# Project Report

Tommy Zhang 22181467

## Program Overview

All of the required functions from the project details page are working, although I am not very sure, it looks working almost in the same way as the videos show. And I also implement some additional functions and made some change to the source skeleton code which I will mention below.

## Steps for Tasks

### Part I

At the beginning of this project, I just followed the introduction from the COMPILATON-HELP file to construct the environment for running code in my own computer. Then the C compiler show me the error from the source code, I removed the part of codes which generate the trouble, and the scene runs successfully in my computer. After that, I read the code to understand the flow of this program, then fix the cameraRotate easily. Then I can try the basic functionality already implemented based on reading the code.
The first seven questions(A-G) are quite simple, just modified the code as the lecture or textbook shows or writing new functions as the similar function implemented before.

**A cameraRotate: Change the view equation from display()**

```
view = Translate(0.0, 0.0, -viewDist)* RotateX(camRotUpAndOverDeg) *
RotateY(camRotSidewaysDeg);
```

**B objRotateXYZ: Change the model equation from darwMesh()**

```
mat4 model = Translate(sceneObj.loc) *RotateX(sceneObj.angles[0])*
RotateY(sceneObj.angles[1])*RotateZ(sceneObj.angles[2])*
Scale(sceneObj.scale);
```

**C materials: Change the materialMenu like other submenu function, add the adjustAmbientDiffuse and adjustSpecularShine function like adjustBlueBrightness.**

**D closeup: Change the nearDist parameter for projection.**

**E reshape: Change the reshape function to change the projection matrix when the aspect change**

**F light reducing: Change the color.rgb from the vertex shader to apply attenuation to the diffuse and specular for the distance increases to the light, the equation for the attenuation from the lecture slide.**

**G light per frag: Move the related parameter from the vertex shader to the fragment shader, I rewrite the whole fragment shader as the lecture slide and textbook shows.**

**H shine: Change the calculation for light in the fragment shader, move the specular outside so that it won't be affected by the texture colour.**

For the last question in part 1, I duplicate the same code as the original light firstly, then try to make it a directional light source followed the instruction from the project details page. And simply modified the code to make it a little bit different from the first light.
That's how I completed the first part. I decided to make the additional function after I complete writing the required tasks. During modifying the code, I gained some idea about additional function or change of the source code to correct some function, I will put them in the additional functionalities part. I also discussed with my friend about the thought to solve problems and how to improve the code.

# Part II

After the testing of part 1, I followed the instruction given in addingAnimation.txt to create the environment for animation. The first question just needed modify the gl_FragColor equation in fragment shader.

**A Texture Scale:**

```
gl_FragColor = color * texture2D( texture, texCoord * texScale )
```

Then I downloaded the newest version of Blender and MakeHuman in my own Mac computer, then I realised it is a mistake. After making three different human models, I exported them as mhx2 and imported them in Blender. The mhx2 files are 3 times to 4 times larger than the example file Mate.mhx. And when blender process the imported human model, the FPS is 1.2 even there's no animation at all. I tried to use the MakeWalk plugin to retarget MoCap and export DirectX file, it takes almost half hour, and the DirectX file is 220MB, and the model didn't show normally inside the sense program. So I decided to implement the animation code first.

**D Coding to show animations:**

Most of the codes needed for animation are already implemented by following the instruction, the only missing parameter is the <POSE_TIME> for calculateAnimPose() in the drawMesh() function. This function is to display the current animation position, by calling this function continuously the positions of the models keep changing so the viewer can see a moving animation. So we have to tell the function what is the current tick to display.
To record the time, I add some extra variable in the structure of SceneObject.

| Variable | Description |
|---|---|
| aniStartime | The start time of animation, will be record as the time when the object added into the sense. |
| aniSpeed | The playback speed of animation which can be adjusted in the menu, the default value is 1.0. |
| aniCurrentTicks | The current tick is playing for animation, sets as 0 when the object added into the sense. |
| aniTotalTicks | The total number of ticks for animation. |
| aniTicksPerSec | The number of ticks will be played per second for animation. |
| aniTotalTime | Use aniTotalTicks and aniTicksPerSec to calculate the total time of animation for calculate the current ticks inside an animation circle |

When calculating all of the necessary variables once the model added to the sense, the animation information hasn't been loaded at that time so the program went crash, so I have to call loadMeshIfNotAlreadyLoaded() first to read animation information.
Then I send the aniCurrentTicks as the parameter of calculateAnimPose() to show the current animation position, then calculate the next tick by count the already ran time first and use fmod() to calculate the current time inside a circle period and multiple aniTicksPerSec to get the new aniCurrentTicks.

```
float runnedTime = glutGet(GLUT_ELAPSED_TIME) - so.aniStartime;
sceneObjs[i].aniCurrentTicks =
          so.aniTicksPerSec * fmod(runnedTime, so.aniTotalTime)/1000;
```

After the successfully showing animation, I went back to school labs for build the human models again, because my friend told me that she creates the models in Windows Lab using the older version of MakeHuman and the blender and models work correctly. After downloading the old MakeHuman, I found some functions are missing but the exported file is quite smaller than the new

version and it can be easily imported by Blender and export the DirectX animation files. So I just rebuild my three human models and export the x files successfully. After testing and checking the animation functions, I start to write additional functions.

