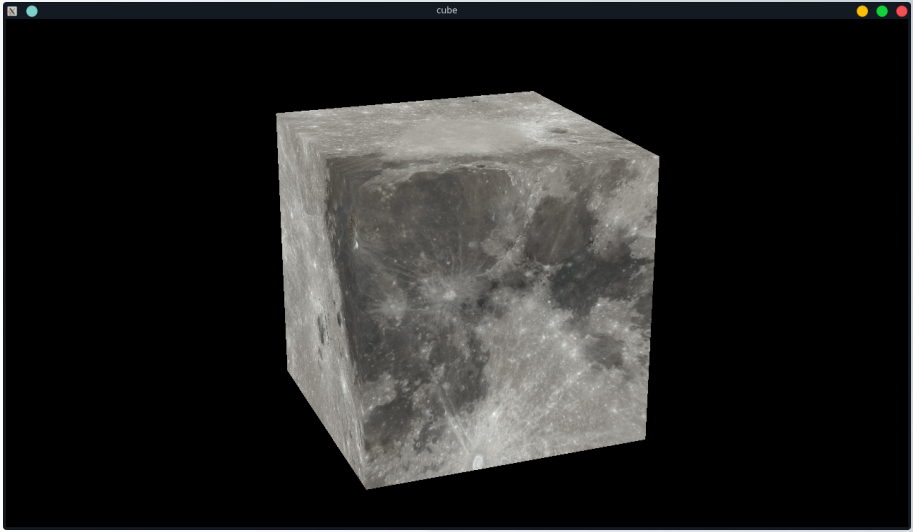
Cube - Uniform Variables

```
(GL20/glUniform1f (GL20/glGetUniformLocation program "fov")
                  (to-radians 35.0))
(GL20/glUniform1f (GL20/glGetUniformLocation program "distance") 10.0)
```

Cube - Event Loop

```
(GL11/glEnable GL11/GL_CULL_FACE)
(GL11/glCullFace GL11/GL_BACK)

(while (not (GLFW/glfwWindowShouldClose window))
       (when @mouse-button
         (GL20/glUniform2f (GL20/glGetUniformLocation program "iMouse")
                           (@mouse-pos 0) (@mouse-pos 1)))
       (GL11/glClearColor 0.0 0.0 0.0 1.0)
       (GL11/glClear GL11/GL_COLOR_BUFFER_BIT)
       (GL11/glDrawElements GL11/GL_QUADS (count indices)
                            GL11/GL_UNSIGNED_INT 0)
       (GLFW/glfwSwapBuffers window)
       (GLFW/glfwPollEvents))
```

Cube - Result



Rendering a Sphere

Sphere - Vertices and Indices 1/4

```
(def points
  [(vec3 -1.0 -1.0 -1.0)
   (vec3  1.0 -1.0 -1.0)
   (vec3 -1.0  1.0 -1.0)
   (vec3  1.0  1.0 -1.0)
   (vec3 -1.0 -1.0  1.0)
   (vec3  1.0 -1.0  1.0)
   (vec3 -1.0  1.0  1.0)
   (vec3  1.0  1.0  1.0)])
```

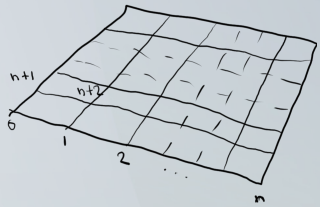
```
(def quads
  [[0 1 3 2]
   [6 7 5 4]
   [0 2 6 4]
   [5 7 3 1]
   [2 3 7 6]
   [4 5 1 0]])
```

Sphere - Vertices and Indices 2/4

```
(def radius 1737.4)

(def corners
  (map (fn [[i _ _ _]] (nth points i))
       quads))

(def u-vectors
  (map (fn [[i j _ _]] (sub (nth points j) (nth points i)))
       quads))

(def v-vectors
  (map (fn [[i _ _ l]] (sub (nth points l) (nth points i)))
       quads))
```

