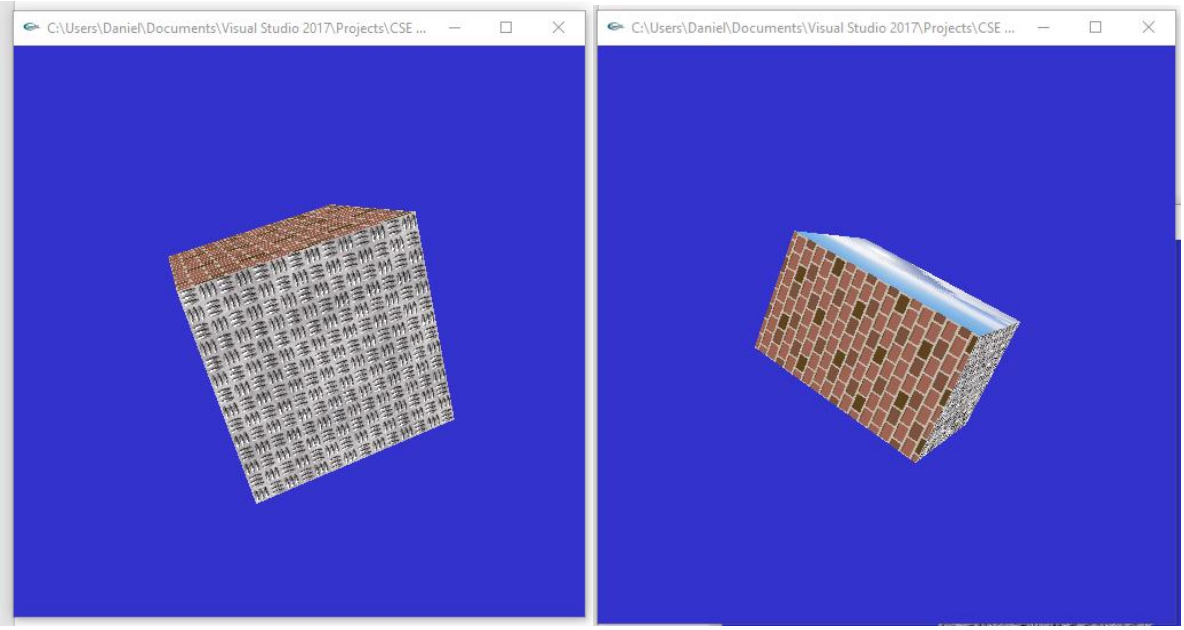
Daniel Meyer

CSE 520

Tong Yu

Homework 3 Report

Part 1 (success):




```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

#define GLEW_STATIC 1
#include <GL/glut.h>
#include <SOIL/SOIL.h>

int texImageWidth;
int texImageHeight;
int window;
static GLuint texName[6];             //texture names
int anglex = 0, angley = 0, anglez = 0;          //rotation angles
float scale = 1;
int timeCounter = 0;  // time counter
int time_interval = 10;     // how much time has to be before action is
done

//images for texture maps for 6 faces of cube
```

```
char maps[][20] = { "floor.png",  "player.png",  "right.png",
"stone.png",
            "wall.png", "cubemap_dn.png" };

//load texture image
GLubyte *makeTexImage(char *loadfile)
{
      int i, j, c, width, height;
      GLubyte *texImage;

      /*
        Only works for .png or .tif images.  NULL is returned if errors
occurred.
        loadImageRGA() is from imageio library downloaded from
Internet.
      */
      //texImage = loadImageRGBA((char *)loadfile, &width, &height);
      texImage = SOIL_load_image(loadfile, &width, &height, 0,
SOIL_LOAD_RGBA);
      texImageWidth = width;
      texImageHeight = height;

      return texImage;
}

void init(void)
{
      glClearColor(0.2, 0.2, 0.8, 0.0);
      glShadeModel(GL_FLAT);

      glEnable(GL_DEPTH_TEST);

      glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
      //texName is global
      glGenTextures(6, texName);
      for (int i = 0; i < 6; ++i)
      {
            GLubyte *texImage = makeTexImage(maps[i]);
            if (!texImage) {
                  printf("\nError reading %s \n", maps[i]);
                  continue;
            }
            glBindTexture(GL_TEXTURE_2D, texName[i]);        //now we
work on texName
            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_MAG_FILTER,
GL_NEAREST);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
            glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, texImageWidth,
                  texImageHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE,
texImage);

            delete texImage;                        //free memory holding
texture image
      }
}

void display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glEnable(GL_TEXTURE_2D);
      glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
      float x0 = -1.0, y0 = -1, x1 = 1, y1 = 1, z0 = 1;
      float face[6][4][3] = { {{x0, y0, z0}, {x1, y0, z0}, {x1, y1,
z0}, {x0, y1, z0}},   //front
                  {{x0, y1, -z0}, {x1, y1, -z0}, {x1, y0, -z0}, {x0,
y0, -z0}},       //back
