CITS 3003
Graphics and Animation
Project

Levente Zombori and Jong Su Jang
22228596 - 22522128

Semester One, 2019

Overview

All tasks required in both Part One and Part Two of the project have been completed. The tasks have been tested and compared against the requirements listed and should be fully functional.

The Project Report aims to go over the steps taken to complete the tasks, and the reasoning behind those steps. Please refer to the appendices for more in depth information about the project tasks. Whilst we aimed to include all changed sections of code within the appendices, it is possible that minor changes or additions have been omitted. However, the source code for the project includes all changed areas, with comments wherever necessary. In addition, certain sections of code have had to be re-modified and as such the report may or may not show the most up to date version of the code.

We have opted to work alongside eachother for the whole duration of the project and worked within the Computer Science labs during lab hours. It is not possible to divide up parts of our personal contribution as we have worked together through each individual task and brainstormed possible solutions together.

This Project overall has been a positive learning experience and was good in getting hands on experience with Graphics and Animation. The best part about the project was how it included both design and animation as well as programming. This proved to be an exciting challenge and allowed us to gain experience in a wide range of areas. To improve upon the existing project I would recommend clearing up the instructions, and updating the sections that have since depreciated due to new software releases (mainly the makehuman and blender instructions).

1 Project – Part One

Reflection - 1

Having completed the first part of the Project, we were able to benefit from a variety of new things learned from the required tasks. All tasks have been in general relatively attainable, with the shaders proving to be the most demanding of them all.

Due to the nature of the Project, it was difficult to get started at first. We first had to reflect on what the source code was doing, before we could properly undertake the outlined tasks. Thus having been given the vast majority of the source code proved to be a mixed experience. The majority of our time was spent figuring out what parts of the code needed modification or addition, at times having to experiment with the code until the right outcome was achieved.

In hindsight Part One of the Project has been an overall positive learning experience as it allowed us to study and learn how bigger projects are set out and how they function. We have found that the supplied materials combined with the ones available have been sufficient in completing the first half of the project.

All tasks (A, B, C, D, E, F, G, H, I) have been completed.

1.1 Camera Rotation

The display function needed to be modified in order to allow the camera to rotate as required. This was achieved through the RotateX and the RotateY rotations using the predefined camera angle variables.

See appendix 3.1 on page 5 for reference.

1.2 Object Rotation

The desired object rotation was achieved comparably to the camera rotation. The drawMesh function was modified to include the RotateX, RotateY and RotateZ rotations using the angles array for the objects. Furthermore, the X and Z rotations were inverted to match the direction of movement required.

See appendix 3.2 on page 5 for reference.

1.3 Lighting and Shininess Adjustments

Adjustment functions were added in order to adjust the required values as necessary. Furthermore, an additional menu item was also implemented to allow for the adjustment of these values from the menu options. The shine value was further boosted to 20 to allow it to scale to 100.

See appendix 3.3 on page 5 for reference.

1.4 Enhanced Close-Up View

In order to allow for a more close-up view, the reshape callback was adjusted. The view-distance and near-distance variables have been adjusted by a scale of 4x in order to increase the field of view of the camera, as well as to solve existing issues with triangles clipping if they are even slightly close to the camera.

See appendix 3.4 on page 6 for reference.

1.5 Window Reshaping

The reshape function was modified to scale the viewport the same way both horizontally and vertically. The result was achieved through the use of the height and width ratio, which allowed the top and bottom planes to scale when the width is less than the height. Whatever is visible when the window is square continues to be visible as the width is decreased, similar to what happens if the window height is decreased starting with a square.

See appendix 3.5 on page 7 for reference.

1.6 Light Reduction

The vertex shader was modified so that the first light reduces noticeably proportionally to the as the distance increases. The calculations were later modified to account for multiple light sources and migrated to the fragment shader as per later requirements.

See appendix 3.6 on page 7 for reference.

1.7 Fragment Lighting

The contents of the vertex shader were moved to the fragment shader, and adjusted as necessary to account for the shader differing shader requirements. The fragment-shader calculates the directions for individual fragments rather than for the vertices.