Sphere - Vertices and Indices 3/4

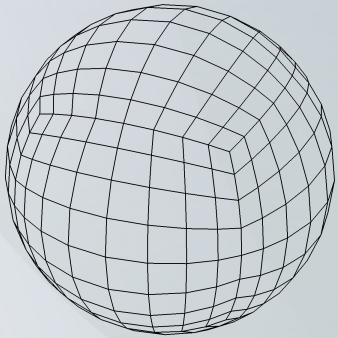
```
(defn sphere-points [n corner u-vector v-vector]
  (for [j (range (inc n)) i (range (inc n))]
       (-> corner
           (add (mult u-vector (/ i n)))
           (add (mult v-vector (/ j n)))
           normalize
           (mult radius))))
```




```
(defn sphere-indices [n face]
  (for [j (range n) i (range n)]
       (let [offset (+ (* face (inc n) (inc n)) (* j (inc n)) i)]
         [offset
          (+ offset 1)
          (+ offset (inc n) 1)
          (+ offset (inc n))])))
```

Sphere - Vertices and Indices 4/4

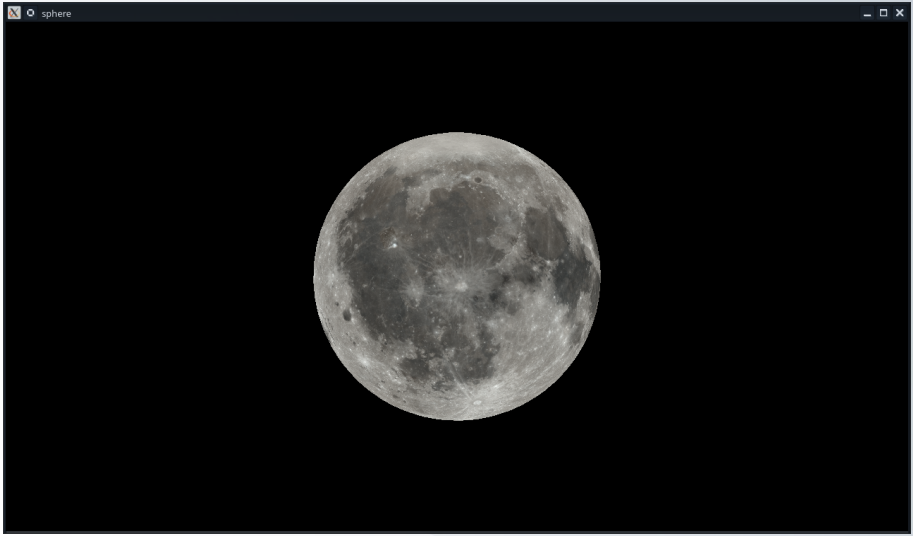
```
(def n 16)

(def vertices
  (float-array
    (flatten
      (map (partial sphere-points n)
           corners u-vectors v-vectors))))

(def indices
  (int-array
    (flatten
      (map (partial sphere-indices n)
           (range 6)))))
```



Sphere - Uniform Variables

```
(GL20/glUniform1f (GL20/glGetUniformLocation program "distance")
                  (* radius 10.0))
```

Sphere - Result



Rendering the Moon

Moon - Load Elevation Data

```
;; NASA CGI Moon Kit https://svs.gsfc.nasa.gov/4720/
;; Lunar Digital Elevation Model (LDEM)
(def ldem (ImageIO/read (io/file "ldem_4.tif")))
(def ldem-raster (.getRaster ldem))
(def ldem-width (.getWidth ldem))
(def ldem-height (.getHeight ldem))
(def ldem-pixels (float-array (* ldem-width ldem-height)))
(.getPixels ldem-raster 0 0 ldem-width ldem-height ldem-pixels)
(def resolution (/ (* 2.0 PI radius) ldem-width))
```

