Graphical Programing Assignment – CO563

Code

Main file

```
#include <gl/glut.h>
#include <stdlib.h>
#include <stdio.h>
#include "fileFunctions.h"

const float BASE_HEIGHT = 1;
const float LOWER_LEG_LENGTH = 1.0;
const float UPPER_LEG_LENGTH = 0.4;

GLfloat corners[8][3] = { {0.35,0.35,-0.5},{-0.35,0.35,-0.5},
                                          {-0.35,-0.35,-0.5},{0.35,-0.35,-0.5},
                                          {0.35,0.35,0.25},{-0.35,0.35,0.25},
                                          {-0.35,-0.35,0.25},{0.35,-0.35,0.25} };

float forward = 0.0;
float angle[3];
bool lowerLegDown = true;

// Returns a pointer to the bitmap image of the bitmap specified
//  by filename. Also returns the bitmap header information.
//  No support for 8-bit bitmaps.
unsigned char* LoadBitmapFile(const char* filename, BITMAPINFOHEADER*
bitmapInfoHeader)
{
       FILE* filePtr;                                      // the file pointer
       BITMAPFILEHEADER bitmapFileHeader; // bitmap file header
       unsigned char* bitmapImage;              // bitmap image data
       int imageIdx = 0;                                  // image index counter
       unsigned char       tempRGB;              // swap variable
       // open filename in "read binary" mode
       errno_t err = fopen_s(&filePtr, filename, "rb");
       if (err != NULL)
              return NULL;
       // read the bitmap file header
       fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER), 1, filePtr);
       // read the bitmap information header
       fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER), 1, filePtr);
       // move file pointer to beginning of bitmap data
       fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK_SET);
       // allocate enough memory for the bitmap image data
       bitmapImage = (unsigned char*)malloc(bitmapInfoHeader->biSizeImage);
       // read in the bitmap image data
       fread(bitmapImage, sizeof(BYTE), bitmapInfoHeader->biSizeImage, filePtr);

       // swap the R and B values to get RGB since the bitmap color format is in BGR
       // Added -3 to stop Heap Error being generated when freeing bitmap  in VS2012
       for (imageIdx = 0; imageIdx < (bitmapInfoHeader->biSizeImage - 3); imageIdx +=
3)
       {
              tempRGB = bitmapImage[imageIdx];
              bitmapImage[imageIdx] = bitmapImage[imageIdx + 2];
              bitmapImage[imageIdx + 2] = tempRGB;

       }
       // close the file and return the bitmap image data
```

```
	fclose(filePtr);
	return bitmapImage;
}

void drawFace(int a, int b, int c, int d) {
	glBegin(GL_POLYGON);
	glTexCoord2f(0.0, 0.0);
	glVertex3fv(corners[a]);
	glTexCoord2f(0.0, 1.0);
	glVertex3fv(corners[b]);
	glTexCoord2f(1.0, 1.0);
	glVertex3fv(corners[c]);
	glTexCoord2f(1.0, 0.0);
	glVertex3fv(corners[d]);
	glEnd();
}

void arrayCube() {
	drawFace(0, 3, 2, 1);
	drawFace(3, 0, 4, 7);
	drawFace(2, 3, 7, 6);
	drawFace(1, 2, 6, 5);
	drawFace(4, 5, 6, 7);
	drawFace(5, 4, 0, 1);
}

float walkForward() {
	if (GetAsyncKeyState(0x57)) //Forward (W)
	{
		return 1.0;
	}
	else if (GetAsyncKeyState(0x53)) // Back (S)
	{
		return -1.0;
	}
}

void rotate() {
	if (walkForward() == 1.0) {
		forward += -0.025;

		if (lowerLegDown) angle[1] -= 0.2;
		else angle[1] += 0.2;

		if (angle[1] < 348.75) lowerLegDown = false;
		if (angle[1] > 360) lowerLegDown = true;
	}
	else if (walkForward() == -1.0) {
		forward += 0.025;

		if (lowerLegDown) angle[1] -= 0.2;
		else angle[1] += 0.2;

		if (angle[1] < 348.75) lowerLegDown = false;
		if (angle[1] > 360) lowerLegDown = true;
	}
	else {
		forward += 0.0;
	}
	glutPostRedisplay();
}
```