See appendix 3.7 on page 8 for reference.

1.8 Specular Highlights

The shader was modified so that the specular component always shines towards white, regardless of the texture and the colour assigned to the object. The brightness and colour of the lighting sources are passed into the shader separately rather than being multiplied into RGB. Because the specular component should be independent of the light's colour it is not multiplied by brightness. The component is also added as the last element, hence it should not be effected by the texture's colour.

See appendix 3.8 on page 9 for reference.

1.9 Second Light

The code was modified to account for a second light source. A new sphere object was created, with differring positions from the first. The second light was created to be directional, whilst the first one remains positional. The lighting positions are passed into the shader separately, with sections modified duplicated as necessary.

See appendix 3.9 on page 10 for reference.

2 Project – Part Two

Reflection - 2

Part Two of the project has been significantly easier in terms of complexity. It required less coding in general and we were already familiar with the source code so changes were relatively easy to make.

Difficulty aside the biggest hurdle has been time management. Whilst the Project has been given out to us with sufficient time to complete, certain tasks had to be delayed due to ongoing issues with the lab computers as well insufficient instructions. The Makehuman and Blender components of the tasks were difficult to get working and took a lot of trial and error.

The modeling and animation parts have been a challenge due to the lack of familiarity with the enviroment, overall however they have been a positive learning experience and proved to be an entertaining part of the project. The coding of the animations has been relatively straight forward once we figured out what parts of the code need additions and changes.

All tasks (A, B, C, D) have been completed.

2.1 Texture Scale

In order to implement texture scaling the fragment shader was modified to include the texScale variable by replacing the default placeholder value. The scene-start.cpp file did not need any changes as it already contained the necessary code needed to make the texture scaling function as required.

See appendix 3.10 on page 10 for reference.

2.2 Makehuman Model

The models were created using the Makehuman open source program, as per task requirements three unique models were designed. Then a skeleton was rigged to each model to allow for animation in Blender. The models were then exported in '.mhx' files, which is compatible with Blender.

See appendix 3.11 on page 11 for reference.

2.3 Blender Animation

The Makehuman models were loaded into Blender and paired with a selected mocap file, the mocap file was then retargeted to the skeleton of the model. This was repeated for all models, each model receiving a unique animation. The model and animation were both exported in directx format to allow for compatibility with our project.

See appendix 3.12 on page 12 for reference.

2.4 Code Animation

Animating the models within the project required several changes. The number of meshes has been increased, and corresponding menu entries added. Model scaling and orientation has been adjusted as necessary. The model animations were not uniform in length, so unique frame lengths have been added for each model, with base models having one frame. In order to get the animation working the frame variable was added to calculateAnimPos. The frame variable is modified by the new 'Animation' menu. The animation functionality has been extended by adding options pause/play, decrease the animation speed, increase the animation speed and reset the animation speed to normal.

See appendix 3.13 on page 13 for reference.

3 Appendices

3.1 Task 1.A

scene-start.cpp

```
570. // Modified camera rotation
571. mat4 rotation = RotateX(camRotUpAndOverDeg) * RotateY(camRotSidewaysDeg);
572.view = Translate(0.0, 0.0, -viewDist) * rotation;
```

3.2 Task 1.B

scene-start.cpp

```
507. // Object rotation
508. mat4 rotation = RotateX(sceneObj.angles[0]) * RotateY(sceneObj.angles[1])
* RotateZ(sceneObj.angles[2]);
509. mat4 model = Translate(sceneObj.loc) * rotation * Scale(sceneObj.scale);
```

scene-start.cpp

```
804. // Inverted angles to match requirement
805. else if (id == 55 && currObject >= 0)
806. {
807. setToolCallbacks(adjustAngleYX, mat2(400, 0, 0, 400),
                     adjustAngleZTexscale, mat2(-400, 0, 0, 15));
808. }
```

