COSC 422 Assignment 3

Skeleton Makeover

Important Dates

- Due date:

  Wednesday, 20th October 11:55pm

  Late penalty:

  21st Oct – 24th Oct: 1 Mark

  25th Oct – 27th Oct: 2 Marks

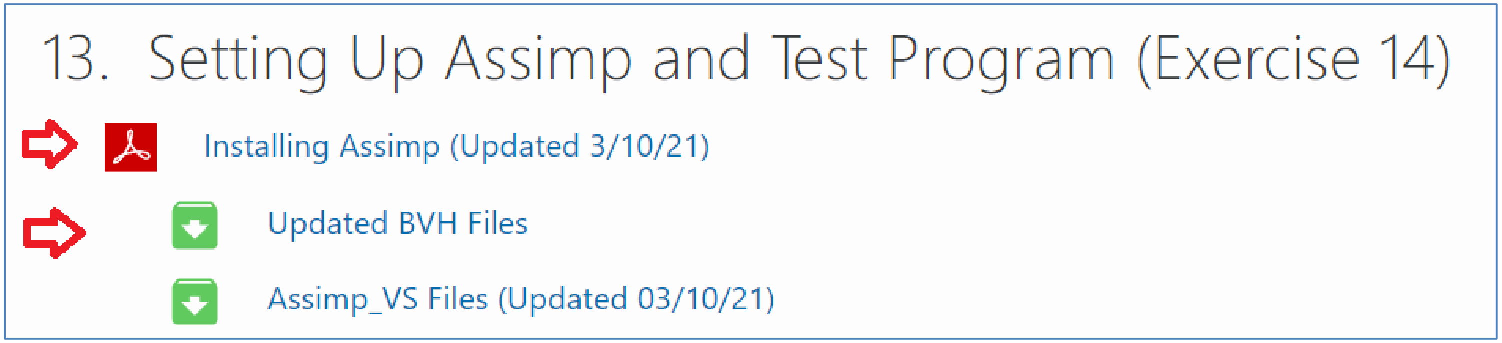
- Drop-dead date:

  Wednesday, 27th October

BVH Rotation Error in Assimp

- https://github.com/assimp/assimp/pull/3233

- Please go through the notes in the Programming Exercises section ("13. Setting Up Assimp") before installing Assimp



13. Setting Up Assimp and Test Program (Exercise 14)

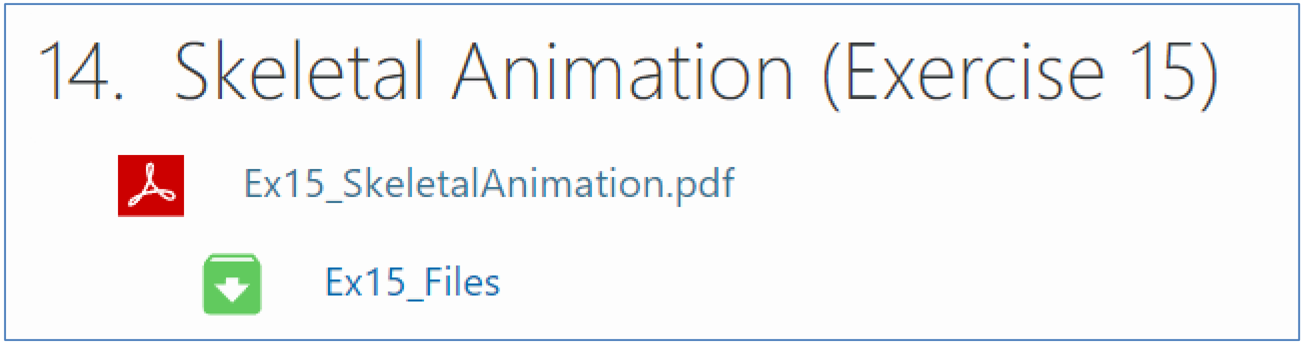
Installing Assimp (Updated 3/10/21)

Updated BVH Files

Assimp_VS Files (Updated 03/10/21)

Extending SkeletalAnimation.cpp

- Use SkeletalAnimation.cpp as the base code, and extend it to include a keyframe animation of the skeletal model.
- Please follow the steps outlined in lecture slides [10]:33-35.



