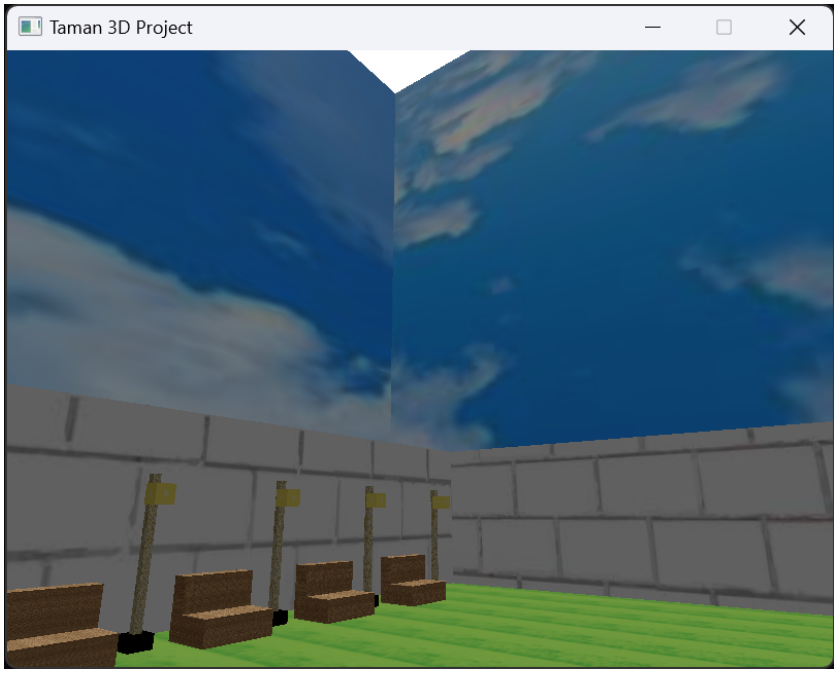
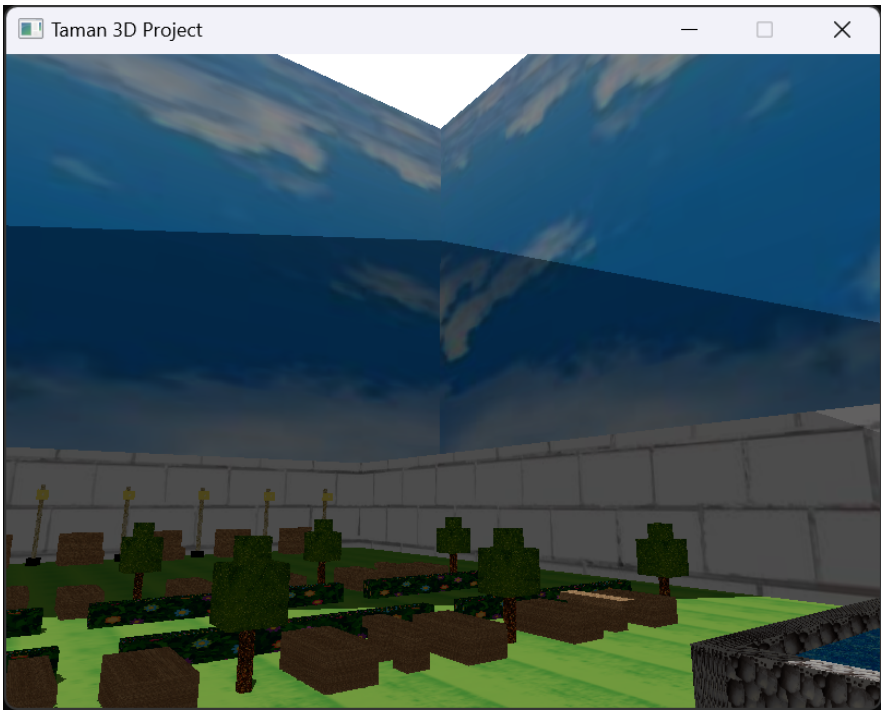
FINAL PROGRESS REPORT

I. Data Tim/Kelompok *(diisi oleh kelompok)*

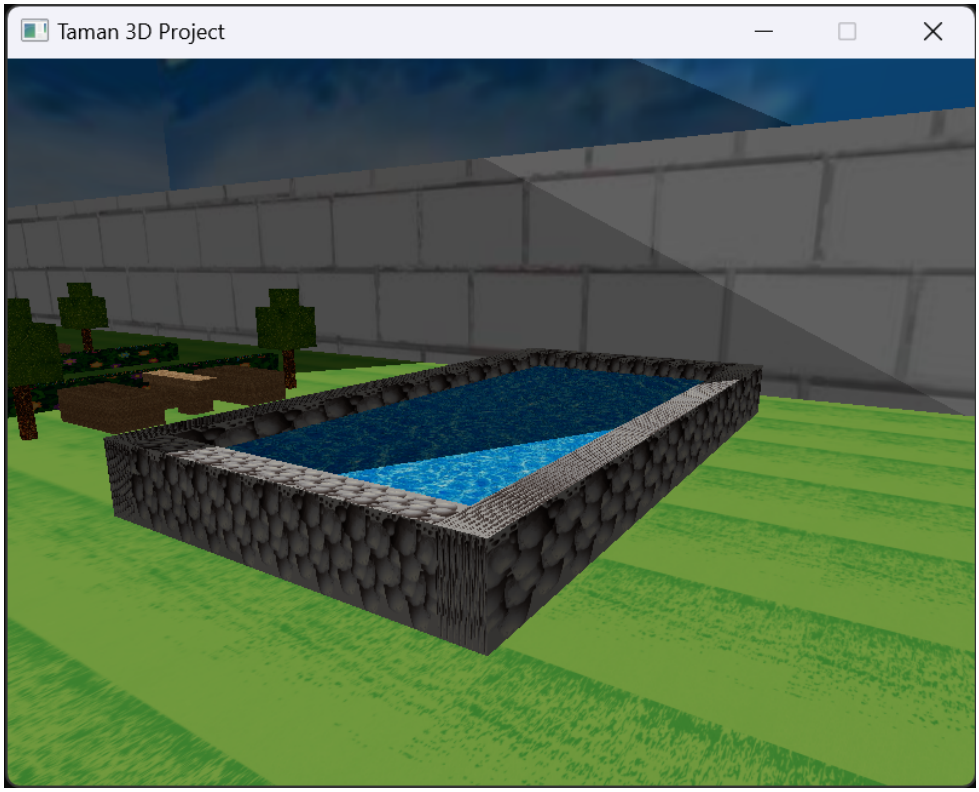
Nama Tim/Kelompok	:	Yeppeo
Judul Aplikasi	:	Taman 3D
Biodata Tim	:	Fakhriyyatum Muslimah - 205150201111004 Zahra Diva - 205150201111005