3.3 Task 1.C

scene-start.cpp

```
738. // Created function for Ambient and Diffuse
739. static void adjustAmbientDiffuse(vec2 ambience)
740. {
741.   sceneObjs[toolObj].ambient = max(0.0f, sceneObjs[toolObj].ambient
+ ambience[0]);
742.   sceneObjs[toolObj].diffuse = max(0.0f, sceneObjs[toolObj].diffuse
+ ambience[1]);
743. }
744.
745. // Created function for Specular and Shine
746. static void adjustSpecularShine(vec2 shiny)
747. {
748.   sceneObjs[toolObj].specular = max(0.0f, sceneObjs[toolObj].specular
+ shiny[0]);
749.   sceneObjs[toolObj].shine = max(0.0f, sceneObjs[toolObj].shine
+ shiny[1]);
750. }
```

scene-start.cpp

```
764. // Created menu items to allow adjustment
765. else if (id == 20)
766. {
767.   toolObj = currObject;
768.   setToolCallbacks(adjustAmbientDiffuse, mat2(1, 0, 0, 1),
adjustSpecularShine, mat2(1, 0, 0, 20));
769. }
```

3.4 Task 1.D

Info: Projection was further modified in Task E of the Project.

scene-start.cpp

```
35. // View distance increased by 4x
36. static float viewDist = 6.0;
```

scene-start.cpp

```
310. // Scaled the sideview by 4x
311. static void doRotate()
312. {
313.   setToolCallbacks(adjustCamrotsideViewdist, mat2(400, 0, 0, -8),
adjustcamSideUp, mat2(400, 0, 0, -90));
314. }
```

scene-start.cpp

```
923. // Increased near distance by one decimal point
924. GLfloat nearDist = 0.02;
```

scene-start.cpp

```
945. projection = Frustum(left, right, bottom, top, nearDist, 100.0);
```

3.5 Task 1.E

scene-start.cpp

```
927. // Window Reshaping
928. GLfloat bottom, top, left, right;
929.
930. if (width < height)
931. {
932.   bottom = -nearDist * (float)height / (float)width;
933.   top = nearDist * (float)height / (float)width;
934.   left = -nearDist;
935.   right = nearDist;
936. }
937. else
938. {
939.   bottom = -nearDist;
940.   top = nearDist;
941.   right = nearDist * (float)width / (float)height;
942.   left = -nearDist * (float)width / (float)height;
943. }
944.
945. projection = Frustum(left, right, bottom, top, nearDist, 100.0);
```

3.6 Task 1.F

Info: Additionally, Light Reduction was further edited in Tasks G, H & I as well as 2A of the Project.

vStart.glsl

```
39. // Vector from the vertex to the light
40. Lvec1 = LightPosition1.xyz - pos;
41. Lvec2 = LightPosition2.xyz;
```

fStart.glsl

```
58. // Light reduction based on distance
59. float dist = 0.01 + length(Lvec1);
```

fStart.glsl

```
63. vec4 color = globalAmbient + ((ambient1 + diffuse1) / dist)
+ (ambient2 + diffuse2);
64. color.a = 1.0;
65. gl_FragColor = color * texture2D(texture, texCoord * texScale)
+ (specular1 / dist) + specular2;
```

3.7 Task 1.G

Info: Fragment Lighting was later modified in Task I of the Project.