            {{x1, y0, z0}, {x1, y0, -z0}, {x1, y1, -z0}, {x1, y1, z0}},
      //right
            {{x0, y0, z0}, {x0, y1, z0}, {x0, y1, -z0}, {x0, y0, -z0}},
      //left
            {{x0, y1, z0}, {x1, y1, z0}, {x1, y1, -z0}, {x0, y1, -z0}},
      //top
            {{x0, y0, z0}, {x0, y0, -z0}, {x1, y0, -z0}, {x1, y0, z0}}
      //bottom
      };
      glEnable(GL_CULL_FACE);
      glCullFace(GL_BACK);

      glPushMatrix();
      glRotatef(anglex, 1.0, 0.0, 0.0);               //rotate the
cube along x-axis
      glRotatef(angley, 0.0, 1.0, 0.0);               //rotate along
y-axis
      glRotatef(anglez, 0.0, 0.0, 1.0);               //rotate along
z-axis
      glScalef(scale, scale, 1);
```

```
	for (int i = 0; i < 6; ++i) {				//draw cube with
texture images
		glBindTexture(GL_TEXTURE_2D, texName[i]);
		glBegin(GL_QUADS);
		glTexCoord2f(0.0, 0.0); glVertex3fv(face[i][0]);
		glTexCoord2f(1.0, 0.0); glVertex3fv(face[i][1]);
		glTexCoord2f(1.0, 1.0); glVertex3fv(face[i][2]);
		glTexCoord2f(0.0, 1.0); glVertex3fv(face[i][3]);
		glEnd();
	}

	glPopMatrix();
	glFlush();
	glDisable(GL_TEXTURE_2D);
}

static void Idle(void)
{
	float time = glutGet(GLUT_ELAPSED_TIME);
	while (time > 2000) time -= 2000;

	anglex += (time / 1000) * 1;
	angley += (time / 1000) * 1;
	anglez += (time / 1000) * 1;

	if (anglex > 360)
		anglex = 0;
	if (angley > 360)
		angley = 0;
	if (anglez > 360)
		anglez = 0;

	if (scale > 1.5)
	{
		scale = 0.5;
	}
	scale += (time / 1000) * 0.0001;

	glutPostRedisplay();
}

void keyboard(unsigned char key, int x, int y)
{
	switch (key) {
	case 'x':
		anglex = (anglex + 3) % 360;
```

```
            break;
      case 'X':
            anglex = (anglex - 3) % 360;
            break;
      case 'y':
            angley = (angley + 3) % 360;
            break;
      case 'Y':
            angley = (angley - 3) % 360;
            break;
      case 'z':
            anglez = (anglez + 3) % 360;
            break;
      case 'Z':
            anglez = (anglez - 3) % 360;
            break;
      case 27: /* escape */
            glutDestroyWindow(window);
            exit(0);
      }
      glutPostRedisplay();
}

void reshape(int w, int h)
{
      glViewport(0, 0, (GLsizei)w, (GLsizei)h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 30.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
}


int main(int argc, char** argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(500, 500);
      glutInitWindowPosition(100, 100);
      window = glutCreateWindow(argv[0]);
      init();
      glutDisplayFunc(display);
      glutReshapeFunc(reshape);
      glutKeyboardFunc(keyboard);
```

