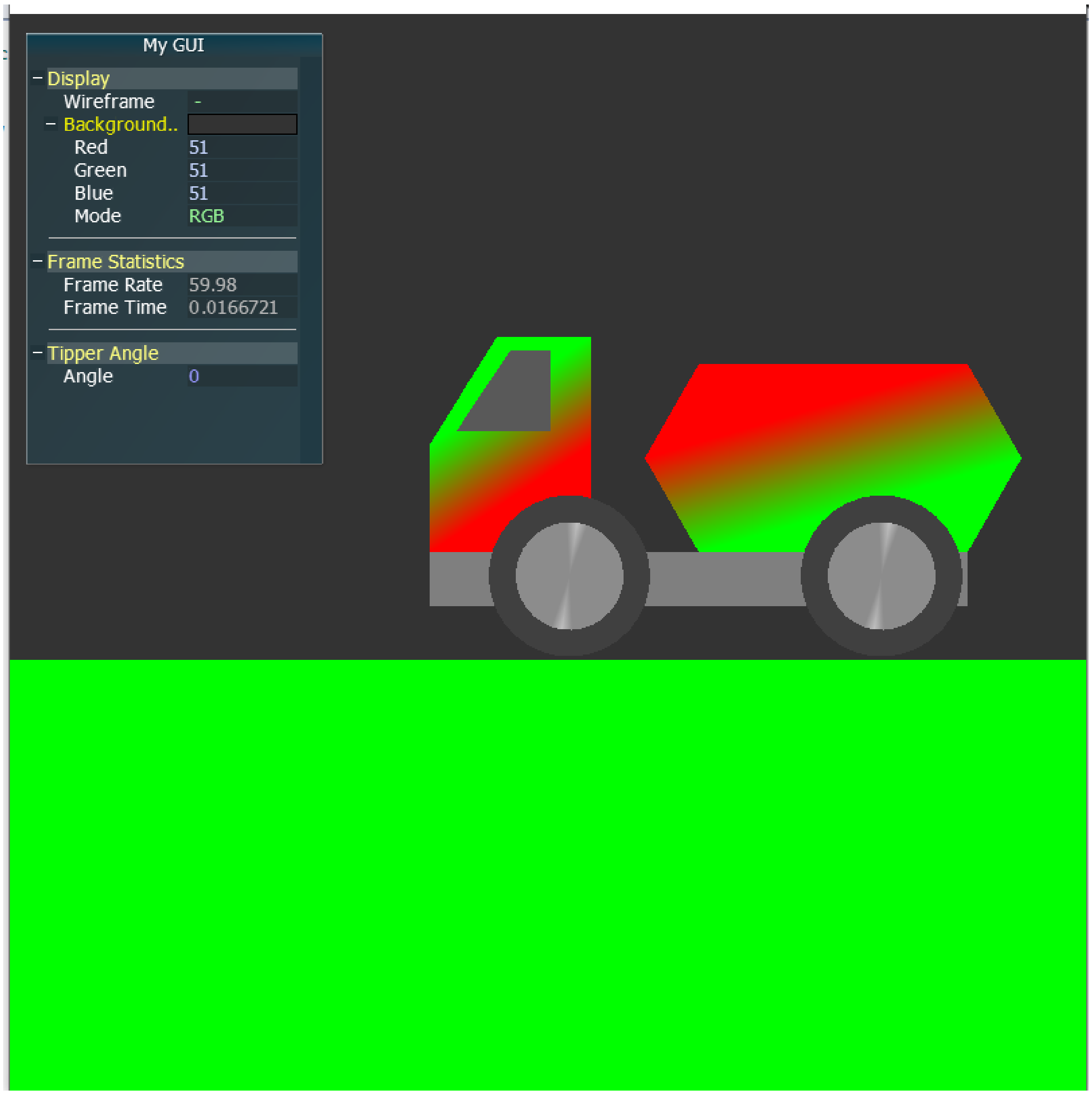
CSCI 336 Assignment 1

2D scene



```
My GUI
− Display
    Wireframe     -
  − Background..
      Red         51
      Green       51
      Blue        51
      Mode        RGB
− Frame Statistics
    Frame Rate    59.98
    Frame Time    0.0166721
− Tipper Angle
    Angle         0
```

The 2D scene with objects: ground, wheels and their hubcaps, truck chassis, and a dump box.