II. Screenshot aplikasi .




Taman 3D Project

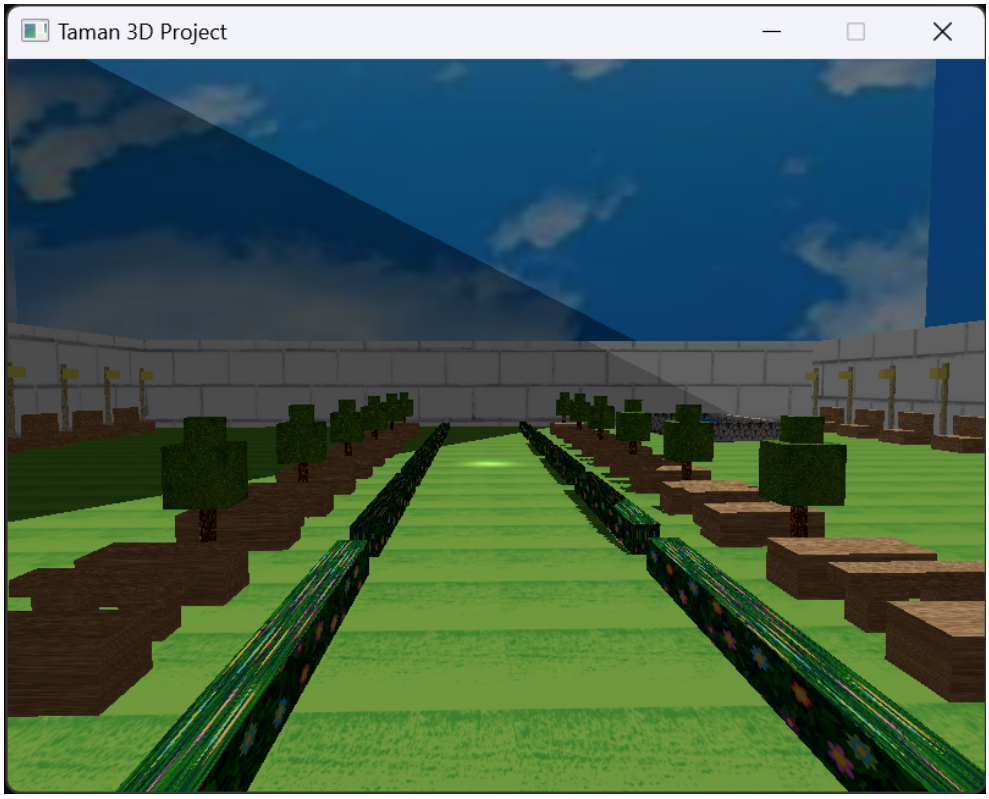

Taman 3D Project


Taman 3D Project


Taman 3D Project


Taman 3D Project


Taman 3D Project



III. Kode Program dan Penjelasannya.

- Aplication.cpp

```
#include "Application.h"

Application::Application() {

}

Application::~Application() {
}

void Application::Init() {
	// build and compile our shader program
	// ------------------------------------
	shadowmapShader = BuildShader("shadowMapping.vert",
"shadowMapping.frag", nullptr);
	depthmapShader = BuildShader("depthMap.vert",
"depthMap.frag", nullptr);

	//Generate DepthMap
	//create FBO
	// configure depth map FBO
	// -----------------------
	glGenFramebuffers(1, &depthMapFBO);
```

```
      // create depth texture
      glGenTextures(1, &depthMap);
      glBindTexture(GL_TEXTURE_2D, depthMap);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, 256,
256, 0, GL_DEPTH_COMPONENT, GL_FLOAT, NULL);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_CLAMP_TO_BORDER);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_CLAMP_TO_BORDER);
      float borderColor[] = { 1.0, 1.0, 1.0, 1.0 };
      glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR,
borderColor);

      // attach depth texture as FBO's depth buffer
      glBindFramebuffer(GL_FRAMEBUFFER, depthMapFBO);
      glFramebufferTexture2D(GL_FRAMEBUFFER,
GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, depthMap, 0);
      glDrawBuffer(GL_NONE);
      glReadBuffer(GL_NONE);
      glBindFramebuffer(GL_FRAMEBUFFER, 0);

      InitCamera();
      //membuat rumput
      BuildColoredPlane();
      //membuat langit
      BuildColoredSky();
      //membuat pagar tembok
      BuildPagar();
      //membuat batang
      BuildBatang();
      //membuat daun
      BuildDaun();
      //membuat rumput semak
      BuildRumputSemak();
      //membuat lampu taman
      BuildPenyanggaTiang();
      BuildTiang();
      BuildLampu();
      //membuat kursi
      BuildKursi();
      //membuat meja
      BuildMeja();
      //membuat kolam
      BuildKolam();
      BuildBatuKolam();
}
```

```
void Application::DeInit() {
	// optional: de-allocate all resources once they've
outlived their purpose:
	//
------------------------------------------------------------------
----------
	glDeleteVertexArrays(1, &VAO2);
	glDeleteBuffers(1, &VBO2);
	glDeleteBuffers(1, &EBO2);
}
// process all input: query GLFW whether relevant keys are
pressed/released this frame and react accordingly
//
------------------------------------------------------------------
----------------------------------------
void Application::ProcessInput(GLFWwindow* window) {
	if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS) {
		glfwSetWindowShouldClose(window, true);
	}

	// zoom camera
	// -----------
	if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_RIGHT)
== GLFW_PRESS) {
		if (fovy < 90) {
			fovy += 0.0001f;
		}
	}

	if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT) ==
GLFW_PRESS) {
		if (fovy > 0) {
			fovy -= 0.0001f;
		}
	}

	// update camera movement
	// -------------
	if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) {
		MoveCamera(CAMERA_SPEED);
	}
	if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS) {
		MoveCamera(-CAMERA_SPEED);
	}

	if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS) {
		StrafeCamera(-CAMERA_SPEED);
	}

	if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) {
		StrafeCamera(CAMERA_SPEED);
```

```
	}

	// update camera rotation
	// ----------------------
	double mouseX, mouseY;
	double midX = screenWidth / 2;
	double midY = screenHeight / 2;
	float angleY = 0.0f;
	float angleZ = 0.0f;

	// Get mouse position
	glfwGetCursorPos(window, &mouseX, &mouseY);
	if ((mouseX == midX) && (mouseY == midY)) {
		return;
	}

	// Set mouse position
	glfwSetCursorPos(window, midX, midY);

	// Get the direction from the mouse cursor, set a
resonable maneuvering speed
	angleY = (float)((midX - mouseX)) / 1000;
	angleZ = (float)((midY - mouseY)) / 1000;

	// The higher the value is the faster the camera looks
around.
	viewCamY += angleZ * 2;

	// limit the rotation around the x-axis
	if ((viewCamY - posCamY) > 8) {
		viewCamY = posCamY + 8;
	}
	if ((viewCamY - posCamY) < -8) {
		viewCamY = posCamY - 8;
	}
	RotateCamera(-angleY);
}

void Application::Update(double deltaTime) {
	angle += (float)((deltaTime * 1.5f) / 100);
	posisiZ += (float)((deltaTime * speed) / 1000);
}

void Application::Render() {
	glViewport(0, 0, this->screenWidth, this->screenHeight);

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

```
     glEnable(GL_DEPTH_TEST);

     // Step 1 Render depth of scene to texture
     // ---------------------------------------
     glm::mat4 lightProjection, lightView;
     glm::mat4 lightSpaceMatrix;
     float near_plane = 1.0f, far_plane = 7.5f;
     lightProjection = glm::ortho(-10.0f, 10.0f, -10.0f,
10.0f, near_plane, far_plane);
     lightView = glm::lookAt(glm::vec3(-2.0f, 4.0f, -1.0f),
glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0, 1.0, 0.0));
     lightSpaceMatrix = lightProjection * lightView;
     // render scene from light's point of view
     UseShader(depthmapShader);

glUniformMatrix4fv(glGetUniformLocation(this->depthmapShader,
"lightSpaceMatrix"), 1, GL_FALSE,
glm::value_ptr(lightSpaceMatrix));
     glViewport(0, 0, 256, 256);
     glBindFramebuffer(GL_FRAMEBUFFER, depthMapFBO);
     glClear(GL_DEPTH_BUFFER_BIT);

     //membuat rumput
     DrawColoredPlane(depthmapShader);
     //membuat langit
     DrawSky(depthmapShader);
     //membuat pagar tembok
     DrawPagar(depthmapShader);
     //membuat pohon
     DrawBatang(depthmapShader);
     DrawDaun(shadowmapShader);
     //membuat rumput semak
     DrawRumputSemak(depthmapShader);
     //membuat lampu taman
     DrawPenyanggaTiang(depthmapShader);
     DrawTiang(depthmapShader);
     DrawLampu(depthmapShader);
     //membuat kursi
     DrawKursi(depthmapShader);
     //membuat meja
     DrawMeja(depthmapShader);
     DrawKolam(depthmapShader);
     DrawBatuKolam(depthmapShader);

     glBindFramebuffer(GL_FRAMEBUFFER, 0);

     // Step 2 Render scene normally using generated depth map
     // ------------------------------------------------------
     glViewport(0, 0, this->screenWidth, this->screenHeight);
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
     // Pass perspective projection matrix
     UseShader(this->shadowmapShader);
     glm::mat4 projection = glm::perspective(fovy,
(GLfloat)this->screenWidth / (GLfloat)this->screenHeight,
0.1f, 100.0f);

glUniformMatrix4fv(glGetUniformLocation(this->shadowmapShader,
"projection"), 1, GL_FALSE, glm::value_ptr(projection));
     // LookAt camera (position, target/direction, up)
     glm::vec3 cameraPos = glm::vec3(0, 5, 2);
     glm::vec3 cameraFront = glm::vec3(0, 0, 0);
     glm::mat4 view = glm::lookAt(glm::vec3(posCamX, posCamY,
posCamZ), glm::vec3(viewCamX, viewCamY, viewCamZ),
glm::vec3(upCamX, upCamY, upCamZ));

glUniformMatrix4fv(glGetUniformLocation(this->shadowmapShader,
"view"), 1, GL_FALSE, glm::value_ptr(view));