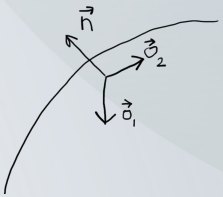
Moon - Create Elevation Texture

```
(def texture-ldem (GL11/glGenTextures))
(GL11/glBindTexture GL11/GL_TEXTURE_2D texture-ldem)
(GL11/glTexImage2D GL11/GL_TEXTURE_2D 0 GL30/GL_R32F
                   ldem-width ldem-height 0
                   GL11/GL_RED GL11/GL_FLOAT
                   ldem-pixels)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_MIN_FILTER
                      GL11/GL_LINEAR)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_MAG_FILTER
                      GL11/GL_LINEAR)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_WRAP_S
                      GL11/GL_REPEAT)
(GL11/glTexParameteri GL11/GL_TEXTURE_2D GL11/GL_TEXTURE_WRAP_T
                      GL11/GL_REPEAT)
```

Moon - Horizon Matrix

```
#version 130
vec3 orthogonal_vector(vec3 n)
{
  vec3 b;
  if (abs(n.x) <= abs(n.y)) {
    if (abs(n.x) <= abs(n.z))
      b = vec3(1, 0, 0);
    else
      b = vec3(0, 0, 1);
  } else {
    if (abs(n.y) <= abs(n.z))
      b = vec3(0, 1, 0);
    else
      b = vec3(0, 0, 1);
  };
  return normalize(cross(n, b));
}
```

```
#version 130
vec3 orthogonal_vector(vec3 n);

mat3 oriented_matrix(vec3 n)
{
  vec3 o1 = orthogonal_vector(n);
  vec3 o2 = cross(n, o1);
  return mat3(n, o1, o2);
}
```



Moon - Elevation Lookup

```
#version 130

uniform sampler2D ldem;

vec2 uv(vec3 p);

float elevation(vec3 p)
{
  vec2 uv = uv(p);
  return texture(ldem, vec2(uv.x, 1.0 - uv.y)).r;
}
```

Moon - Terrain Normal Vector

```
#version 130

uniform float resolution;

float elevation(vec3 p);

vec3 normal(mat3 horizon, vec3 p)
{
  vec3 pl = p + horizon * vec3(0, -1,  0) * resolution;
  vec3 pr = p + horizon * vec3(0,  1,  0) * resolution;
  vec3 pu = p + horizon * vec3(0,  0, -1) * resolution;
  vec3 pd = p + horizon * vec3(0,  0,  1) * resolution;
  vec3 u = horizon * vec3(elevation(pr) - elevation(pl), 2 * resolution, 0);
  vec3 v = horizon * vec3(elevation(pd) - elevation(pu), 0, 2 * resolution);
  return normalize(cross(u, v));
}
```

Moon - Fragment Shader 1/2

```
#version 130

#define AMBIENT 0.1
#define DIFFUSE 0.9

uniform vec3 light;

in vec3 vpoint;
out vec4 fragColor;

vec2 uv(vec3 p);
vec3 color(vec2 uv);
mat3 oriented_matrix(vec3 n);
vec3 normal(mat3 horizon, vec3 p);

// ...
```

Moon - Fragment Shader 2/2

```
// ...

void main()
{
  mat3 horizon = oriented_matrix(normalize(vpoint));
  vec3 normal = normal(horizon, vpoint);
  float brightness = AMBIENT + DIFFUSE * max(0.0, dot(light, normal));
  fragColor = vec4(color(uv(vpoint)) * brightness, 1);
}
```

Moon - Uniform Variables

```
(def light (normalize (vec3 -1 0 -1)))

;; ...
(GL20/glUniform1f (GL20/glGetUniformLocation program "resolution")
                  resolution)
(GL20/glUniform3f (GL20/glGetUniformLocation program "light")
                  (light 0) (light 1) (light 2))
(GL20/glUniform1i (GL20/glGetUniformLocation program "ldem") 1)
;; ...
(GL13/glActiveTexture GL13/GL_TEXTURE1)
(GL11/glBindTexture GL11/GL_TEXTURE_2D texture-ldem)
```