All objects are generated via making multiple triangles into that object by using verticles.

Wheels

```
// generate vertices for a circle based on a radius and number of slices
void generate_wheel(const float radius, const unsigned int slices, const float
scale_factor, std::vector<GLfloat>& vertices, std::vector<GLfloat>& color, const
std::string part)
{
	float slice_angle = M_PI * 2.0f / slices;	// angle of each slice
	float angle = 0;			// angle used to generate x and y
coordinates
	float x, y, z = 0;			// (x, y, z) coordinates

	// generate vertex coordinates for a circle
	for (int i = 0; i < slices + 1; i++)
	{
		x = radius * cos(angle) * scale_factor;
		y = radius * sin(angle);

		vertices.push_back(x);
		vertices.push_back(y);
		vertices.push_back(z);
		if (part == "innerWheel") {
			if (i == (slices - 1) / 2 || i == slices - 1) {
				color.push_back(color[i] + 0.2);
				color.push_back(color[i + 2] + 0.2);
				color.push_back(color[i + 3] + 0.2);
			}
			else {
				color.push_back(color[i]);
				color.push_back(color[i + 2]);
				color.push_back(color[i + 3]);
			}
		}
		else {
			color.push_back(color[i]);
			color.push_back(color[i + 2]);
			color.push_back(color[i + 3]);
		}

		// update to next angle
		angle += slice_angle;
	}
}
```

Ground

```
std::vector<GLfloat> groundVertices = {
		//ground
		-2.0f, -0.2f, 0.0f,	// vertex 0: position
		0.0f, 1.0f, 0.0f,	// vertex 0: colour
		2.0f, -0.2f, 0.0f,	// vertex 1: position
		0.0f, 1.0f, 0.0f,	// vertex 1: colour
		-2.0f, -100.0f, 0.0f,	// vertex 2: position
		0.0f, 1.0f, 0.0f,	// vertex 2: colour
		2.0f, -100.0f, 0.0f,		// vertex 3: position
		0.0f, 1.0f, 0.0f	// vertex 3: colour
	};

	std::vector<GLuint> groundIndices = {
		//ground
		0, 1, 2,	// triangle 13
		2, 1, 3	// triangle 14
	};
```