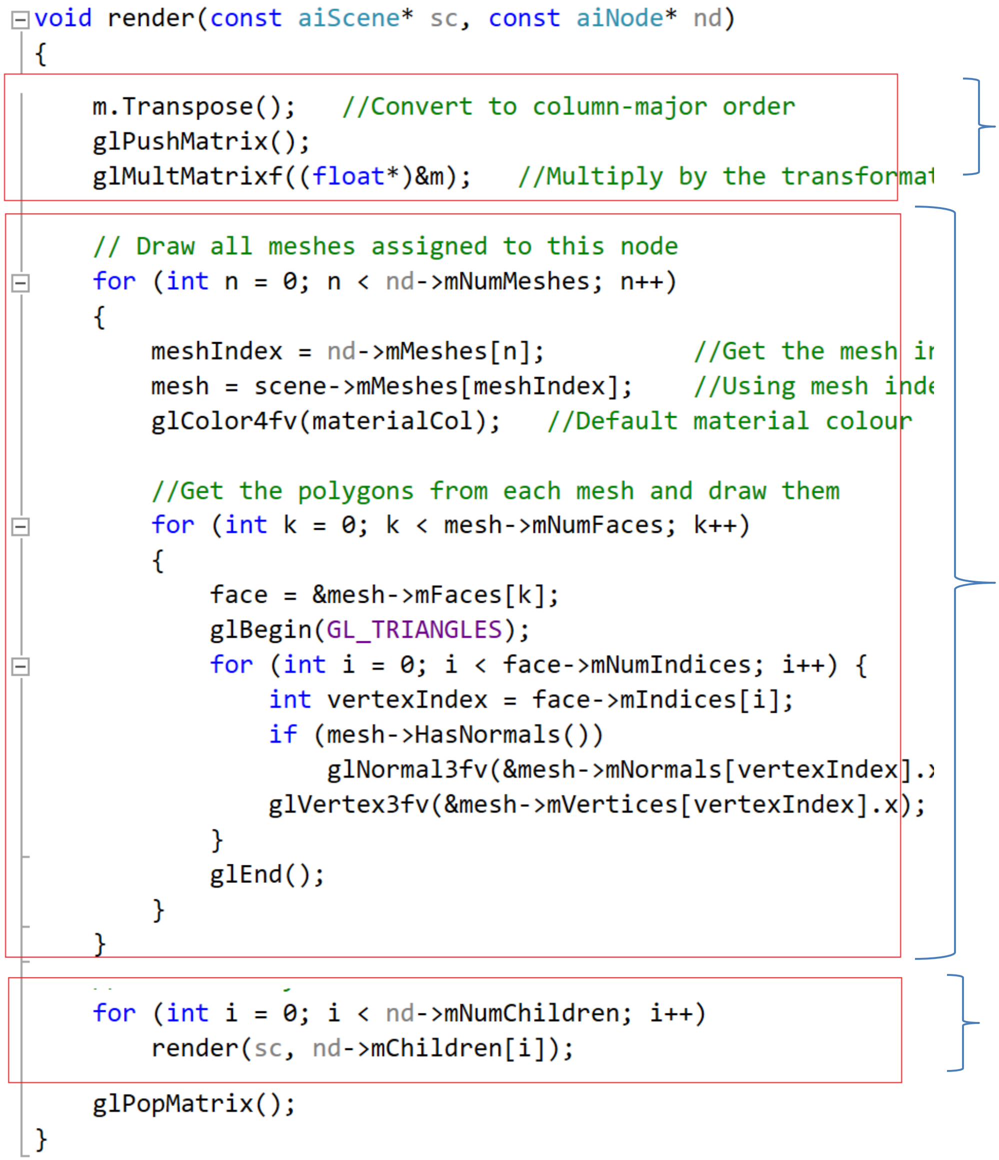
14. Skeletal Animation (Exercise 15)

Ex15_SkeletalAnimation.pdf

Ex15_Files

Modifying Skeleton Mesh

- The render() function consists of three parts:



```
void render(const aiScene* sc, const aiNode* nd)
{
    m.Transpose();   //Convert to column-major order
    glPushMatrix();
    glMultMatrixf((float*)&m);   //Multiply by the transformat

    // Draw all meshes assigned to this node
    for (int n = 0; n < nd->mNumMeshes; n++)
    {
        meshIndex = nd->mMeshes[n];          //Get the mesh ir
        mesh = scene->mMeshes[meshIndex];    //Using mesh inde
        glColor4fv(materialCol);   //Default material colour

        //Get the polygons from each mesh and draw them
        for (int k = 0; k < mesh->mNumFaces; k++)
        {
            face = &mesh->mFaces[k];
            glBegin(GL_TRIANGLES);
            for (int i = 0; i < face->mNumIndices; i++) {
                int vertexIndex = face->mIndices[i];
                if (mesh->HasNormals())
                    glNormal3fv(&mesh->mNormals[vertexIndex].x
                glVertex3fv(&mesh->mVertices[vertexIndex].x);
            }
            glEnd();
        }
    }

    for (int i = 0; i < nd->mNumChildren; i++)
        render(sc, nd->mChildren[i]);

    glPopMatrix();
}
```