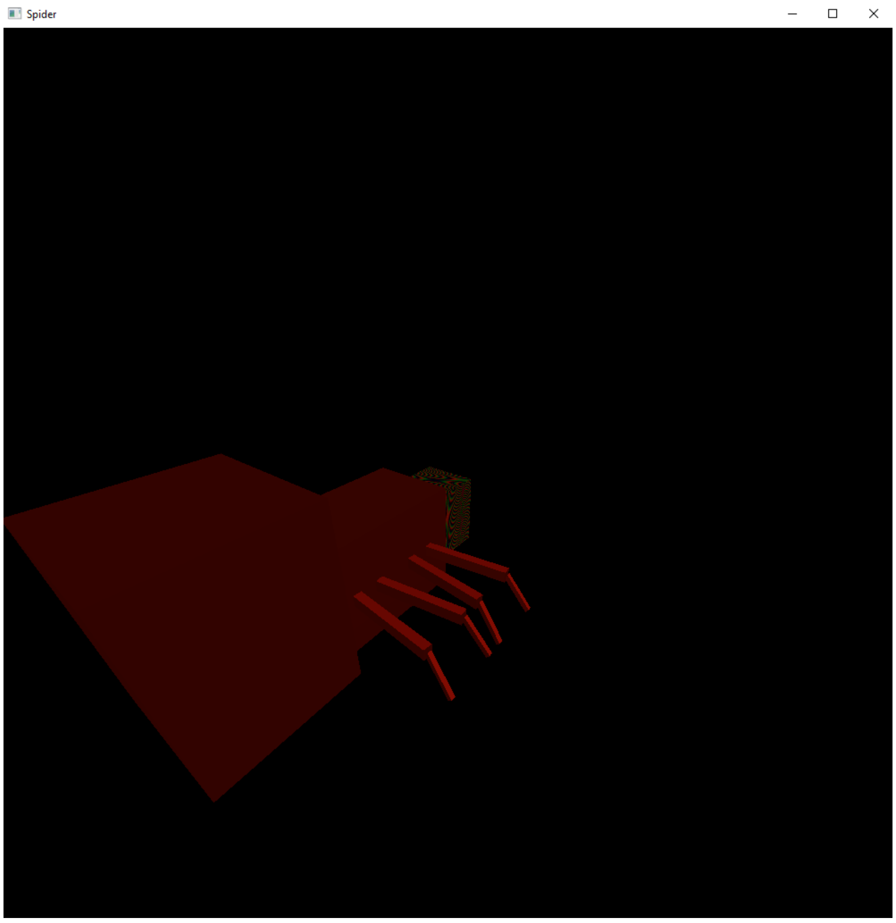
```
void drawUpperLeg() {
	glPushMatrix();
	glRotatef(-45, 0.0, 0.0, 1.0);
	glTranslatef(0.0, (UPPER_LEG_LENGTH * 0.5), 0.0);
	glScalef(0.05, UPPER_LEG_LENGTH, 0.05);
	glutSolidCube(1.0);
	glPopMatrix();
}

void drawLowerLeg() {
	glPushMatrix();
	glTranslatef(0.0, (LOWER_LEG_LENGTH * 0.5), 0.0);
	glScalef(0.1, LOWER_LEG_LENGTH, 0.1);
	glRotatef(90.0, 0.0, 1.0, 0.0);
	glutSolidCube(1.0);
	glRotatef(-90.0, 0.0, 1.0, 0.0);
	glPopMatrix();
}

void drawBase() {
	glPushMatrix();
	glColor3f(0.325, 0.2, 0.05);
	glScalef(1.0, BASE_HEIGHT, 2.0);
	glutSolidCube(1.0);
	glColor3f(0.35, 0.2, 0.05);
	glPopMatrix();
}

void drawLeg() {
	glPushMatrix();
	drawLowerLeg();
	glTranslatef(0.0, LOWER_LEG_LENGTH, 0.0);
	drawUpperLeg();
	glTranslatef(0.0, -LOWER_LEG_LENGTH, 0.0);
	glPopMatrix();
}

void drawRear() {
	glPushMatrix();
	glScalef(1.5, 1.5, 1.5);
	glutSolidCube(1.0);
	glPopMatrix();
}

void drawHead() {
	glPushMatrix();
	drawFace(0, 3, 2, 1);
	drawFace(3, 0, 4, 7);
	drawFace(2, 3, 7, 6);
	drawFace(1, 2, 6, 5);
	drawFace(4, 5, 6, 7);
	drawFace(5, 4, 0, 1);
	glPopMatrix();
}

void drawSpider() {
	glTranslatef(0.0, 0.0, forward);

	//Draws the body of the spider
	glPushMatrix();
	drawBase();
	//Draws the spiders rear
	glPushMatrix();
	glTranslatef(-0.25, 0.0, 1.5);
	drawRear();
	glPopMatrix();

	//Draws the spider head
	glPushMatrix();
	glTranslatef(0.0, 0.0, -1.25);
	drawHead();
	glPopMatrix();

	//Draws all of the legs
	//First set of legs
	glTranslatef(0.0, (BASE_HEIGHT * 0.5), -0.65);
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.3);
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.5);
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.3);
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	glPopMatrix();
	//Draws second set of legs
	glTranslatef(0.0, -(BASE_HEIGHT * 0.5), -0.1);
	glRotatef(180, 100.0, 0.0, 0.0);
	glScalef(1.0, 1.0, 1.0);
	glTranslatef(0.0, 0.0, -0.6);
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.35);
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.5);