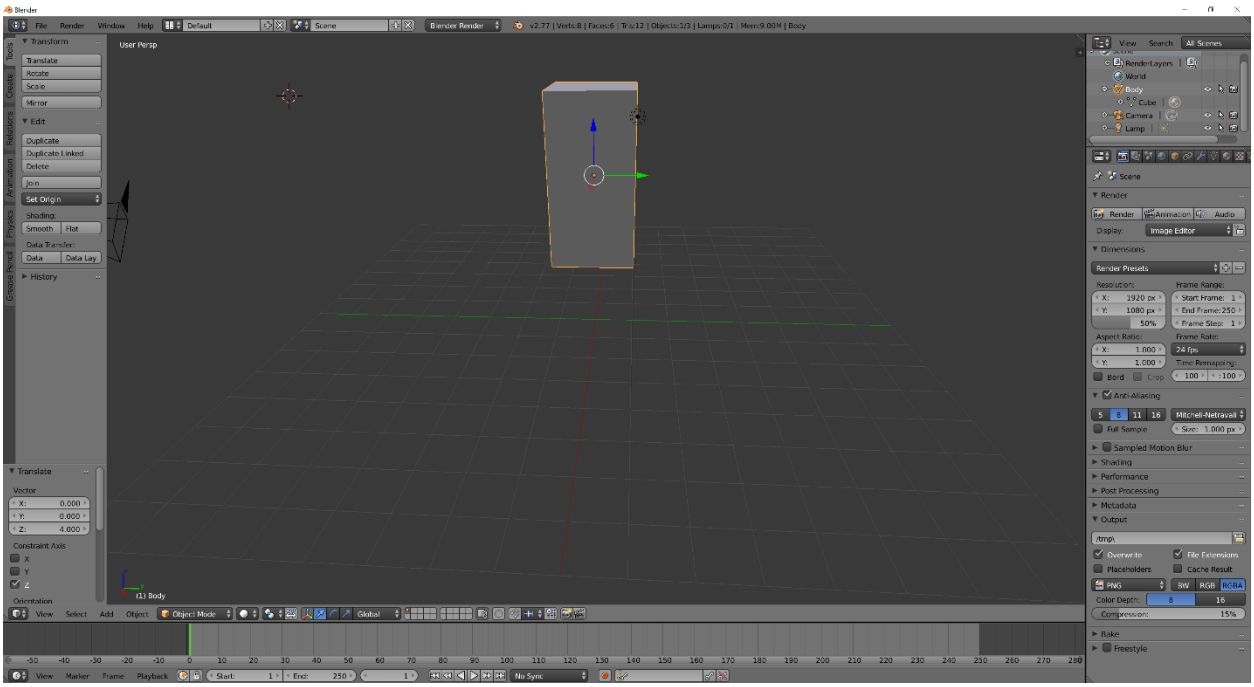
```
        glutIdleFunc(Idle);

        glutMainLoop();
        return 0;
}
```

Part 2 (success):