fStart.glsl

```
// Duplicated to account for the two light sources
vec3 L1 = normalize(Lvec1);  // Direction to the light source
vec3 L2 = normalize(Lvec2);  // Direction to the light source

vec3 H1 = normalize(L1 + E); // Halfway vector
vec3 H2 = normalize(L2 + E); // Halfway vector

vec3 E = normalize(-pos);    // Direction to the eye/camera

// Compute terms in the illumination equation
//----------------------------------------
vec4 ambient1 = vec4((LightColor1 * LightBrightness1), 1.0)
* vec4(AmbientProduct, 1.0);
vec4 ambient2 = vec4((LightColor2 * LightBrightness2), 1.0)
* vec4(AmbientProduct, 1.0);

float Kd1 = max(dot(L1, N), 0.0);
vec4 diffuse1 = Kd1 * vec4((LightColor1 * LightBrightness1), 1.0)
* vec4(DiffuseProduct, 1.0);
float Kd2 = max(dot(L2, N), 0.0);
vec4 diffuse2 = Kd2 * vec4((LightColor2 * LightBrightness2), 1.0)
               * vec4(DiffuseProduct, 1.0);

float Ks1 = pow(max(dot(N, H1), 0.0), Shininess);
float Ks2 = pow(max(dot(N, H2), 0.0), Shininess);

vec4 specular1 = Ks1 * LightBrightness1 * vec4(SpecularProduct, 1.0);
vec4 specular2 = Ks2 * LightBrightness1 * vec4(SpecularProduct, 1.0);

if (dot(L1, N) < 0.0)
{
  specular1 = vec4(0.0, 0.0, 0.0, 1.0);
}
if (dot(L2, N) < 0.0)
{
  specular2 = vec4(0.0, 0.0, 0.0, 1.0);
}

// The globalAmbient is independent of the distance from the light source
//----------------------------------------
vec4 globalAmbient = vec4(0.1, 0.1, 0.1, 1.0);
```

3.8 Task 1.H

scene-start.cpp

```
591. // Separate calculations for two light sources
592. glUniform3fv(glGetUniformLocation(shaderProgram, "LightColor1"), 1,
lightObj1.rgb);
593. CheckError();
594. glUniform3fv(glGetUniformLocation(shaderProgram, "LightColor2"), 1,
lightObj2.rgb);
595. CheckError();
596. glUniform1f(glGetUniformLocation(shaderProgram, "LightBrightness1"),
lightObj1.brightness);
597. CheckError();
598. glUniform1f(glGetUniformLocation(shaderProgram, "LightBrightness2"),
lightObj2.brightness);
599 . CheckError();
```

scene-start.cpp

```
611. // Accounting for two light sources
612. vec3 rgb = so.rgb * so.brightness * 2.0;
613.
614. glUniform3fv(glGetUniformLocation(shaderProgram, "AmbientProduct"),
1, so.ambient * rgb);
615. CheckError();
616. glUniform3fv(glGetUniformLocation(shaderProgram, "DiffuseProduct"),
1, so.diffuse * rgb);
617. CheckError();
618. glUniform3fv(glGetUniformLocation(shaderProgram, "SpecularProduct"),
1, so.specular * rgb);
619. CheckError();
620. glUniform1f(glGetUniformLocation(shaderProgram, "Shininess"),
so.shine);
621. CheckError();
```

fStart.glsl

```
58. // Light reduction based on distance
59. float dist = 0.01 + length(Lvec1);
```

3.9 Task 1.I

Info: Duplicated existing code sections as necessary to account for multiple light sources.

scene-start.cpp

```
465. // Added a second light source
466. addObject(55);
467. sceneObjs[2].loc = vec4(-2.0, 1.0, -1.0, 1.0);
468. sceneObjs[2].scale = 0.2;
469. sceneObjs[2].texId = 0;
470. sceneObjs[2].brightness = 0.2;
```

scene-start.cpp

```
582. // Duplicated light source calculations
583. SceneObject lightObj2 = sceneObjs[2];
584. vec4 lightPosition2 = rotation * lightObj2.loc;
585. glUniform4fv(glGetUniformLocation(shaderProgram, "LightPosition2"),
1, lightPosition2);
587. CheckError();
```

3.10 Task 2.A

fStart.glsl

```
17. uniform float texScale;
```

fStart.glsl

```
62. // Added textures scaling
63. color = globalAmbient + ((ambient1 + diffuse1) / dist)
+ (ambient2 + diffuse2);
64. color.a = 1.0;
65. gl_FragColor = color * texture2D(texture, texCoord * texScale)
+ (specular1 / dist) + specular2;
```

3.11 Task 2.B

Info: The models textures had to later be removed to successfully implement them.