     /*Setting Light Attributes*/

glUniformMatrix4fv(glGetUniformLocation(this->shadowmapShader,
"lightSpaceMatrix"), 1, GL_FALSE,
glm::value_ptr(lightSpaceMatrix));
     glUniform3f(glGetUniformLocation(this->shadowmapShader,
"viewPos"), cameraPos.x, cameraPos.y, cameraPos.z);
     glUniform3f(glGetUniformLocation(this->shadowmapShader,
"lightPos"), 0.0f, 150.0f, 0.0f); //posisi cahaya

     //Configure Shaders

glUniform1i(glGetUniformLocation(this->shadowmapShader,"diffus
eTexture"), 0);
     glUniform1i(glGetUniformLocation(this->shadowmapShader,
"shadowMap"), 1);

     //membuat rumput
     DrawColoredPlane(shadowmapShader);
     //membuat langit
     DrawSky(shadowmapShader);
     //membuat pagar tembok
     DrawPagar(shadowmapShader);
     //membuat pohon
     DrawBatang(shadowmapShader);
     DrawDaun(shadowmapShader);
     //membuat rumput semak
     DrawRumputSemak(shadowmapShader);
     //membuat lampu taman
     DrawPenyanggaTiang(shadowmapShader);
     DrawTiang(shadowmapShader);
     DrawLampu(shadowmapShader);
     //membuat kursi
```

```
	DrawKursi(shadowmapShader);
	//membuat meja
	DrawMeja(shadowmapShader);
	DrawKolam(shadowmapShader);
	DrawBatuKolam(shadowmapShader);

	glDisable(GL_DEPTH_TEST);
}
void Application::BuildColoredPlane()
{
	// Load and create a texture
	glGenTextures(1, &texture2);
	glBindTexture(GL_TEXTURE_2D, texture2);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_REPEAT);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_REPEAT);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);

	int width, height;
	unsigned char* image = SOIL_load_image("Rumput.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
	glGenerateMipmap(GL_TEXTURE_2D);
	SOIL_free_image_data(image);
	glBindTexture(GL_TEXTURE_2D, 0);

	// Build geometry
	GLfloat vertices[] = {
		// format position, tex coords
		// bottom
		-50.0, -0.5, -50.0,  0,  0, 0.0f,  1.0f,  0.0f,
		 50.0, -0.5, -50.0, 50,  0, 0.0f,  1.0f,  0.0f,
		 50.0, -0.5,  50.0, 50, 50, 0.0f,  1.0f,  0.0f,
		-50.0, -0.5,  50.0,  0, 50, 0.0f,  1.0f,  0.0f,
	};


	GLuint indices[] = {
		0,  2,  1,  0,  3,  2,
	};

	glGenVertexArrays(1, &VAO2);
	glGenBuffers(1, &VBO2);
	glGenBuffers(1, &EBO2);

	glBindVertexArray(VAO2);
```

```
	glBindBuffer(GL_ARRAY_BUFFER, VBO2);
	glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO2);
	glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

	// Position attribute
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), 0);
	glEnableVertexAttribArray(0);
	// TexCoord attribute
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);
	// Normal attribute
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);
	glBindVertexArray(0); // Unbind VAO
}

void Application::DrawColoredPlane(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture2);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO2); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(0.2, 0, 1));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(1, 0.1, 1));
	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	glBindVertexArray(0);
}
```

```
void Application::BuildColoredSky()
{
	// load image into texture memory
	// ------------------------------
	// Load and create a texture
	glGenTextures(1, &texture3);
	glBindTexture(GL_TEXTURE_2D, texture3);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
	int width, height;
	unsigned char* image = SOIL_load_image("blueSky.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
	SOIL_free_image_data(image);
	glBindTexture(GL_TEXTURE_2D, 0);

	// set up vertex data (and buffer(s)) and configure
vertex attributes
	//
------------------------------------------------------------
----
	float vertices[] = {
		// format position, tex coords
		// front
		-0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
		0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
		0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
		-0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

		// right
		0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
		0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
		0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
		0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

		// back
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
		0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
		0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
		-0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

		// left
		-0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
		-0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
		-0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
		-0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15
```

```
            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
      glEnableVertexAttribArray(2);
```

```
	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawSky(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture3);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	//depan
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(0, 10, -50));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(200, 100, 0.1));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	//kanan
	for (int i = 0; i < 1; i++) {
```

```
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture3);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(50, 10,
0));
            model2 = glm::scale(model2, glm::vec3(0.1, 100,
300));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //kiri
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture3);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-50, 10,
0));
            model2 = glm::scale(model2, glm::vec3(0.1, 100,
300));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //belakang
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture3);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(0, 10,
50));
            model2 = glm::scale(model2, glm::vec3(200, 100,
0.1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildPagar()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture4);
      glBindTexture(GL_TEXTURE_2D, texture4);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("PagarTembok.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
```

```
            // front
            -0.5, -0.5, 0.5,  0, 0, 0.0f,  0.0f,  1.0f, // 0
             0.5, -0.5, 0.5,  1, 0, 0.0f,  0.0f,  1.0f, // 1
             0.5,  0.5, 0.5,  1, 1, 0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5,  0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
             0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
             0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
             0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
             0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
             0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
             0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
```

```
	glBindVertexArray(VAO7);

	glBindBuffer(GL_ARRAY_BUFFER, VBO7);
	glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
	glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawPagar(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture4);
	glActiveTexture(GL_TEXTURE1);
```

```
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized
	//kiri
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(-25, 0, 0));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(3, 12, 50));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	//kanan
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture4);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(25, 0,
0));
		model2 = glm::scale(model2, glm::vec3(3, 12, 50));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	//depan
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture4);
		glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(0, 0,
-25));
            model2 = glm::scale(model2, glm::vec3(50, 12, 3));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //belakang
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture4);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(0, 0,
25));
            model2 = glm::scale(model2, glm::vec3(50, 12, 3));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}
void Application::BuildBatang()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture5);
      glBindTexture(GL_TEXTURE_2D, texture5);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
```

```
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("batang.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
```

```
          0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
          -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
     };

     unsigned int indices[] = {
          0,  1,  2,  0,  2,  3,   // front
          4,  5,  6,  4,  6,  7,   // right
          8,  9,  10, 8,  10, 11,  // back
          12, 14, 13, 12, 15, 14,  // left
          16, 18, 17, 16, 19, 18,  // upper
          20, 22, 21, 20, 23, 22   // bottom
     };

     glGenVertexArrays(1, &VAO7);
     glGenBuffers(1, &VBO7);
     glGenBuffers(1, &EBO7);
     // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
     glBindVertexArray(VAO7);

     glBindBuffer(GL_ARRAY_BUFFER, VBO7);
     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

     // define position pointer layout 0
     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
     glEnableVertexAttribArray(0);

     // define texcoord pointer layout 1
     glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
     glEnableVertexAttribArray(1);

     // define normal pointer layout 2
     glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
     glEnableVertexAttribArray(2);

     // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
     glBindBuffer(GL_ARRAY_BUFFER, 0);

     // You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
```

```
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawBatang(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture5);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(5, 0, -18));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(0.2, 2, 0.2));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture5);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5, 0,
```

```
-12));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-6));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
0));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
6));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
12));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
18));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
18));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
12));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
6));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture5);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 0,
0));
          model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture5);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 0,
-6));
          model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-12));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture5);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-18));
            model2 = glm::scale(model2, glm::vec3(0.2, 2,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildDaun()
{
      // load image into texture memory
```

```
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture6);
      glBindTexture(GL_TEXTURE_2D, texture6);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("daun.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
```

```
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
      glEnableVertexAttribArray(2);

      // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
```

```
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawDaun(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture6);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(5, 1.5, -18));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(1, 1, 1));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	//Daun Bawah
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture6);
		glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
-12));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
-6));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
```

