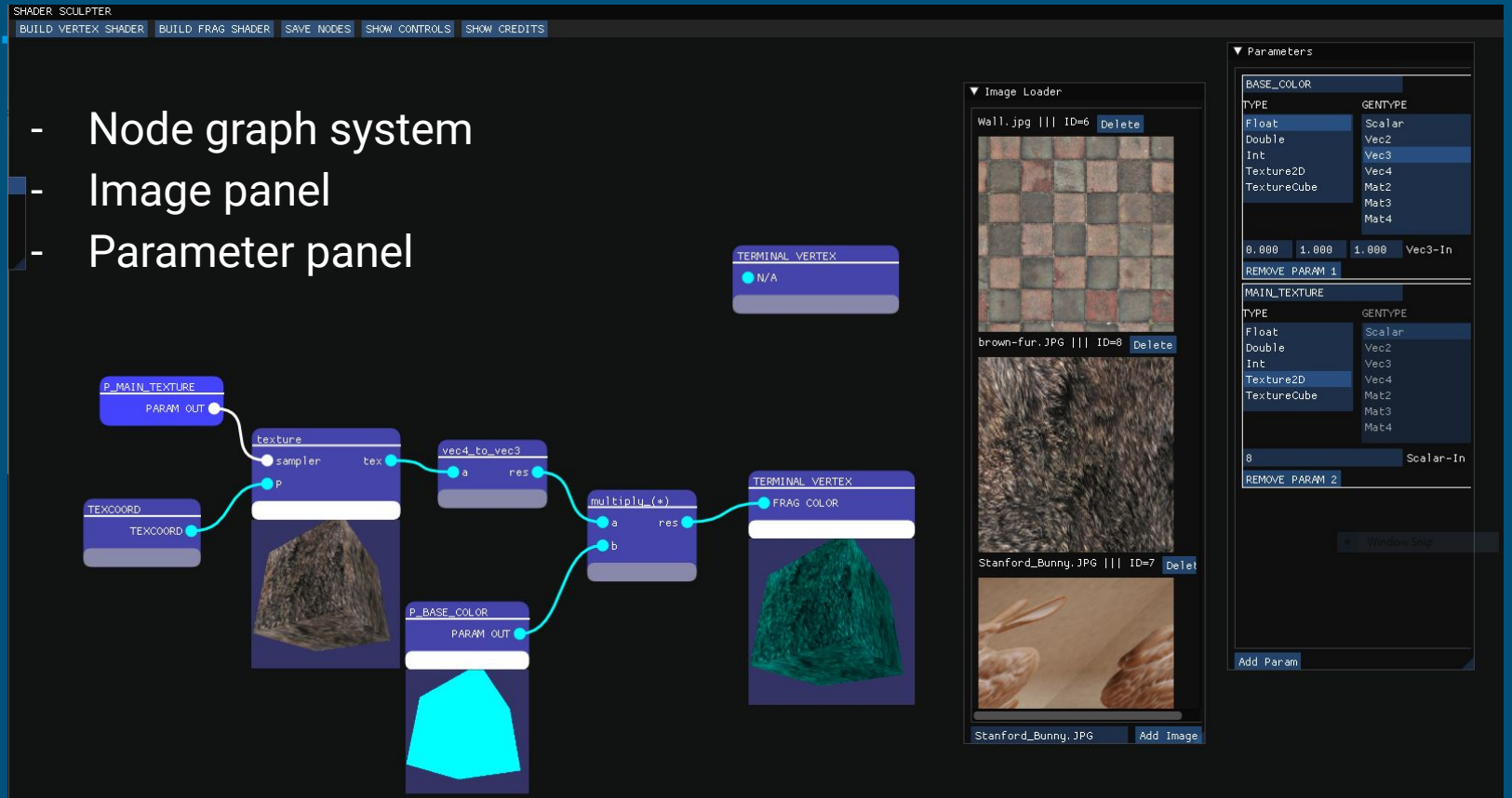
Shader Sculptor

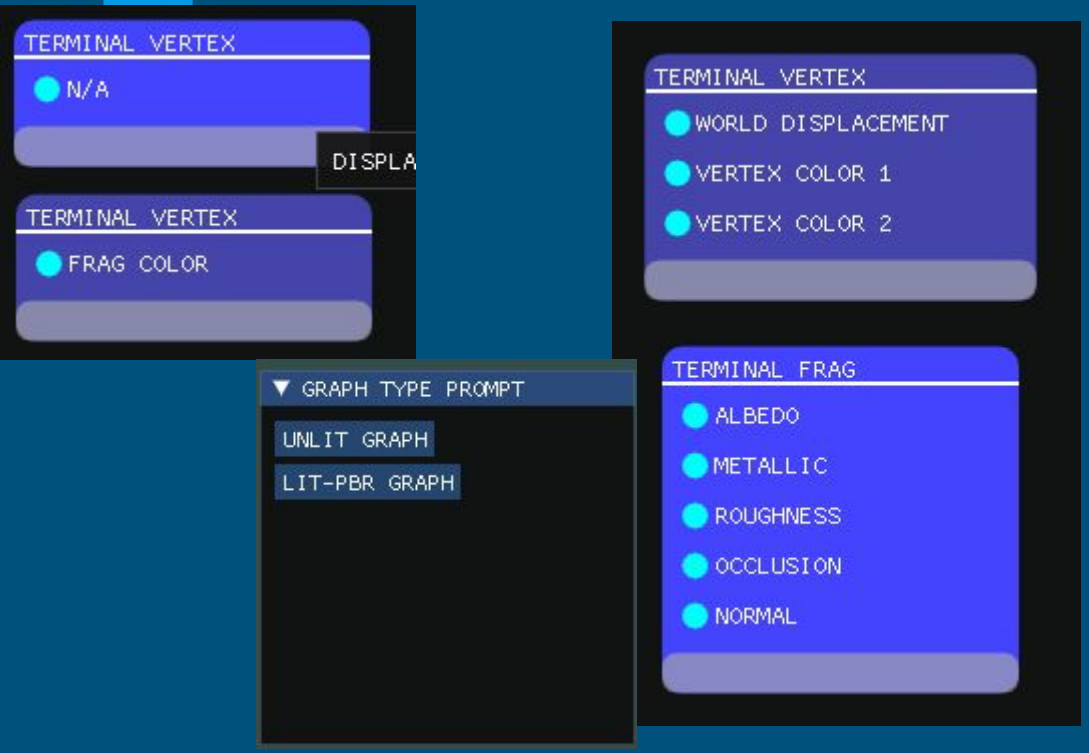
Jay Idema

Main Graph UI

- Node graph system
- Image panel
- Parameter panel



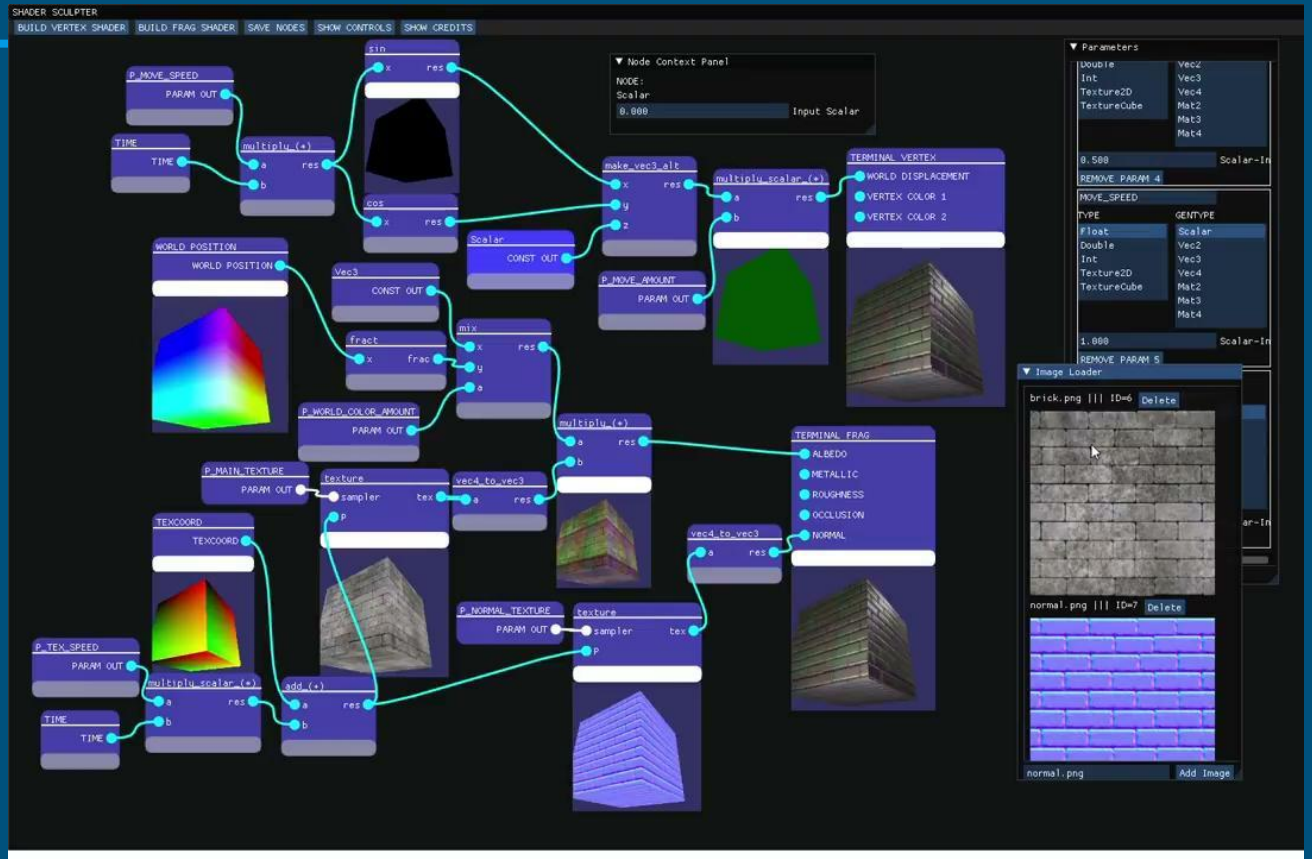
Boilerplates




```cpp
/**
 * @brief Shader with no lighting support.
 *
 */
class Unlit_Boilerplate_Manager: public SS_Boilerplate_Manager{
public:
    Unlit_Boilerplate_Manager();
    void init() override;
    std::string get_vert_terminal_boilerplate_code();
    std::string get_frag_terminal_boilerplate_code();
    class ga_material* make_material() override;
};

/**
 * @brief Shader with PBR-like shading.
 * Allows albedo, roughness, metalness, and lights.
 *
 */
class PBR_Lit_Boilerplate_Manager: public SS_Boilerplate_Manager{
public:
    PBR_Lit_Boilerplate_Manager();
    void init() override;
    std::string get_vert_terminal_boilerplate_code();
    std::string get_frag_terminal_boilerplate_code();
    class ga_material* make_material() override;
};
```