Figure 1: Male.mhx2

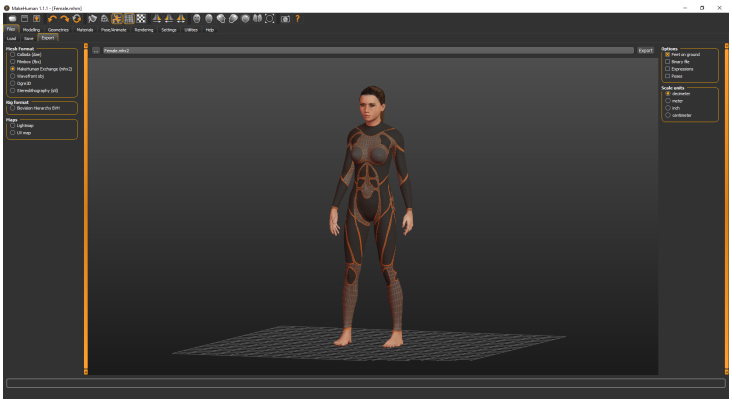

Figure 2: Female.mhx2

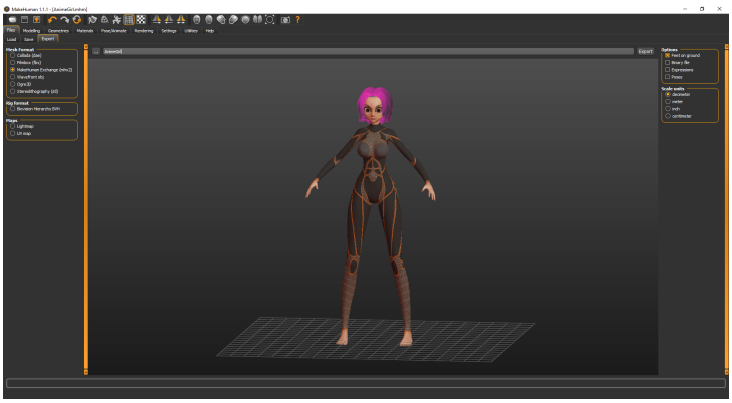

Figure 3: Anime.mhx2

3.12 Task 2.C

Info: The models hair had to later be removed to successfully implement them.