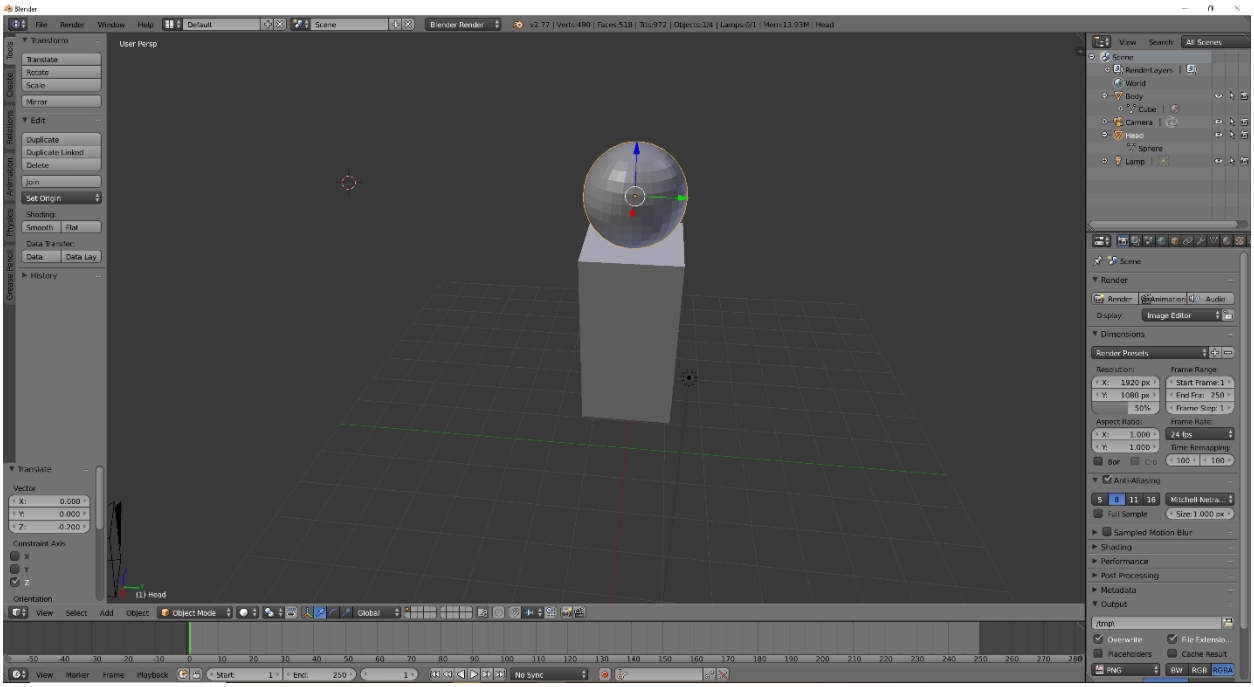
Body

For the body I used the starting cube and scaled it byt 2x in the z direction to make it a rectangle and then transformed it to account for the legs and arms.



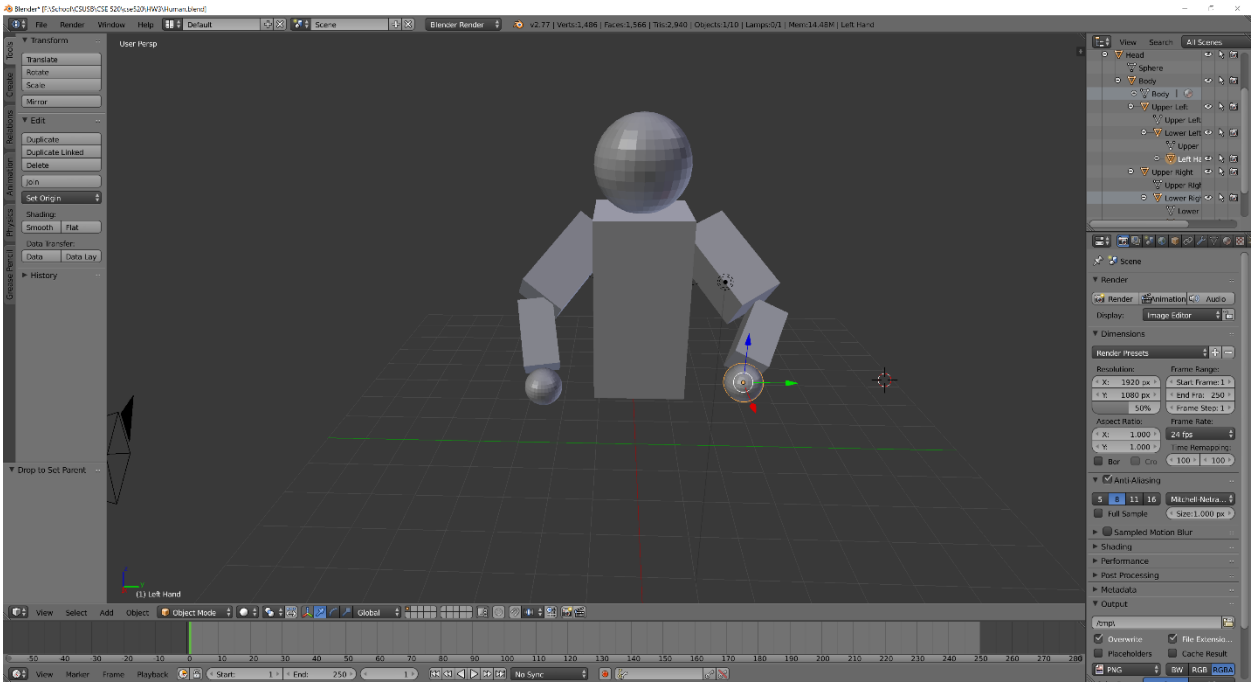
Head

Next, I added a UV sphere to use for the head and adjusted the transform so it would appear on top of the body.