Truck Chassis

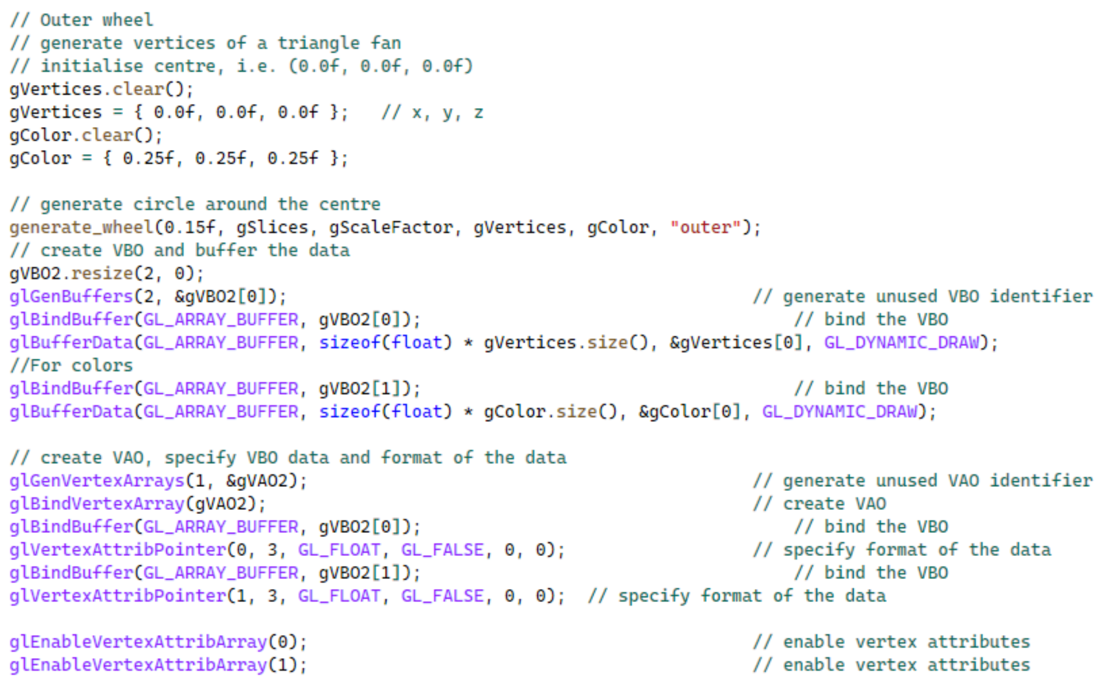
```
// vertex positions and colours
	std::vector<GLfloat> vertices = {
		//Rectangle Below window
		-1.0f, 0.2f, 0.0f,  // vertex 0: position
		0.0f, 1.0f, 0.0f,   // vertex 0: colour
		-1.0f, 0.0f, 0.0f,  // vertex 1: position
		1.0f, 0.0f, 0.0f,   // vertex 1: colour
		-0.7f, 0.2f, 0.0f,  // vertex 2: position
		1.0f, 0.0f, 0.0f,   // vertex 2: colour
		-0.7f, 0.0f, 0.0f,  // vertex 3: position
		1.0f, 0.0f, 0.0f,   // vertex 3: colour

		//Trapezium without window
		-0.875f, 0.4f, 0.0f,      // vertex 4: position
		0.0f, 1.0f, 0.0f,   // vertex 4: colour
		-1.0f, 0.2f, 0.0f,  // vertex 5: position
		0.0f, 1.0f, 0.0f,   // vertex 5: colour
		-0.7f, 0.4f, 0.0f,  // vertex 6: position
		0.0f, 1.0f, 0.0f,   // vertex 6: colour
		-0.7f, 0.2f, 0.0f,  // vertex 7: position
		1.0f, 0.0f, 0.0f,   // vertex 7: colour

		//Window
		-0.85f, 0.375f, 0.0f,     // vertex 8: position
		0.35f, 0.35f, 0.35f,      // vertex 8: colour
		-0.95f, 0.225f, 0.0f,     // vertex 9: position
		0.35f, 0.35f, 0.35f,      // vertex 9: colour
		-0.775f, 0.375f, 0.0f,// vertex 10: position
		0.35f, 0.35f, 0.35f,      // vertex 10: colour
		-0.775f, 0.225f, 0.0f,    // vertex 11: position
		0.35f, 0.35f, 0.35f,      // vertex 11: colour

		//truck base
		-1.0f, 0.0f, 0.0f,  // vertex 12: position
		0.5f, 0.5f, 0.5f,   // vertex 12: colour
		0.0f, 0.0f, 0.0f,   // vertex 13: position
		0.5f, 0.5f, 0.5f,   // vertex 13: colour
		-1.0f, -0.1f, 0.0f, // vertex 14: position
		0.5f, 0.5f, 0.5f,   // vertex 14: colour
		0.0f, -0.1f, 0.0f,  // vertex 15: position
		0.5f, 0.5f, 0.5f,   // vertex 15: colour
	};

// object indices
	std::vector<GLuint> indices = {
		//Rectangle Below window
		0, 1, 2,     // triangle 1
		2, 1, 3,     // triangle 2

		//Trapezium without window
		4, 5, 6,     // triangle 3
		6, 5, 7,     // triangle 4

		//Window
		8, 9, 10,    // triangle 5
		10, 9, 11,   // triangle 6

		//truck base
		12, 13, 14,  // triangle 7
		15, 13, 14,  // triangle 8
	};
```

Dump Box

```
std::vector<GLfloat> dumpBoxVertices = {
		//Back of truck
		//Bottom half
		0.0f, 0.0f, 0.0f,   // vertex 0: position
		0.0f, 1.0f, 0.0f,   // vertex 0: colour
		-0.5f, 0.0f, 0.0f,  // vertex 1: position
		0.0f, 1.0f, 0.0f,   // vertex 1: colour
		0.1f, 0.175f, 0.0f, // vertex 2: position
		0.0f, 1.0f, 0.0f,   // vertex 2: colour
		-0.6f, 0.175f, 0.0f,      // vertex 3: position
		1.0f, 0.0f, 0.0f,   // vertex 3: colour

		//Top half
		0.0f, 0.35f, 0.0f,  // vertex 4: position
		1.0f, 0.0f, 0.0f,   // vertex 4: colour
		-0.5f, 0.35f, 0.0f, // vertex 5: position
		1.0f, 0.0f, 0.0f    // vertex 5: colour
	};

	std::vector<GLuint> dumpBoxIndices = {
		//Back of truck
		0, 1, 2,
		2, 1, 3,

		3, 2, 4,
		4, 3, 5
	};
```