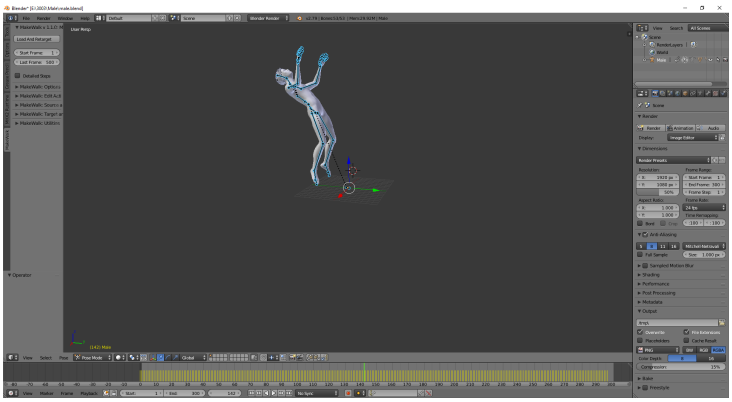

Figure 4: model57.x

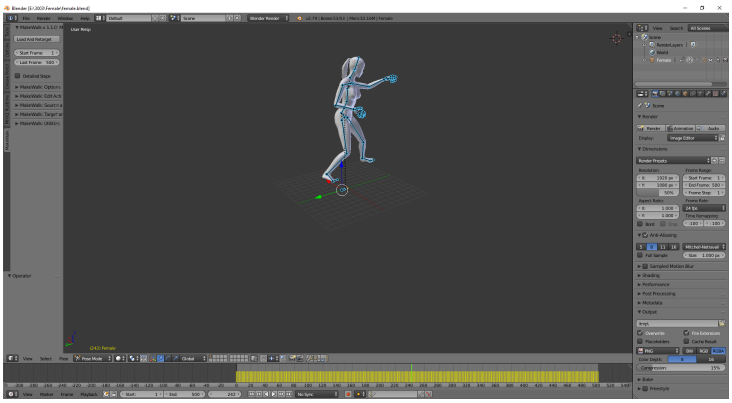

Figure 5: model58.x

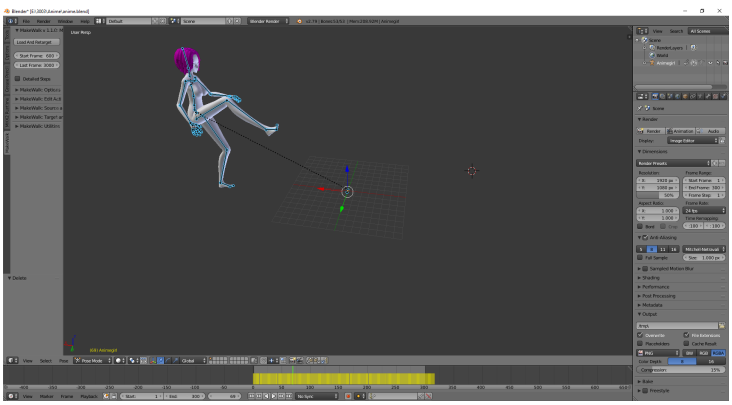

Figure 6: model59.x

3.13 Task 2.D

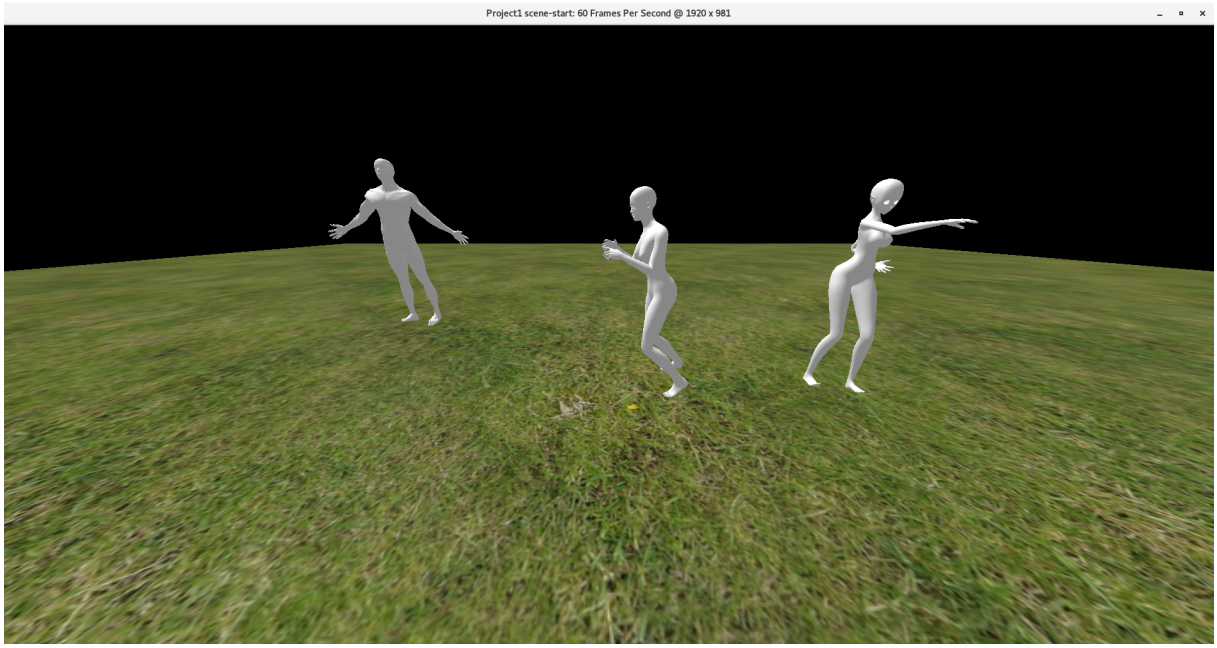

Figure 7: Animation Screenshot

gnatidread.h

```
13. // Increased the number of meshes
14. const int numMeshes = 60;
```

gnatidread.h

```
109. // Added new mesh entries in the menu
110. "57 (Added) Male", "58 (Added) Female", "59 (Added) Anime"};
```