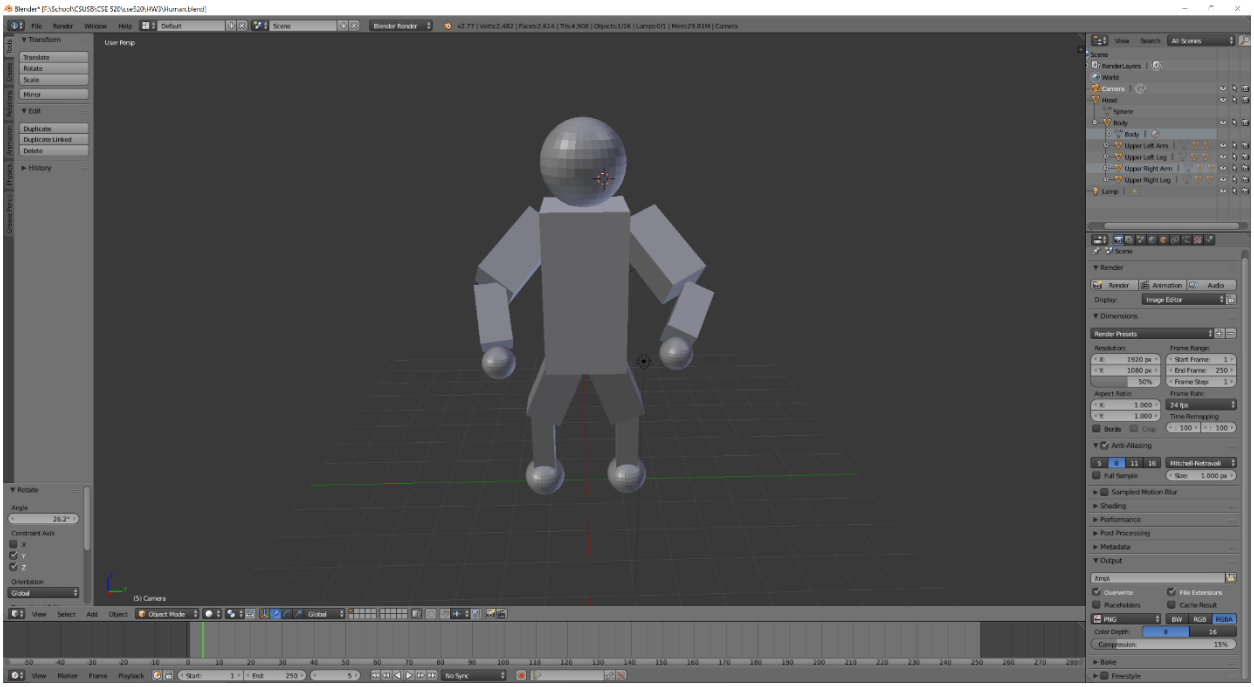
Arms

For the arms I created another cube then scaled it to be smaller than the body and be a rectangle. Next, I rotated it and then transformed it to attach to the body. I repeated these steps for the lower arm and finally made another UV sphere to use as the hand and scaled it down. Then I transformed it to the end of the lower arm. For the left arm I simply copied the right arm steps but rotated and translated in the opposite direction.



Legs

For the legs I performed the same steps for the arms, but instead I used different rotations and translations to have the cubes and spheres appear under the body as legs. I also organized the hierarchy of all the limbs to appear from head down (i.e. head parent of body, body parent of limbs).



Summary:

Overall this assignment was a lot of fun. The first part creating the cube was a lot of fun and I did have some issues getting the rotations to work properly, but incidentally created I cool shuffling effect that I decided to keep. The portion of Blender was different, and I enjoyed being able to work in that environment for a bit. I do want to investigate importing the human I made in Blender into my final project in some way. Overall, my program compiled and ran successfully as well as I created a human in Blender. I believe I earned the full 40 points for the assignment.