Hubcap of Wheels

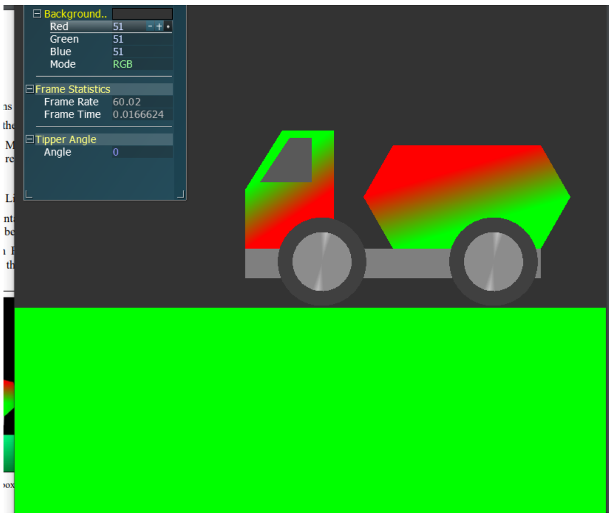
The hubcap is generated by generating an outer circle around the inner wheel(circle)



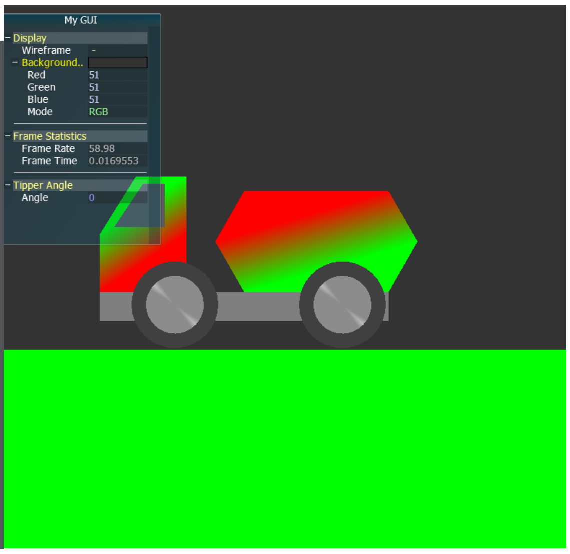
```
// Outer wheel
// generate vertices of a triangle fan
// initialise centre, i.e. (0.0f, 0.0f, 0.0f)
gVertices.clear();
gVertices = { 0.0f, 0.0f, 0.0f };   // x, y, z
gColor.clear();
gColor = { 0.25f, 0.25f, 0.25f };

// generate circle around the centre
generate_wheel(0.15f, gSlices, gScaleFactor, gVertices, gColor, "outer");
// create VBO and buffer the data
gVBO2.resize(2, 0);
glGenBuffers(2, &gVBO2[0]);                                        // generate unused VBO identifier
glBindBuffer(GL_ARRAY_BUFFER, gVBO2[0]);                                // bind the VBO
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * gVertices.size(), &gVertices[0], GL_DYNAMIC_DRAW);
//For colors
glBindBuffer(GL_ARRAY_BUFFER, gVBO2[1]);                                // bind the VBO
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * gColor.size(), &gColor[0], GL_DYNAMIC_DRAW);

// create VAO, specify VBO data and format of the data
glGenVertexArrays(1, &gVAO2);                                      // generate unused VAO identifier
glBindVertexArray(gVAO2);                                          // create VAO
glBindBuffer(GL_ARRAY_BUFFER, gVBO2[0]);                                // bind the VBO
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);             // specify format of the data
glBindBuffer(GL_ARRAY_BUFFER, gVBO2[1]);                                // bind the VBO
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0);  // specify format of the data

glEnableVertexAttribArray(0);                                      // enable vertex attributes
glEnableVertexAttribArray(1);                                      // enable vertex attributes
```