scene-start.cpp

```
330. // Scaled the human models by a factor of 10x to have right size
331. if (id > 0 && id < 55)
332. {
333.   sceneObjs[nObjects].scale = 0.005;
334. }
335. else if (id > 55)
336. {
337.   sceneObjs[nObjects].scale = 0.05;
338. }
```

scene-start.cpp

```
352. // Rotated the models to face the right way when added to scene
353. if (id > 55 && id < 59)
354. {
355.   // Human models turning right way
356.   sceneObjs[nObjects].angles[0] = 0.0;
357.   sceneObjs[nObjects].angles[1] = 0.0;
358.   sceneObjs[nObjects].angles[2] = 0.0;
359. }
360. else if (id == 59)
361. {
362.   // Anime model turning right way
363.   sceneObjs[nObjects].angles[0] = 0.0;
364.   sceneObjs[nObjects].angles[1] = 90.0;
365.   sceneObjs[nObjects].angles[2] = 0.0;
366. }
367. else
368. {
369.   // All other base models
370.   sceneObjs[nObjects].angles[0] = 0.0;
371.   sceneObjs[nObjects].angles[1] = 180.0;
372.   sceneObjs[nObjects].angles[2] = 0.0;
373. }
```

vStart.glsl

```
28. // Animation of bones
29. vec4 vpos = uBoneTransform * vec4(vPosition, 1.0);
30. vec4 vnorm = uBoneTransform * vec4(vNormal, 0.0);
```

scene-start.cpp

```
82. // Added frames counter for animation of models
83. int frames;
```

scene-start.cpp

```
95. // Declared variables for animation of model
96. // Animation Speed begins at 4 to allow for adjustment within menu
97. int frame = 0;
98. int animationSpeed = 4;
99. bool animationActive = true;
```

scene-start.cpp

```
381. // Added custom frame length for each animated human model
382. if (id < 56)
383. {
384.   sceneObjs[nObjects].frames = 1;
385. }
386. else if (id == 56)
387. {
388.   sceneObjs[nObjects].frames = 300;
389. }
390. else if (id == 57)
391. {
392.   sceneObjs[nObjects].frames = 300;
393. }
394. else if (id == 58)
395. {
396.   sceneObjs[nObjects].frames = 500;
397. }
398. else if (id == 59)
399. {
400.   sceneObjs[nObjects].frames = 300;
401. }
```

scene-start.cpp

```
534. // Adjusting animation speed through animationSpeed variable
535. if (animationActive)
536. {
537.   // '10' used to allow tweaking of the animation speed
538.   frame = glutGet(GLUT_ELAPSED_TIME) / (10 * animationSpeed)
% sceneObj.frames;
539. }
```

scene-start.cpp

```
543. // Added frame variable to keep track of position time
544. calculateAnimPose(meshes[sceneObj.meshId], scenes[sceneObj.meshId], 0,
frame, boneTransforms);
```

scene-start.cpp

```
817. // Added new menu to allow for animation tweaking
818. static void animationMenu(int id)
819. {
820.   // Pause and Play option
821.   if (id == 100)
822.   {
823.     animationActive = !animationActive;
824.   }
825.   // Speed Down option
826.   else if (id == 101)
827.   {
828.     animationSpeed += 1;
829.   }
830.   // Speed Up option
831.   else if (id == 102)
832.   {
833.     // Limit the speed to above 1 so it cannot crash
834.     if (animationSpeed > 1)
835.     {
836.       animationSpeed -= 1;
837.     }
838.   }
839.   // Speed Reset option
840.   else if (id == 103)
841.   {
842.     animationSpeed = 4;
843.   }
844. }
```

scene-start.cpp

```
869. // Added sub-menu entry for Animation
870. int animationMenuId = glutCreateMenu(animationMenu);
871. glutAddMenuEntry("Pause / Play", 100);
872. glutAddMenuEntry("Speed Down", 101);
873. glutAddMenuEntry("Speed Up", 102);
874. glutAddMenuEntry("Speed Reset", 103);
```

scene-start.cpp

```
887. // Added Animation main menu entry
888. glutAddSubMenu("Animation", animationMenuId);
```