Moon - Terminate

```
(GL11/glDeleteTextures texture-ldem)
```

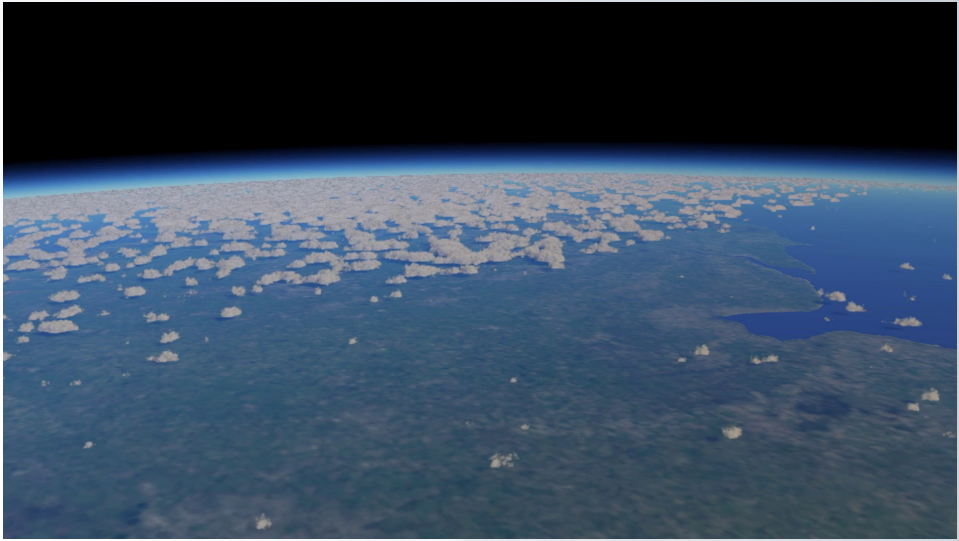
Moon - Result

In Practice

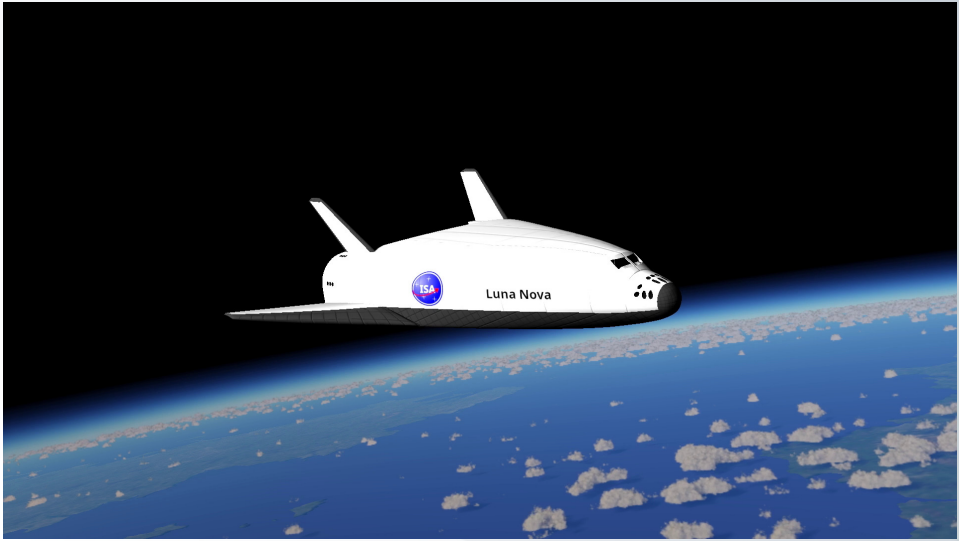
- flip images when loading and saving
- high resolution data
- multiple tiles
- multiresolution map
- precomputed normal maps
- tessellation shaders to increase mesh resolution
- elevation data to deform the mesh
- load map tiles in the background using *Clojure futures*