Transformations and keyboard input

Move truck left and right

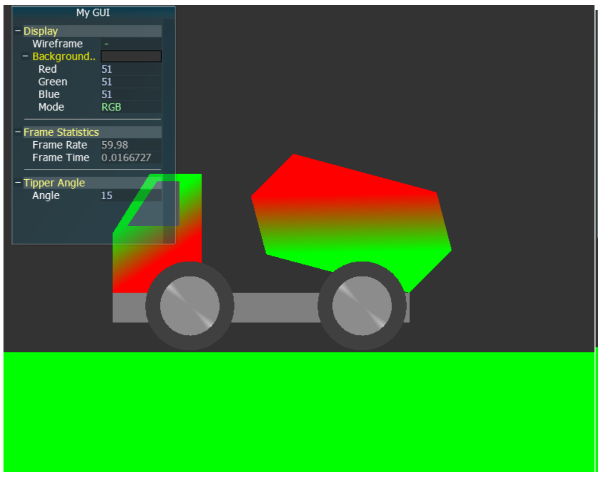


```
⊟ Background..
  Red        51   - + •
  Green      51
  Blue       51
  Mode       RGB
⊟ Frame Statistics
  Frame Rate  60.02
  Frame Time  0.0166624
⊟ Tipper Angle
  Angle       0
```

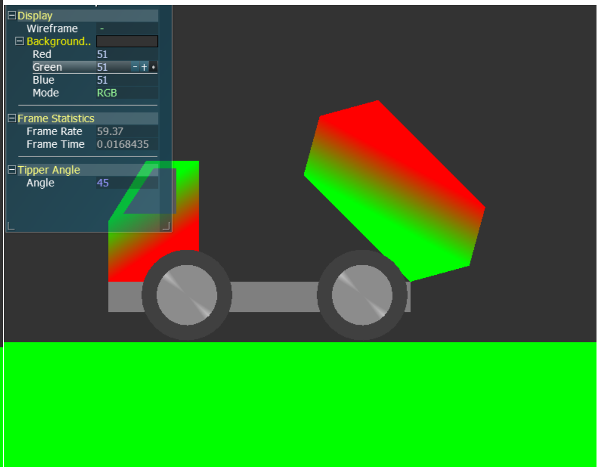


```
My GUI
- Display
  Wireframe  -
  - Background..
    Red       51
    Green     51
    Blue      51
    Mode      RGB
- Frame Statistics
  Frame Rate  58.98
  Frame Time  0.0169553
- Tipper Angle
  Angle       0
```

Wheels are rotating too.



```
My GUI
- Display
  Wireframe  -
  - Background..
    Red       51
    Green     51
    Blue      51
    Mode      RGB
- Frame Statistics
  Frame Rate  59.98
  Frame Time  0.0166727
- Tipper Angle
  Angle       15
```

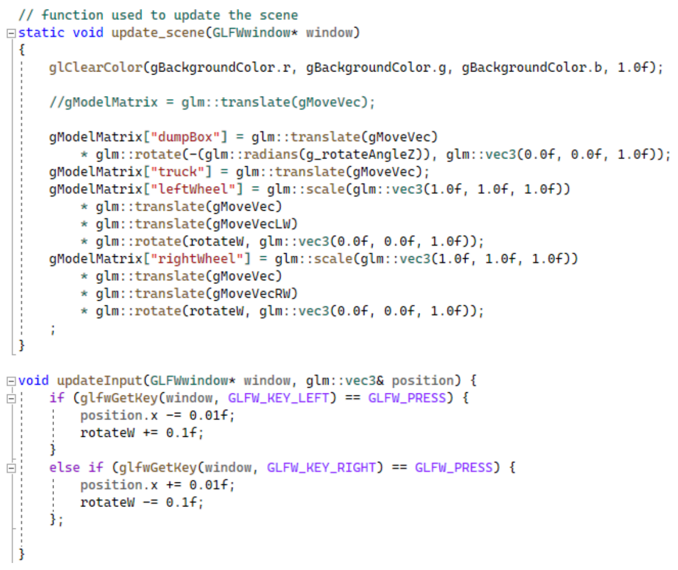


```
⊟ Display
  Wireframe  -
  ⊟ Background..
    Red       51
    Green     51   - + •
    Blue      51
    Mode      RGB
⊟ Frame Statistics
  Frame Rate  59.37
  Frame Time  0.0168435
⊟ Tipper Angle
  Angle       45
```