	glRotatef(angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	glTranslatef(0.0, 0.0, 0.35);
	glRotatef(-angle[1], 1.0, 0.0, 0.0);
	drawLeg();
	glRotatef(angle[1], 1.0, 0.0, 0.0);
}

void display() {
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(3, 2, 4, 0.0, -2.5, 0.0, 0.0, 1.0, 0.0);

	//Set Material Properties
	GLfloat qaBlack[] = { 0.0, 0.0, 0.0, 1.0 };
	GLfloat qaBrown[] = { 0.35, 0.2, 0.05, 1.0 };
	GLfloat qaWhite[] = { 1.0, 1.0, 1.0, 1.0 };
	glMaterialfv(GL_FRONT, GL_AMBIENT, qaBrown);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, qaBrown);
	glMaterialfv(GL_FRONT, GL_SPECULAR, qaBrown);
	glMaterialf(GL_FRONT, GL_SHININESS, 40.0);

	//Set normal
	glNormal3f(.0, 0.0, 1.0);

	//Draws spider
	glRotatef(-90.0, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, forward);
	drawSpider();
	glutSwapBuffers();
}

void init() {
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glEnable(GL_DEPTH_TEST);
	glColor3f(0.35, 0.2, 0.05);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glFrustum(-1.0, 1.0, -0.5, 1.2, 0.8, 100.0);

	//Set Up Lighting
	glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);

	// Intensity and color of the lighting
	GLfloat qaAmbientLight[]    = { 0.85, 0.85, 0.85, 1.0 };
	GLfloat qaDiffuseLight[]    = { 0.15, 0.15, 0.15, 1.0 };
	GLfloat qaSpecularLight[]   = { 0.0, 0.0, 0.0, 1.0 };
	glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight);
	glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight);

	//Define light position
	GLfloat qaLightPosition[] = { 5, 5, 5, 1.0 };
	glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition);

	//Texture Stuff
	BITMAPINFOHEADER bitmapInfoHeader;
	unsigned char* bitmapData;
	bitmapData = LoadBitmapFile("spider.bmp", &bitmapInfoHeader);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
	glTexImage2D(GL_TEXTURE_2D,
		0,
		GL_RGB,
		bitmapInfoHeader.biWidth,
		bitmapInfoHeader.biHeight,
		0,
		GL_RGB,
		GL_UNSIGNED_BYTE,
		bitmapData);
	free(bitmapData);

	glEnable(GL_TEXTURE_2D);
	glEnable(GL_DEPTH_TEST);
}
```