```
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
0));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
6));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
12));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 1.5,
18));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
18));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
```

```
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
12));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
6));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
0));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
-6));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
-12));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 1.5,
-18));
            model2 = glm::scale(model2, glm::vec3(1, 1, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //Daun Atas
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
```

```
-18));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
-12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
-6));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
0));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
6));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 2.2,
18));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 2.2,
18));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 2.2,
12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture6);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 2.2,
6));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture6);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 2.2,
0));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 2.2,
-6));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 2.2,
-12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture6);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 2.2,
-18));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildRumputSemak()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture11);
      glBindTexture(GL_TEXTURE_2D, texture11);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("RumputSemak.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
```

```
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
```

```
GL_STATIC_DRAW);

     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

     // define position pointer layout 0
     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
     glEnableVertexAttribArray(0);

     // define texcoord pointer layout 1
     glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
     glEnableVertexAttribArray(1);

     // define normal pointer layout 2
     glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
     glEnableVertexAttribArray(2);

     // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
     glBindBuffer(GL_ARRAY_BUFFER, 0);

     // You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
     // VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
     glBindVertexArray(0);

     // remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawRumputSemak(GLuint shaderProgram)
{
     glUseProgram(shaderProgram);

     glActiveTexture(GL_TEXTURE0);
     glBindTexture(GL_TEXTURE_2D, texture11);
     glActiveTexture(GL_TEXTURE1);
     glBindTexture(GL_TEXTURE_2D, depthMap);

     glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
```

so to keep things a bit more organized

```
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(2.5, 0, -16));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(0.5, 1, 7));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture11);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(2.5, 0,
-8));
		model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture11);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
```

```
            model2 = glm::translate(model2, glm::vec3(2.5, 0,
0));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(2.5, 0,
8));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(2.5, 0,
16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-2.5, 0,
16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-2.5, 0,
8));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
```

```
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-2.5, 0,
0));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-2.5, 0,
-8));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture11);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-2.5, 0,
-16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 7));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildPenyanggaTiang()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture9);
      glBindTexture(GL_TEXTURE_2D, texture9);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("hitam.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
```

```
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);
```

```
	glBindBuffer(GL_ARRAY_BUFFER, VBO7);
	glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
	glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawPenyanggaTiang(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture9);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);
```

```
	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(22, 0, -20));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(0.6, 0.6, 0.6));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture9);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(22, 0,
-16));
		model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture9);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
-12));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
-8));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
```

```
            model2 = glm::translate(model2, glm::vec3(22, 0,
-4));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
0));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
4));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
```

```
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
8));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
12));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
16));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 0,
20));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
-20));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
-16));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
```

```
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture9);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(-22, 0,
-12));
		model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture9);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(-22, 0,
-8));
		model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
-4));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
0));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
```

```
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
4));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
8));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
12));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 0,
16));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture9);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
```

```
            model2 = glm::translate(model2, glm::vec3(-22, 0,
20));
            model2 = glm::scale(model2, glm::vec3(0.6, 0.6,
0.6));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildTiang()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture7);
      glBindTexture(GL_TEXTURE_2D, texture7);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("tiang.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
```

```
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);
```

```
	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawTiang(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture7);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(22, 1, -20));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
```

```
     model = glm::scale(model, glm::vec3(0.2, 6, 0.2));


     GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
     glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

     glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

     glBindTexture(GL_TEXTURE_2D, 0);

     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture7);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
-16));
          model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture7);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
-12));
          model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-8));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-4));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
0));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
4));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
8));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
12));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture7);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
16));
          model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture7);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
20));
          model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-20));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-16));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-12));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-8));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-4));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
0));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
```

```
4));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
8));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
12));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
16));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture7);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
20));
            model2 = glm::scale(model2, glm::vec3(0.2, 6,
0.2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildLampu()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture8);
      glBindTexture(GL_TEXTURE_2D, texture8);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("lampu.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
```

```
          -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

          // left
          -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
          -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
          -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
          -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

          // upper
          0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
          -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
          -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
          0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

          // bottom
          -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
          0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
          0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
          -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
     };

     unsigned int indices[] = {
          0,  1,  2,  0,  2,  3,   // front
          4,  5,  6,  4,  6,  7,   // right
          8,  9,  10, 8,  10, 11,  // back
          12, 14, 13, 12, 15, 14,  // left
          16, 18, 17, 16, 19, 18,  // upper
          20, 22, 21, 20, 23, 22   // bottom
     };

     glGenVertexArrays(1, &VAO7);
     glGenBuffers(1, &VBO7);
     glGenBuffers(1, &EBO7);
     // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
     glBindVertexArray(VAO7);

     glBindBuffer(GL_ARRAY_BUFFER, VBO7);
     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

     // define position pointer layout 0
     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
     glEnableVertexAttribArray(0);

     // define texcoord pointer layout 1
```

```
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawLampu(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture8);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(21.65, 3.5,
-20));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(0.5, 0.5, 0.5));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model));

     glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

     glBindTexture(GL_TEXTURE_2D, 0);

     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture8);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.65,
3.5, -16));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture8);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.65,
3.5, -12));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, -8));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, -4));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture8);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 0));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture8);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 4));
          model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 8));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 16));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.65,
3.5, 20));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, -20));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, -16));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
```

```
3.5, -12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, -8));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, -4));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 0));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 4));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 8));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 12));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 16));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture8);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.65,
3.5, 20));
            model2 = glm::scale(model2, glm::vec3(0.5, 0.5,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
	glBindVertexArray(0);
}
void Application::BuildKursi()
{
	// load image into texture memory
	// ------------------------------
	// Load and create a texture
	glGenTextures(1, &texture10);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
	int width, height;
	unsigned char* image = SOIL_load_image("MejaKursi.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
	SOIL_free_image_data(image);
	glBindTexture(GL_TEXTURE_2D, 0);

	// set up vertex data (and buffer(s)) and configure
vertex attributes
	//
------------------------------------------------------------
----
	float vertices[] = {
		// format position, tex coords
		// front
		-0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
		0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
		0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
		-0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

		// right
		0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
		0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
		0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
		0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

		// back
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
		0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
		0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
		-0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

		// left
		-0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
		-0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
		-0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
		-0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15
```

```
            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
```

```
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawKursi(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(21.5, 0, -18));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(1.5, 1.2, 2));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	//dekat lampu
	for (int i = 0; i < 1; i++) {
```

```
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
```

```
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
18));
```

```
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-18));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
```

```
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
```

```
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
18));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //dekat pohon
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
4.5));
```

```
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 0,
-10.5));
          model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 0,
-13.5));
          model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
```

```
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
```

```
-18));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }

     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
-14));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(22, 1,
-10));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
```

```
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
10));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
14));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-14));
```

```
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-10));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
6));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
10));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
```

```
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
14));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildMeja()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture10);
      glBindTexture(GL_TEXTURE_2D, texture10);
```

```
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
	int width, height;
	unsigned char* image = SOIL_load_image("MejaKursi.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
	SOIL_free_image_data(image);
	glBindTexture(GL_TEXTURE_2D, 0);

	// set up vertex data (and buffer(s)) and configure
vertex attributes
	//
------------------------------------------------------------
----
	float vertices[] = {
		// format position, tex coords
		// front
		-0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
		0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
		0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
		-0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3
		// right
		0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
		0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
		0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
		0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

		// back
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
		0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
		0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
		-0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

		// left
		-0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
		-0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
		-0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
		-0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

		// upper
		0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
		-0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
		-0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
		0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19
		// bottom
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
		0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
		0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
```

```
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
      glEnableVertexAttribArray(2);

      // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
      glBindBuffer(GL_ARRAY_BUFFER, 0);

      // You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
      // VAOs requires a call to glBindVertexArray anyways so
```

```
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
     glBindVertexArray(0);

     // remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawMeja(GLuint shaderProgram)
{
     glUseProgram(shaderProgram);

     glActiveTexture(GL_TEXTURE0);
     glBindTexture(GL_TEXTURE_2D, texture10);
     glActiveTexture(GL_TEXTURE1);
     glBindTexture(GL_TEXTURE_2D, depthMap);

     glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

     glm::mat4 model;
     model = glm::translate(model, glm::vec3(-5, 0.6,15));
     //model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
     model = glm::scale(model, glm::vec3(2, 0.2, 0.5));

     GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
     glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

     glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

     glBindTexture(GL_TEXTURE_2D, 0);
     //Bagian Meja
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5, 0.6,
9));
          model2 = glm::scale(model2, glm::vec3(2, 0.2,
```

```
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5,
0.6,3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-9));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-15));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
15));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
9));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
```

```
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5, 0.6,
3));
		model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5, 0.6,
-3));
		model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);
```

```
          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(5, 0.6,
-9));
          model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(5, 0.6,
-15));
          model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     //bagian kaki kanan meja
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
```

```
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
15));
```

```
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
-3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //bagian kaki kiri meja
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-3));
          model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-9));
          model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
-3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
```

```
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildKolam()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture12);
      glBindTexture(GL_TEXTURE_2D, texture12);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("Air.png", &width,
&height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
```

```
          0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

          // back
          -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
          0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
          0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
          -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

          // left
          -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
          -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
          -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
          -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

          // upper
          0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
          -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
          -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
          0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

          // bottom
          -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
          0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
          0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
          -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
     };

     unsigned int indices[] = {
          0,  1,  2,  0,  2,  3,   // front
          4,  5,  6,  4,  6,  7,   // right
          8,  9,  10, 8,  10, 11,  // back
          12, 14, 13, 12, 15, 14,  // left
          16, 18, 17, 16, 19, 18,  // upper
          20, 22, 21, 20, 23, 22   // bottom
     };

     glGenVertexArrays(1, &VAO7);
     glGenBuffers(1, &VBO7);
     glGenBuffers(1, &EBO7);
     // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
     glBindVertexArray(VAO7);

     glBindBuffer(GL_ARRAY_BUFFER, VBO7);
     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);
```

```
	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawKolam(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture12);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized
	//kanan
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(13, 0, -16));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(5, 1.3, 10));
```

```
	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture12);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(-13, 0,
16));
		model2 = glm::scale(model2, glm::vec3(5, 1.3, 10));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	glBindVertexArray(0);
}

void Application::BuildKursi()
{
	// load image into texture memory
	// ------------------------------
	// Load and create a texture
	glGenTextures(1, &texture10);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
	int width, height;
	unsigned char* image = SOIL_load_image("MejaKursi.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
```

```
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
```

```
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
      glEnableVertexAttribArray(2);

      // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
      glBindBuffer(GL_ARRAY_BUFFER, 0);

      // You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
      // VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
      glBindVertexArray(0);

      // remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
```

```
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawKursi(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(21.5, 0, -18));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(1.5, 1.2, 2));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);

	//dekat lampu
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(21.5, 0,
-14));
		model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
-2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.5, 0,
6));
          model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(21.5, 0,
10));
          model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(21.5, 0,
18));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-18));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
-6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
```

```
-2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
2));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
6));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
10));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
14));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-21.5, 0,
18));
            model2 = glm::scale(model2, glm::vec3(1.5, 1.2,
2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //dekat pohon
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
```

```
-4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0,
-16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
```

```
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5, 0,
16.5));
		model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5, 0,
13.5));
		model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
```

```
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-1.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-4.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-7.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-10.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-13.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
```

```
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0,
-16.5));
            model2 = glm::scale(model2, glm::vec3(2, 1.2, 1));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }

      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
```

```
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-14));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-10));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
-2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
10));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
14));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(22, 1,
18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-18));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
-14));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
-10));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
-6));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
```

```
-2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
2));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-22, 1,
6));
            model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
```

```
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
10));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-22, 1,
14));
          model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
```

```
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(-22, 1,
18));
		model2 = glm::scale(model2, glm::vec3(0.5, 1, 2));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	glBindVertexArray(0);
}

void Application::BuildMeja()
{
	// load image into texture memory
	// ------------------------------
	// Load and create a texture
	glGenTextures(1, &texture10);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
	int width, height;
	unsigned char* image = SOIL_load_image("MejaKursi.png",
&width, &height, 0, SOIL_LOAD_RGBA);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
	SOIL_free_image_data(image);
	glBindTexture(GL_TEXTURE_2D, 0);

	// set up vertex data (and buffer(s)) and configure
vertex attributes
	//
------------------------------------------------------------
----
```

```
	float vertices[] = {
		// format position, tex coords
		// front
		-0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
		0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
		0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
		-0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3
		// right
		0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
		0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
		0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
		0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

		// back
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
		0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
		0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
		-0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

		// left
		-0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
		-0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
		-0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
		-0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

		// upper
		0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
		-0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
		-0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
		0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19
		// bottom
		-0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
		0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
		0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
		-0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
	};

	unsigned int indices[] = {
		0,  1,  2,  0,  2,  3,   // front
		4,  5,  6,  4,  6,  7,   // right
		8,  9,  10, 8,  10, 11,  // back
		12, 14, 13, 12, 15, 14,  // left
		16, 18, 17, 16, 19, 18,  // upper
		20, 22, 21, 20, 23, 22   // bottom
	};

	glGenVertexArrays(1, &VAO7);
	glGenBuffers(1, &VBO7);
	glGenBuffers(1, &EBO7);
	// bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
```

```
	glBindVertexArray(VAO7);

	glBindBuffer(GL_ARRAY_BUFFER, VBO7);
	glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
	glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawMeja(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture10);
	glActiveTexture(GL_TEXTURE1);
```

```
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized

	glm::mat4 model;
	model = glm::translate(model, glm::vec3(-5, 0.6,15));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(2, 0.2, 0.5));

	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	//Bagian Meja
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(-5, 0.6,
9));
		model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5,
0.6,3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-9));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5, 0.6,
-15));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
15));
```

```
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
9));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
-3));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
-9));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5, 0.6,
-15));
            model2 = glm::scale(model2, glm::vec3(2, 0.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //bagian kaki kanan meja
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
```

```
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
```

```
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-3));
          model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);

          glActiveTexture(GL_TEXTURE0);
          glBindTexture(GL_TEXTURE_2D, texture10);
          glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

          glBindVertexArray(VAO7);

          glm::mat4 model2;
          model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-9));
          model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

          glGetUniformLocation(shaderProgram, "model");
          glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

          glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
          glBindTexture(GL_TEXTURE_2D, 0);
     }
     for (int i = 0; i < 1; i++) {
          glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-4.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
```

```
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);
```

```
            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
-3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(4.5, 0,
```

```
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      //bagian kaki kiri meja
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
```

```
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-3));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
```

```
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-5.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
```

```
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5.5, 0,
3));
		model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture10);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(5.5, 0,
-3));
		model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);
```

```
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
-9));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture10);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(5.5, 0,
-15));
            model2 = glm::scale(model2, glm::vec3(0.5, 1.2,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildKolam()
{
```

```
     // load image into texture memory
     // ------------------------------
     // Load and create a texture
     glGenTextures(1, &texture12);
     glBindTexture(GL_TEXTURE_2D, texture12);
     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
     int width, height;
     unsigned char* image = SOIL_load_image("Air.png", &width,
&height, 0, SOIL_LOAD_RGBA);
     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
     SOIL_free_image_data(image);
     glBindTexture(GL_TEXTURE_2D, 0);

     // set up vertex data (and buffer(s)) and configure
vertex attributes
     //
------------------------------------------------------------
----
     float vertices[] = {
          // format position, tex coords
          // front
          -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
          0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
          0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
          -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3

          // right
          0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
          0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
          0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
          0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

          // back
          -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
          0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
          0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
          -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

          // left
          -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
          -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
          -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
          -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

          // upper
          0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
          -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
```

```
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);

      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

      // define position pointer layout 0
      glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
      glEnableVertexAttribArray(0);

      // define texcoord pointer layout 1
      glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
      glEnableVertexAttribArray(1);

      // define normal pointer layout 2
      glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
      glEnableVertexAttribArray(2);

      // note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
```

```
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawKolam(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture12);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized
	//kanan
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(13, 0, -16));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(5, 1.3, 10));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture12);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);
```