Able to lift/lower angle of dump box (0-45 degree)

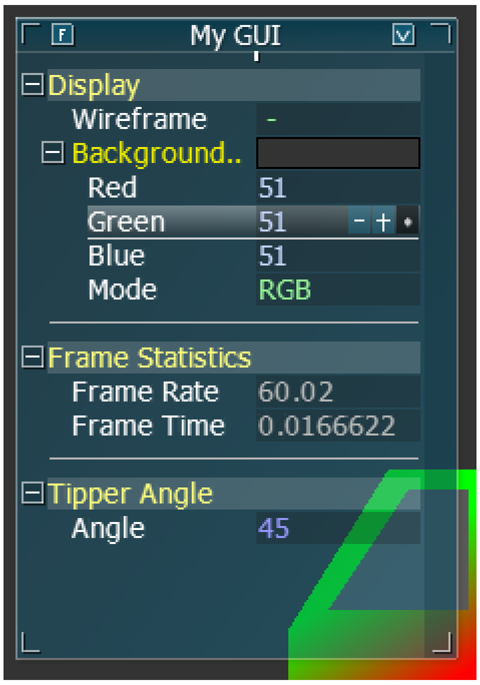
Truck is able to move left and right using translate and increase/decrease the position x.

The Wheels are able to rotate via increase/decrease the value rotateW.



```
// function used to update the scene
static void update_scene(GLFWwindow* window)
{
    glClearColor(gBackgroundColor.r, gBackgroundColor.g, gBackgroundColor.b, 1.0f);

    //gModelMatrix = glm::translate(gMoveVec);

    gModelMatrix["dumpBox"] = glm::translate(gMoveVec)
        * glm::rotate(-(glm::radians(g_rotateAngleZ)), glm::vec3(0.0f, 0.0f, 1.0f));
    gModelMatrix["truck"] = glm::translate(gMoveVec);
    gModelMatrix["leftWheel"] = glm::scale(glm::vec3(1.0f, 1.0f, 1.0f))
        * glm::translate(gMoveVec)
        * glm::translate(gMoveVecLW)
        * glm::rotate(rotateW, glm::vec3(0.0f, 0.0f, 1.0f));
    gModelMatrix["rightWheel"] = glm::scale(glm::vec3(1.0f, 1.0f, 1.0f))
        * glm::translate(gMoveVec)
        * glm::translate(gMoveVecRW)
        * glm::rotate(rotateW, glm::vec3(0.0f, 0.0f, 1.0f));
    ;
}

void updateInput(GLFWwindow* window, glm::vec3& position) {
    if (glfwGetKey(window, GLFW_KEY_LEFT) == GLFW_PRESS) {
        position.x -= 0.01f;
        rotateW += 0.1f;
    }
    else if (glfwGetKey(window, GLFW_KEY_RIGHT) == GLFW_PRESS) {
        position.x += 0.01f;
        rotateW -= 0.1f;
    };

}
```