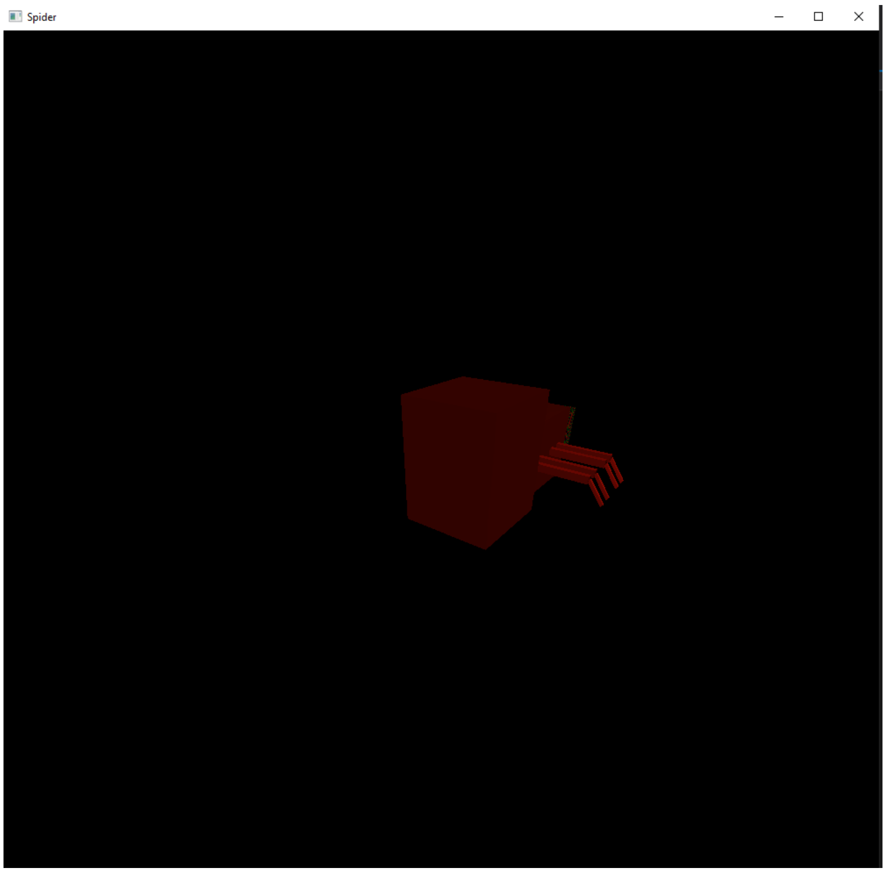
```
int main(int argc, char** argv) {
	angle[0] = 0;
	angle[1] = 360;
	angle[2] = 315;
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
	glutInitWindowSize(1000, 1000);
	glutInitWindowPosition(1000, 200);
	glutCreateWindow("Spider");
	glutDisplayFunc(display);

	//Texture setup
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 64, 64, 0, GL_RGB, GL_UNSIGNED_BYTE, 
"spider.bmp");

	init();
	glutIdleFunc(rotate);
	glutMainLoop();
}
```

Header file (fileFunctions.h)

```
#pragma once

unsigned char* LoadBitmapFile(const char* filename,
	BITMAPINFOHEADER* bitmapInfoHeader);
```

Screenshots





Notes

It is to be noted that not all the code is working as expected. For example, I could not get the texture working properly. All the code is there and set up, but it will not seem to apply the texture to the head of the spider (note that the head of the spider is the only body part that has been drawn vertex by vertex specifically for the accommodation of a texture). It displays a strange pattern in place of the texture on the head which I assume is some form of default stand in texture for when there is an error in what is being loaded.

I additionally had some issues where having implemented the code for the texture, the rest of the spider's model changed color and made it slightly darker and with a more red-ish tint for an unknown reason. Thus, the strange color of the spider, if you take the texture setup code out the spider should revert to the prior more natural brown color.

Texture file for reference