```
            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-13, 0,
16));
            model2 = glm::scale(model2, glm::vec3(5, 1.3, 10));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::BuildBatuKolam()
{
      // load image into texture memory
      // ------------------------------
      // Load and create a texture
      glGenTextures(1, &texture13);
      glBindTexture(GL_TEXTURE_2D, texture13);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
      int width, height;
      unsigned char* image = SOIL_load_image("BatuKolam.png",
&width, &height, 0, SOIL_LOAD_RGBA);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, image);
      SOIL_free_image_data(image);
      glBindTexture(GL_TEXTURE_2D, 0);

      // set up vertex data (and buffer(s)) and configure
vertex attributes
      //
------------------------------------------------------------
----
      float vertices[] = {
            // format position, tex coords
            // front
            -0.5, -0.5, 0.5, 0, 0, 0.0f,  0.0f,  1.0f, // 0
            0.5, -0.5, 0.5, 1, 0,  0.0f,  0.0f,  1.0f, // 1
            0.5,  0.5, 0.5, 1, 1,  0.0f,  0.0f,  1.0f, // 2
            -0.5,  0.5, 0.5, 0, 1, 0.0f,  0.0f,  1.0f, // 3
```

```
            // right
            0.5,  0.5,  0.5, 0, 0, 1.0f,  0.0f,  0.0f, // 4
            0.5,  0.5, -0.5, 1, 0, 1.0f,  0.0f,  0.0f, // 5
            0.5, -0.5, -0.5, 1, 1, 1.0f,  0.0f,  0.0f, // 6
            0.5, -0.5,  0.5, 0, 1, 1.0f,  0.0f,  0.0f, // 7

            // back
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  0.0f,  -1.0f, // 8
            0.5,  -0.5, -0.5, 1, 0, 0.0f,  0.0f,  -1.0f, // 9
            0.5,   0.5, -0.5, 1, 1, 0.0f,  0.0f,  -1.0f, // 10
            -0.5,  0.5, -0.5, 0, 1, 0.0f,  0.0f,  -1.0f, // 11

            // left
            -0.5, -0.5, -0.5, 0, 0, -1.0f,  0.0f,  0.0f, // 12
            -0.5, -0.5,  0.5, 1, 0, -1.0f,  0.0f,  0.0f, // 13
            -0.5,  0.5,  0.5, 1, 1, -1.0f,  0.0f,  0.0f, // 14
            -0.5,  0.5, -0.5, 0, 1, -1.0f,  0.0f,  0.0f, // 15

            // upper
            0.5, 0.5,  0.5, 0, 0,   0.0f,  1.0f,  0.0f, // 16
            -0.5, 0.5, 0.5, 1, 0,   0.0f,  1.0f,  0.0f, // 17
            -0.5, 0.5, -0.5, 1, 1,  0.0f,  1.0f,  0.0f, // 18
            0.5, 0.5, -0.5, 0, 1,   0.0f,  1.0f,  0.0f, // 19

            // bottom
            -0.5, -0.5, -0.5, 0, 0, 0.0f,  -1.0f,  0.0f, // 20
            0.5, -0.5, -0.5, 1, 0,  0.0f,  -1.0f,  0.0f, // 21
            0.5, -0.5,  0.5, 1, 1,  0.0f,  -1.0f,  0.0f, // 22
            -0.5, -0.5,  0.5, 0, 1, 0.0f,  -1.0f,  0.0f, // 23
      };

      unsigned int indices[] = {
            0,  1,  2,  0,  2,  3,   // front
            4,  5,  6,  4,  6,  7,   // right
            8,  9,  10, 8,  10, 11,  // back
            12, 14, 13, 12, 15, 14,  // left
            16, 18, 17, 16, 19, 18,  // upper
            20, 22, 21, 20, 23, 22   // bottom
      };

      glGenVertexArrays(1, &VAO7);
      glGenBuffers(1, &VBO7);
      glGenBuffers(1, &EBO7);
      // bind the Vertex Array Object first, then bind and set
vertex buffer(s), and then configure vertex attributes(s).
      glBindVertexArray(VAO7);

      glBindBuffer(GL_ARRAY_BUFFER, VBO7);
      glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);
```

```
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO7);
	glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices),
indices, GL_STATIC_DRAW);

	// define position pointer layout 0
	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(0 * sizeof(GLfloat)));
	glEnableVertexAttribArray(0);

	// define texcoord pointer layout 1
	glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(3 * sizeof(GLfloat)));
	glEnableVertexAttribArray(1);

	// define normal pointer layout 2
	glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (GLvoid*)(5 * sizeof(GLfloat)));
	glEnableVertexAttribArray(2);

	// note that this is allowed, the call to
glVertexAttribPointer registered VBO as the vertex attribute's
bound vertex buffer object so afterwards we can safely unbind
	glBindBuffer(GL_ARRAY_BUFFER, 0);

	// You can unbind the VAO afterwards so other VAO calls
won't accidentally modify this VAO, but this rarely happens.
Modifying other
	// VAOs requires a call to glBindVertexArray anyways so
we generally don't unbind VAOs (nor VBOs) when it's not
directly necessary.
	glBindVertexArray(0);

	// remember: do NOT unbind the EBO while a VAO is active
as the bound element buffer object IS stored in the VAO; keep
the EBO bound.
	glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

void Application::DrawBatuKolam(GLuint shaderProgram)
{
	glUseProgram(shaderProgram);

	glActiveTexture(GL_TEXTURE0);
	glBindTexture(GL_TEXTURE_2D, texture13);
	glActiveTexture(GL_TEXTURE1);
	glBindTexture(GL_TEXTURE_2D, depthMap);

	glBindVertexArray(VAO7); // seeing as we only have a
single VAO there's no need to bind it every time, but we'll do
so to keep things a bit more organized
```

```
	glm::mat4 model;
	model = glm::translate(model, glm::vec3(13, 0.5, -21));
	//model = glm::rotate(model,angle, glm::vec3(0, 1, 0));
	model = glm::scale(model, glm::vec3(4.5, 1, 0.5));


	GLint modelLoc = glGetUniformLocation(shaderProgram,
"model");
	glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));

	glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);

	glBindTexture(GL_TEXTURE_2D, 0);
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture13);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(13, 0.5,
-11));
		model2 = glm::scale(model2, glm::vec3(4.5, 1,
0.5));

		glGetUniformLocation(shaderProgram, "model");
		glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

		glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
		glBindTexture(GL_TEXTURE_2D, 0);
	}
	for (int i = 0; i < 1; i++) {
		glUseProgram(shaderProgram);

		glActiveTexture(GL_TEXTURE0);
		glBindTexture(GL_TEXTURE_2D, texture13);
		glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

		glBindVertexArray(VAO7);

		glm::mat4 model2;
		model2 = glm::translate(model2, glm::vec3(15.5,
0.5, -16));
```

```
            model2 = glm::scale(model2, glm::vec3(0.5, 1,
10.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture13);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(10.5,
0.5, -16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1,
10.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture13);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-13, 0.5,
21));
            model2 = glm::scale(model2, glm::vec3(4.5, 1,
0.5));
```

```
            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture13);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-13, 0.5,
11));
            model2 = glm::scale(model2, glm::vec3(4.5, 1,
0.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture13);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-15.5,
0.5, 16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1,
10.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));
```

```
            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      for (int i = 0; i < 1; i++) {
            glUseProgram(shaderProgram);

            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, texture13);
            glUniform1i(glGetUniformLocation(shaderProgram,
"ourTexture"), 0);

            glBindVertexArray(VAO7);

            glm::mat4 model2;
            model2 = glm::translate(model2, glm::vec3(-10.5,
0.5, 16));
            model2 = glm::scale(model2, glm::vec3(0.5, 1,
10.5));

            glGetUniformLocation(shaderProgram, "model");
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model2));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT,
0);
            glBindTexture(GL_TEXTURE_2D, 0);
      }
      glBindVertexArray(0);
}

void Application::InitCamera()
{
      posCamX = 0.0f;
      posCamY = 3.0f;
      posCamZ = 8.0f;
      viewCamX = 0.0f;
      viewCamY = 1.0f;
      viewCamZ = 0.0f;
      upCamX = 0.0f;
      upCamY = 2.0f;
      upCamZ = 0.0f;
      CAMERA_SPEED = 0.001f;
      fovy = 45.0f;
      glfwSetInputMode(this->window, GLFW_CURSOR,
GLFW_CURSOR_DISABLED);
}

void Application::MoveCamera(float speed)
{
```

```
	float x = viewCamX - posCamX;
	float z = viewCamZ - posCamZ;
	// forward positive cameraspeed and backward negative
-cameraspeed.
	posCamX = posCamX + x * speed;
	posCamZ = posCamZ + z * speed;
	viewCamX = viewCamX + x * speed;
	viewCamZ = viewCamZ + z * speed;
}

void Application::StrafeCamera(float speed)
{
	float x = viewCamX - posCamX;
	float z = viewCamZ - posCamZ;
	float orthoX = -z;
	float orthoZ = x;