User Interface



My GUI

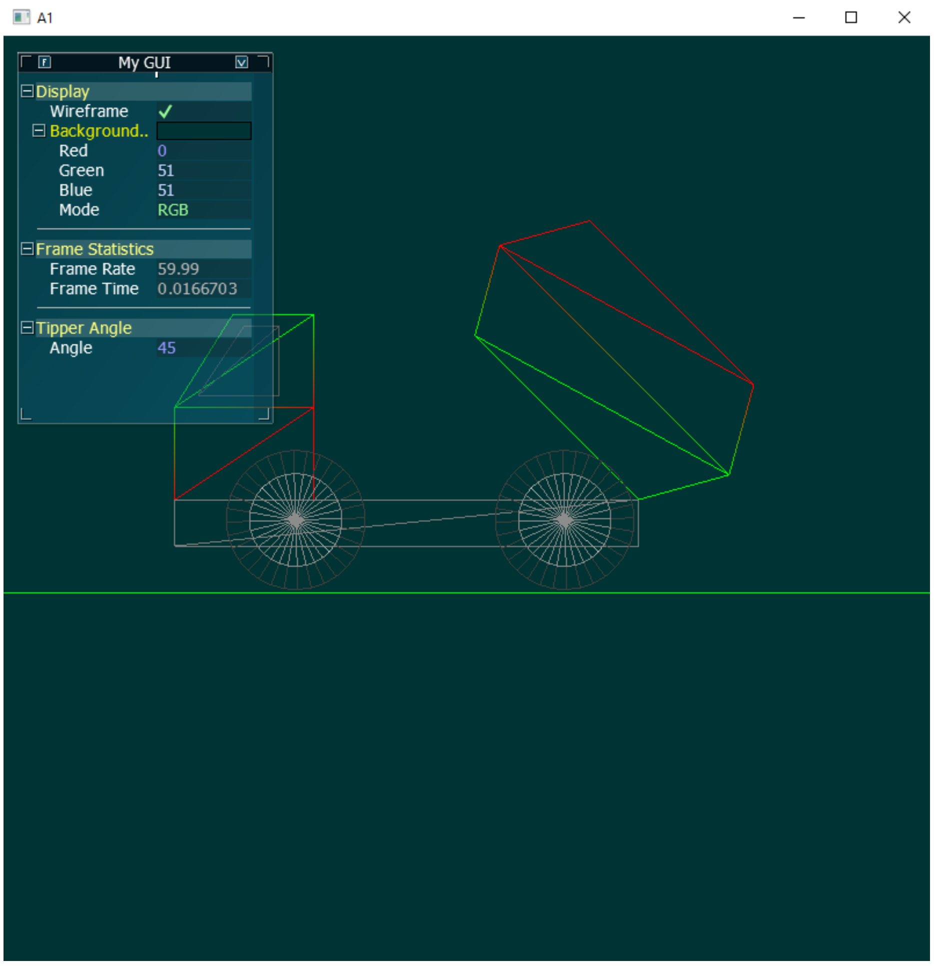
Display
Wireframe -
Background..
Red 51
Green 51
Blue 51
Mode RGB

Frame Statistics
Frame Rate 60.02
Frame Time 0.0166622

Tipper Angle
Angle 45

Interface with wireframe toggle, background colour change, displays fps and frame time, display and adjust dump box angle.

Interface is created using twBars, and linked to the corresponding values via variable declaration in the code.



```
My GUI
⊟Display
  Wireframe      ✓
 ⊟Background..
   Red           0
   Green         51
   Blue          51
   Mode          RGB
⊟Frame Statistics
  Frame Rate     59.99
  Frame Time     0.0166703
⊟Tipper Angle
  Angle          45
```

```
// create and populate tweak bar elements
TwBar* create_UI(const std::string name)
{
    // create a tweak bar
    TwBar* twBar = TwNewBar(name.c_str());

    TwWindowSize(gWindowWidth, gWindowHeight);

    TwDefine(" TW_HELP visible=false "); // disable help menu
    TwDefine(" GLOBAL fontsize=3 "); // set large font size

    //Define GUI
    TwDefine(" Main label='My GUI' refresh=0.02 text=light size='220 320' ");

    //Wireframe UI
    TwAddVarRW(twBar, "Wireframe", TW_TYPE_BOOLCPP, &gWireframe, " group='Display' ");

    //Background color UI
    TwAddVarRW(twBar, "BgColor", TW_TYPE_COLOR3F, &gBackgroundColor, " label='Background Color' group = 'Display' opened = true ");

    TwAddSeparator(twBar, nullptr, nullptr);

    //Frame statistics UI
    TwAddVarRO(twBar, "Frame Rate", TW_TYPE_FLOAT, &gFrameRate, " group='Frame Statistics' precision = 2 ");
    TwAddVarRO(twBar, "Frame Time", TW_TYPE_FLOAT, &gFrameTime, " group='Frame Statistics' ");

    TwAddSeparator(twBar, nullptr, nullptr);

    TwAddVarRW(twBar, "Angle", TW_TYPE_FLOAT, &g_rotateAngleZ, " group='Tipper Angle' min=0.0 max=45.0 step=1");

    return twBar;
}
```