sfsim 1/4

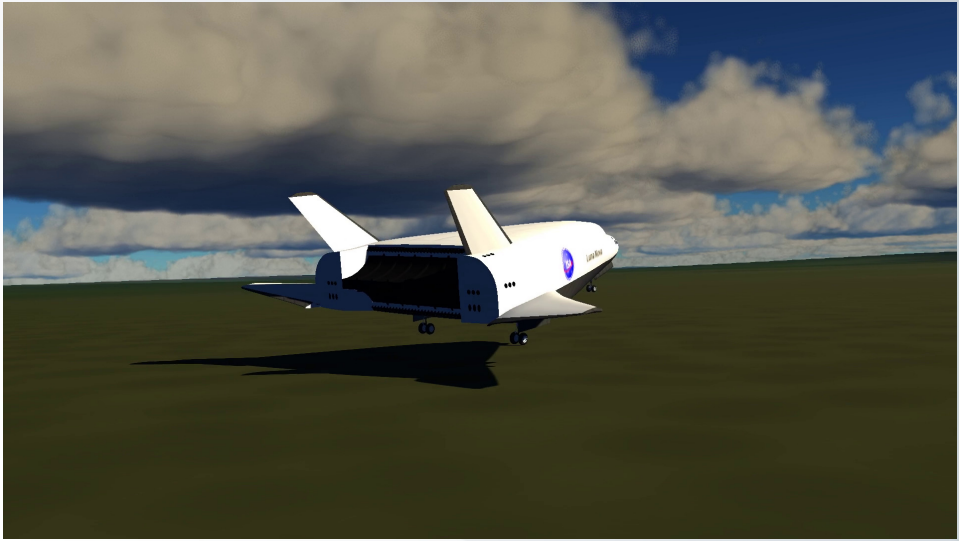
sfsim 2/4



sfsim 3/4



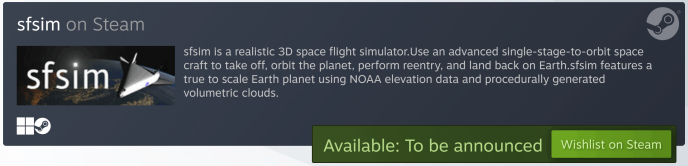
sfsim 4/4



Thanks for listening!

Thanks to Daniel Slutsky for hosting Macroexpand-Noj 2025!
Thanks to Timothy Pratley for helping me get the reproducable code online.

- Source Code: https://github.com/wedesoft/sfsim
- Maintainer: Jan Wedekind
- Contributors: Ben Sless



TDD with OpenGL 1/2

```
(def orthogonal-probe
  (template/fn [x y z]
    "#version 410 core
out vec3 fragColor;
vec3 orthogonal_vector(vec3 n);
void main()
{
  fragColor = orthogonal_vector(vec3(<%= x %>, <%= y %>, <%= z %>));
}"))

(def orthogonal-test
    (shader-test (fn [_program]) orthogonal-probe orthogonal-vector))
```

TDD with OpenGL 2/2

```
(tabular "Shader for generating an orthogonal vector"
         (fact (orthogonal-test [] [?x ?y ?z])
               => (roughly-vector (vec3 ?rx ?ry ?rz) 1e-6))
         ?x  ?y  ?z ?rx  ?ry  ?rz
         1   0   0  0    0    1
         2   0   0  0    0    1
         0   1   0  0    0   -1
         0   2   0  0    0   -1
         0   0   1  0    1    0
         0   0   2  0    1    0
         0.6 0.8 0  0.8 -0.6  0
         3   4   0  0.8 -0.6  0)
```