	// left positive cameraspeed and right negative
-cameraspeed.
	posCamX = posCamX + orthoX * speed;
	posCamZ = posCamZ + orthoZ * speed;
	viewCamX = viewCamX + orthoX * speed;
	viewCamZ = viewCamZ + orthoZ * speed;
}

void Application::RotateCamera(float speed)
{
	float x = viewCamX - posCamX;
	float z = viewCamZ - posCamZ;
	viewCamZ = (float)(posCamZ + glm::sin(speed) * x +
glm::cos(speed) * z);
	viewCamX = (float)(posCamX + glm::cos(speed) * x -
glm::sin(speed) * z);
}

int main(int argc, char** argv) {
	Application app = Application();
	app.Start("Taman 3D Project", 800, 600, false, false);
}
```

- Aplication.h

```
#pragma once

#include "RenderEngine.h"
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <glm/gtx/vector_angle.hpp>
```

```
#include <SOIL/SOIL.h>

class Application :
    public RenderEngine
{
public:
    Application();
    ~Application();
private:
     GLuint
          //Buat Rumput
          VBO2, VAO2, EBO2, texture2,
          //buat sky, pagar, batang, daun, rumput semak,
tiang, lampu, penyangga, meja, kursi kolam, batu kolam
          VBO7, VAO7, EBO7, texture3, texture4, texture5,
texture6, texture7, texture8, texture9, texture10,
          texture11, texture12, texture13, shadowmapShader,
depthmapShader,
          //shadow
          depthMapFBO, depthMap;

     float angle = 0;
     float posisiX = 0;
     float posisiY = 0;
     float posisiZ = 0;
     float speed = 0.7;
     float viewCamX, viewCamY, viewCamZ, upCamX, upCamY,
upCamZ, posCamX, posCamY, posCamZ, CAMERA_SPEED, fovy;
     virtual void Init();
     virtual void DeInit();
     virtual void Update(double deltaTime);
     virtual void Render();
     virtual void ProcessInput(GLFWwindow* window);
     void MoveCamera(float speed);
     void StrafeCamera(float speed);
     void RotateCamera(float speed);
     void InitCamera();
     void BuildColoredPlane();
     void BuildColoredSky();
     void BuildPagar();
     void BuildBatang();
     void BuildDaun();
     void BuildRumputSemak();
     void BuildPenyanggaTiang();
     void BuildTiang();
     void BuildLampu();
     void BuildKursi();
     void BuildMeja();
     void BuildKolam();
     void BuildBatuKolam();
```

```
	void DrawColoredPlane(GLuint shaderProgram);
	void DrawSky(GLuint shaderProgram);
	void DrawPagar(GLuint shaderProgram);
	void DrawBatang(GLuint shaderProgram);
	void DrawDaun(GLuint shaderProgram);
	void DrawRumputSemak(GLuint shaderProgram);
	void DrawPenyanggaTiang(GLuint shaderProgram);
	void DrawTiang(GLuint shaderProgram);
	void DrawLampu(GLuint shaderProgram);
	void DrawKursi(GLuint shaderProgram);
	void DrawMeja(GLuint shaderProgram);
	void DrawKolam(GLuint shaderProgram);
	void DrawBatuKolam(GLuint shaderProgram);
};
```

- RenderEngine.cpp

```
#include "RenderEngine.h"
RenderEngine::RenderEngine() {
}


RenderEngine::~RenderEngine() {
	glfwDestroyWindow(window);
}



void RenderEngine::Start(const char* title, unsigned int
width, unsigned int height, bool vsync, bool fullscreen) {

	// set app configuration
	this->screenHeight = height;
	this->screenWidth = width;

	// glfw: initialize and configure
	// ------------------------------
	glfwInit();
	glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
	glfwWindowHint(GLFW_OPENGL_PROFILE,
GLFW_OPENGL_CORE_PROFILE);

	// glfw window creation
	// --------------------
	GLFWmonitor* monitor = glfwGetPrimaryMonitor();
	window = glfwCreateWindow(this->screenWidth,
this->screenHeight, title, fullscreen ? monitor : NULL, NULL);
```

```
      if (window == NULL)
      {
            Err("Failed to create GLFW window");
      }

      // set window position on center of screen
      // ---------------------------------------
      if (!fullscreen) {
            const GLFWvidmode* mode =
glfwGetVideoMode(monitor);
            glfwSetWindowPos(window, mode->width / 4,
mode->height / 4);
      }

      // set opengl context
      // ------------------
      glfwMakeContextCurrent(window);

      // glad: load all OpenGL function pointers
      // ---------------------------------------
      if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
      {
            Err("Failed to initialize GLAD");
      }

      // set vsync
      // ---------
      glfwSwapInterval(vsync ? 1 : 0);

      // user defined function
      // ---------------------
      Init();

      lastFrame = glfwGetTime() * 1000;

      // render loop
      // -----------
      while (!glfwWindowShouldClose(window)) {
            // Calculate framerate and frametime
            double deltaTime = GetDeltaTime();
            GetFPS();

            // user defined function
            // ---------------------
            ProcessInput(window);
            Update(deltaTime);
            Render();

            // glfw: swap buffers and poll IO events (keys
pressed/released, mouse moved etc.)
            //
```

```
-----------------------------------------------------------------
-----------------
            glfwSwapBuffers(window);
            glfwPollEvents();

            //Debug print framerate
            PrintFrameRate();
      }

      // user defined function
      // ---------------------
      DeInit();

      // glfw: terminate, clearing all previously allocated
GLFW resources.
      //
-----------------------------------------------------------------
----
      glfwTerminate();
}

double RenderEngine::GetDeltaTime()
{

      double time = glfwGetTime() * 1000;
      double delta = time - lastFrame;
      lastFrame = time;

      return delta;
}

void RenderEngine::GetFPS()
{
      if (glfwGetTime() * 1000 - last > 1000) {
            fps = _fps;
            _fps = 0;
            last += 1000;
      }
      _fps++;
}


//Prints out an error message and exits the game
void RenderEngine::Err(std::string errorString)
{
      std::cout << errorString << std::endl;
      glfwTerminate();
      exit(1);
}

static int frameCounter = 0;
```

```
void RenderEngine::PrintFrameRate() {
	//print only once every 60 frames
	frameCounter++;
	if (frameCounter == 60) {
		std::cout << "FPS: " << fps << std::endl;
		frameCounter = 0;
	}
}

void RenderEngine::CheckShaderErrors(GLuint shader,
std::string type)
{
	GLint success;
	GLchar infoLog[1024];
	if (type != "PROGRAM")
	{
		glGetShaderiv(shader, GL_COMPILE_STATUS, &success);
		if (!success)
		{
			glGetShaderInfoLog(shader, 1024, NULL,
infoLog);
			Err("| ERROR::::SHADER-COMPILATION-ERROR of
type: " + type + "|\n" + infoLog + "\n| --
---------------------------------------------------- -- |");
		}
	}
	else
	{
		glGetProgramiv(shader, GL_LINK_STATUS, &success);
		if (!success)
		{
			glGetProgramInfoLog(shader, 1024, NULL,
infoLog);
			Err("| ERROR::::PROGRAM-LINKING-ERROR of type:
" + type + "|\n" + infoLog + "\n| --
---------------------------------------------------- -- |");
		}
	}
}

GLuint RenderEngine::BuildShader(const char* vertexPath, const
char* fragmentPath, const char* geometryPath)
{
	// 1. Retrieve the vertex/fragment source code from
filePath
	std::string vertexCode, fragmentCode, geometryCode;
	std::ifstream vShaderFile, fShaderFile, gShaderFile;
	// ensures ifstream objects can throw exceptions:
	vShaderFile.exceptions(std::ifstream::failbit |
std::ifstream::badbit);
	fShaderFile.exceptions(std::ifstream::failbit |
```