# Additional Functionalities and Modified

### A Repair the shift functional key:

Because I use my MacBook as the development environment, I don't have a mouse so that I can't use the middle key, the only solution is to use the shift functional key, but I found it doesn't work as the description, so I changed the mouseClickOrScroll() in gnatidread.h . Change the activateTool to GLUT_MIDDLE_BUTTON so that the shift functional key works.

```
static void mouseClickOrScroll(int button, int state, int x, int y)
{
    if (button==GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        if (glutGetModifiers() != GLUT_ACTIVE_SHIFT) activateTool(button);
        else activateTool(GLUT_MIDDLE_BUTTON);
    }
```

### B Record the previous clicking position:

When writing the first tasks, I found the view position always get changed after clicking and dragging on the screen. After a discussion with my friend, I found I can change the gnatidread.h to fix it.

```
// clickPrev = currMouseXYscreen(mouseX, mouseY); The Original Code
prevPos = currMouseXYscreen(mouseX, mouseY); //Changed
```

### C Implement an object choice menu:

I found after adding a new object, all of the operations are on the latest object, then I can't change the status of other objects. So I decided to add a new menu to change the currentObject. First I add submenu in the makeMenu() function.

```
    currentObjectId =glutCreateMenu(currentObjectMenu);
glutAddMenuEntry(objectMenuEntries[sceneObjs[currObject].meshId-1],currObject
+100);
    glutAddSubMenu("Current Object", currentObjectId);
```

Inside the makeMenu function, I also add the first item as an menu entry, the ID is ObjectID +100 so that there won't be conflict with existent ID. The menu ID 'currentObjectId' is saved as a global parameter so that it can be used inside the addObject().

Then the menu function is created to change the toolObj and currObject value to the chosen object.

```
static void currentObjectMenu(int id){
    deactivateTool();
    toolObj = currObject = id-100;
}
```

Finally, a glutAddMenuEntry() will be called inside the addObject() function, so the new object will be added into the menu when added.

```
    glutSetMenu(currentObjectId);
    glutAddMenuEntry(objectMenuEntries[id-1],currObject+100);
```

### D Hide and Redisplay Object

I added a hide boolean variable into the SceneObject structure, and add a if loop in the display() function, this object will not display if hide is true. Then add control code to the mainmenu() function as below.

```
    if (id == 90 && currObject>=0) {
        bool statu = sceneObjs[currObject].hide;
        if (statu) sceneObjs[currObject].hide = false;
        if (!statu) sceneObjs[currObject].hide = true;
    }
```

The Object will hide if choose Hide/Display from the menu or redisplay when it's hiding.

**E Duplicate Object**

I modified the addObject() function so that it accepts the second parameter to designate the texture ID for object to add. And if the texture ID is zero then just use a random one. Then add a new item to main menu and a duplicateObject() function to duplicate an object with same parameter.

**F Animation Upend**

I added another boolean variable aniUpend to record the play status and add a new item to animation menu to change it. Then modified the display() function to change the calculation of aniCurrentTicks. When aniUpend is true, the animation will play in a reserve order.

```
if (!so.aniUpend) {
    sceneObjs[i].aniCurrentTicks =
    so.aniTicksPerSec * fmod(runnedTime, so.aniTotalTime)/1000;
} else {
    sceneObjs[i].aniCurrentTicks = so.aniTicksPerSec *
    (so.aniTotalTime-fmod(runnedTime, so.aniTotalTime))/1000;
}
```

**G Unify Animations**

I add another animation extra functions to unite the parameter of animation for all objects with animation in the senses, so that all the human models with the same animation will have the same position like they're rising their legs at the same time.

```
void unifyAnimation() {
    float currentTime = glutGet(GLUT_ELAPSED_TIME);
    for (int i=0; i < nObjects; i++) {
        if(sceneObjs[i].aniTotalTime > 0.0) {
        sceneObjs[i].scale = 0.05;
        sceneObjs[i].aniStartime = currentTime;
        sceneObjs[i].aniSpeed = 1.0;
        sceneObjs[i].aniCurrentTicks = 0.0;
        }
    }
}
```

# Personal Reflection

When working on this project, I realised that finishing a whole program likes an adventure. I can get an achievability from finishing it. It's better to have a deep read of the code before modifying it so that you can understand the running flow and functions to avoid crash and bug.

I was stuck when modifying the <POSE_TIME> for animation implement, it's harder than I thought. Spent an afternoon and got no progress, checked resources from the internet and asked for help from friends. But the thought to solve the question finally came to my senses when I was lying in the bed after waking up on the second day. No matter what kind of project to do, hurry doesn't help. And it's necessary to inspect the details, the animation didn't show continuous just because a & is missing in the front of parameter.

By finishing this project and this unit, I basically understand how the graphic engine works, and how to send code to GPU for improving the efficiency of graphic system. These knowledge will be very useful if I engaged game development industry.