Intermediate Display



- Brick textures by Zabacar

Save out Shaders

```
// FRAGMENT CODE
        // Node TIME, id=17
        // Node P_TEX_SPEED, id=19
        vec2 INTERNAL_VAR_18_0 = ( P_TEX_SPEED* u_time);  // Node multiply_scalar_(*), id=18
        // Node TEXCOORD, id=16
        // Node P_MAIN_TEXTURE, id=11
        // Node WORLD POSITION, id=8
        vec2 INTERNAL_VAR_20_0 = ( f_texcoord+ INTERNAL_VAR_18_0);  // Node add_(+), id=20
        // Node P_NORMAL_TEXTURE, id=10
        vec4 INTERNAL_VAR_12_0 = texture( P_MAIN_TEXTURE, INTERNAL_VAR_20_0);  // Node texture, id=12
        // Node P_WORLD_COLOR_AMOUNT, id=23
        vec3 INTERNAL_VAR_24_0 = fract( f_WorldPos);  // Node fract, id=24
        // Node Vec3, id=22
        vec4 INTERNAL_VAR_13_0 = texture( P_NORMAL_TEXTURE, INTERNAL_VAR_20_0);  // Node texture, id=13
        vec3 INTERNAL_VAR_15_0 = ( INTERNAL_VAR_12_0.xyz);  // Node vec4_to_vec3, id=15
        vec3 INTERNAL_VAR_21_0 = mix( vec3(1, 1, 1), INTERNAL_VAR_24_0, P_WORLD_COLOR_AMOUNT);  // Node mix, id=21
        vec3 INTERNAL_VAR_14_0 = ( INTERNAL_VAR_13_0.xyz);  // Node vec4_to_vec3, id=14
        vec3 INTERNAL_VAR_9_0 = ( INTERNAL_VAR_21_0* INTERNAL_VAR_15_0);  // Node multiply_(*), id=9

        vec3 albedo = INTERNAL_VAR_9_0;
        float metallic = 0.5;
        float roughness = 0.5;
        float ao = 1.0;
        vec3 ts_normal = INTERNAL_VAR_14_0;

        vec3 N = normalize(getNormalFromMapping(ts_normal));
        …
```

```
uniform  vec2 P_TEX_SPEED;
uniform sampler2D P_MAIN_TEXTURE;
uniform sampler2D P_NORMAL_TEXTURE;
uniform float P_WORLD_COLOR_AMOUNT;
uniform float P_MOVE_SPEED;
uniform float P_MOVE_AMOUNT;

void main() {
 ….
}
```

Possible Future Work

- Allowing user to specify custom varying variables
  - Linker issues and confusion
  - Allowed boilerplate itself to specify some for the user, PBR-like uses 2 ‘vertex colors’
- Saving out the actual graph structure
  - You can save out and use the shader in the viper engine,
    - The cube displayed is done using an adapted ga-cube_component class
  - Serializing and reconstructing a directed graph was not a high priority
    - Poor design also meant no simple way to serialize what a node WAS/IS
- Low time and effort for testing
  - Common functions work