```
std::ifstream::badbit);
     gShaderFile.exceptions(std::ifstream::failbit |
std::ifstream::badbit);
     try
     {
          // Open files
          vShaderFile.open(vertexPath);
          fShaderFile.open(fragmentPath);
          std::stringstream vShaderStream, fShaderStream;
          // Read file's buffer contents into streams
          vShaderStream << vShaderFile.rdbuf();
          fShaderStream << fShaderFile.rdbuf();
          // close file handlers
          vShaderFile.close();
          fShaderFile.close();
          // Convert stream into string
          vertexCode = vShaderStream.str();
          fragmentCode = fShaderStream.str();
          // If geometry shader path is present, also load a
geometry shader
          if (geometryPath != nullptr)
          {
               gShaderFile.open(geometryPath);
               std::stringstream gShaderStream;
               gShaderStream << gShaderFile.rdbuf();
               gShaderFile.close();
               geometryCode = gShaderStream.str();
          }
     }
     catch (std::ifstream::failure e)
     {
          Err("ERROR::SHADER::FILE_NOT_SUCCESFULLY_READ");
     }
     const GLchar* vShaderCode = vertexCode.c_str();
     const GLchar * fShaderCode = fragmentCode.c_str();
     // 2. Compile shaders
     GLuint vertex, fragment;

     // Vertex Shader
     vertex = glCreateShader(GL_VERTEX_SHADER);
     glShaderSource(vertex, 1, &vShaderCode, NULL);
     glCompileShader(vertex);
     CheckShaderErrors(vertex, "VERTEX");
     // Fragment Shader
     fragment = glCreateShader(GL_FRAGMENT_SHADER);
     glShaderSource(fragment, 1, &fShaderCode, NULL);
     glCompileShader(fragment);
     CheckShaderErrors(fragment, "FRAGMENT");
     // If geometry shader is given, compile geometry shader
     GLuint geometry;
     if (geometryPath != nullptr)
```

```
	{
		const GLchar * gShaderCode = geometryCode.c_str();
		geometry = glCreateShader(GL_GEOMETRY_SHADER);
		glShaderSource(geometry, 1, &gShaderCode, NULL);
		glCompileShader(geometry);
		CheckShaderErrors(geometry, "GEOMETRY");
	}
	// Shader Program
	GLuint program = glCreateProgram();
	glAttachShader(program, vertex);
	glAttachShader(program, fragment);
	if (geometryPath != nullptr)
		glAttachShader(program, geometry);
	glLinkProgram(program);
	CheckShaderErrors(program, "PROGRAM");
	// Delete the shaders as they're linked into our program
now and no longer necessery
	glDeleteShader(vertex);
	glDeleteShader(fragment);
	if (geometryPath != nullptr)
		glDeleteShader(geometry);
	return program;

}

void RenderEngine::UseShader(GLuint program)
{
	// Uses the current shader
	glUseProgram(program);
}
```

- RenderEngine.h

```
#pragma once
#pragma once

#include <GLAD/glad.h>
#include <GLFW/glfw3.h>
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>


class RenderEngine
{
public:
	RenderEngine();
	~RenderEngine();
	void Start(const char* title, unsigned int width,
```

```
unsigned int height, bool vsync, bool fullscreen);
protected:
     unsigned int screenWidth, screenHeight, last = 0, _fps =
0, fps = 0;
     double lastFrame = 0;
     GLFWwindow* window;

     virtual void Init() = 0;
     virtual void DeInit() = 0;
     virtual void Update(double deltaTime) = 0;
     virtual void Render() = 0;
     virtual void ProcessInput(GLFWwindow *window) = 0;

     double GetDeltaTime();
     void GetFPS();
     void Err(std::string errorString);
     void PrintFrameRate();
     void CheckShaderErrors(GLuint shader, std::string type);
     GLuint BuildShader(const char* vertexPath, const char*
fragmentPath, const char* geometryPath);
     void UseShader(GLuint program);
};
```

- depthMap.frag

```
#version 330 core

struct Material {
    sampler2D diffuse;
    sampler2D specular;
    float shininess;
};

uniform Material material;

void main()
{
    //gl_FragDepth = gl_FragCoord.z; // no need to set depth
explicitly
}
```

- depthMap.vert

```
#version 330 core
layout (location = 0) in vec3 aPos;

uniform mat4 lightSpaceMatrix;
uniform mat4 model;

void main()
```

```
{
    gl_Position = lightSpaceMatrix * model * vec4(aPos, 1.0);
}
```

- FragmentShader.frag

```
#version 330 core

in vec2 TexCoord;
out vec4 FragColor;

// Texture samplers
uniform sampler2D ourTexture;

void main()
{
    FragColor = texture(ourTexture, TexCoord);
}
```

- VertexShader.vert

```
#version 330
layout (location = 0) in vec3 position;
layout (location = 1) in vec2 texCoord;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

out vec2 TexCoord;

void main()
{
     gl_Position = projection * view * model * vec4(position,
1.0f);

     // We swap the y-axis by substracing our coordinates from
1. This is done because most images have the top y-axis
inversed with OpenGL's top y-axis.
     TexCoord = vec2(texCoord.x, 1.0 - texCoord.y);
}
```

- ShadowMapping.frag

```
#version 330 core
out vec4 FragColor;

in VS_OUT {
    vec3 FragPos;
    vec3 Normal;
```

```
    vec2 TexCoords;
    vec4 FragPosLightSpace;
} fs_in;

uniform sampler2D diffuseTexture;
uniform sampler2D shadowMap;

uniform vec3 lightPos;
uniform vec3 viewPos;

float ShadowCalculation(vec4 fragPosLightSpace)
{
    // perform perspective divide
    vec3 projCoords = fragPosLightSpace.xyz /
fragPosLightSpace.w;
    // transform to [0,1] range
    projCoords = projCoords * 0.5 + 0.5;
    // get closest depth value from light's perspective (using
[0,1] range fragPosLight as coords)
    float closestDepth = texture(shadowMap, projCoords.xy).r;
    // get depth of current fragment from light's perspective
    float currentDepth = projCoords.z;
    // calculate bias (based on depth map resolution and
slope)
    vec3 normal = normalize(fs_in.Normal);
    vec3 lightDir = normalize(lightPos - fs_in.FragPos);
    float bias = max(0.05 * (1.0 - dot(normal, lightDir)),
0.005);
    // check whether current frag pos is in shadow
    float shadow = currentDepth - bias > closestDepth  ? 1.0 :
0.0;
    return shadow;
}

void main()
{
    vec3 color = texture(diffuseTexture, fs_in.TexCoords).rgb;
    vec3 normal = normalize(fs_in.Normal);
    vec3 lightColor = vec3(0.5);
    // ambient
    vec3 ambient = 0.3 * color;

    // diffuse
    vec3 lightDir = normalize(lightPos - fs_in.FragPos);
    float diff = max(dot(lightDir, normal), 0.0);
    vec3 diffuse = diff * lightColor;

     // specular
    vec3 viewDir = normalize(viewPos - fs_in.FragPos);
    vec3 reflectDir = reflect(-lightDir, normal);
    float spec = 0.0;
```

```
    spec = pow(max(dot(viewDir, reflectDir), 0.0), 64.0);
    vec3 specular = spec * lightColor;

     // calculate shadow
    float shadow = ShadowCalculation(fs_in.FragPosLightSpace);

     vec3 lighting = (ambient + (1.0 - shadow) * (diffuse +
specular)) * color;

    FragColor = vec4(lighting, 1.0);
}
```

- ShadowMapping.vert

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec2 aTexCoords;
layout (location = 2) in vec3 aNormal;

//out vec2 TexCoords;

out VS_OUT {
    vec3 FragPos;
    vec3 Normal;
    vec2 TexCoords;
    vec4 FragPosLightSpace;
} vs_out;

uniform mat4 projection;
uniform mat4 view;
uniform mat4 model;
uniform mat4 lightSpaceMatrix;

void main()
{
    vs_out.FragPos = vec3(model * vec4(aPos, 1.0));
    vs_out.Normal = transpose(inverse(mat3(model))) * aNormal;
    vs_out.TexCoords = aTexCoords;
    vs_out.FragPosLightSpace = lightSpaceMatrix *
vec4(vs_out.FragPos, 1.0);
    gl_Position = projection * view * model * vec4(aPos, 1.0);
}
```