A: Transformation

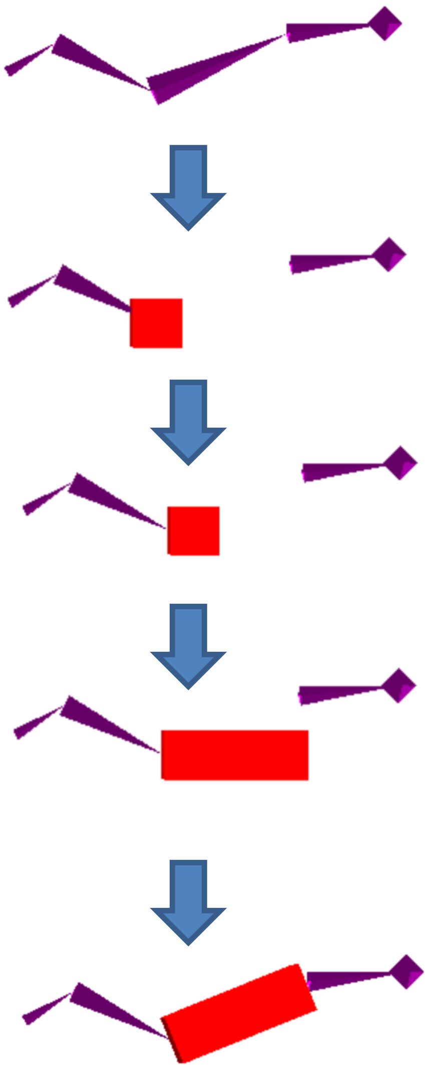
B: Rendering a skeleton mesh

C: Recursion

Replacing a Link

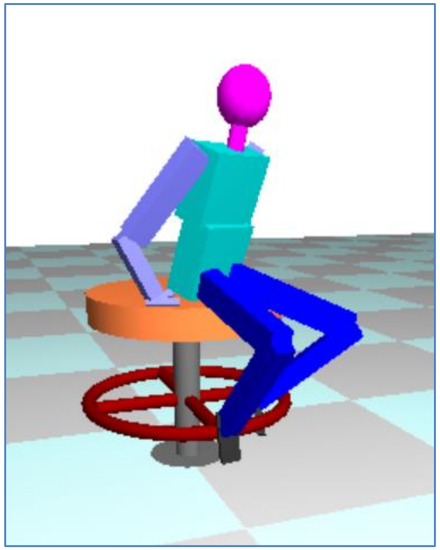
```
if ((strcmp((nd->mName).data, "Elbow") == 0))
{
    glPushMatrix();
    glColor3f(1, 0, 0);
    glutSolidCube(0.5);
    glPopMatrix();
}
else
{
    // Draw all meshes assigned to this node
    for (int n = 0; n < nd->mNumMeshes; n++)
    {
        meshIndex = nd->mMeshes[n];
        mesh = scene->mMeshes[meshIndex];
        glColor4fv(materialCol);

        for (int k = 0; k < mesh->mNumFaces; k++)
        {
            face = &mesh->mFaces[k];
            glBegin(GL_TRIANGLES);
            for (int i = 0; i < face->mNumIndices; i++) {
                int vertexIndex = face->mIndices[i];
                if (mesh->HasNormals())
                    glNormal3fv(&mesh->mNormals[vertexIndex].x);
                glVertex3fv(&mesh->mVertices[vertexIndex].x);
            }
            glEnd();
        }
    }
}
```

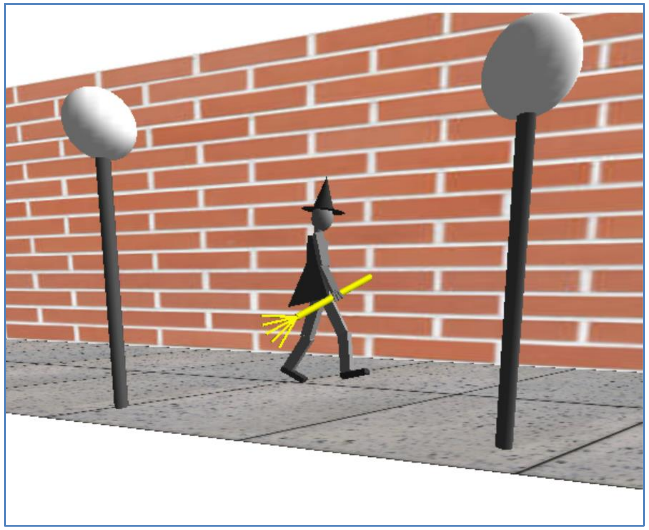


Examples

13_01.bvh

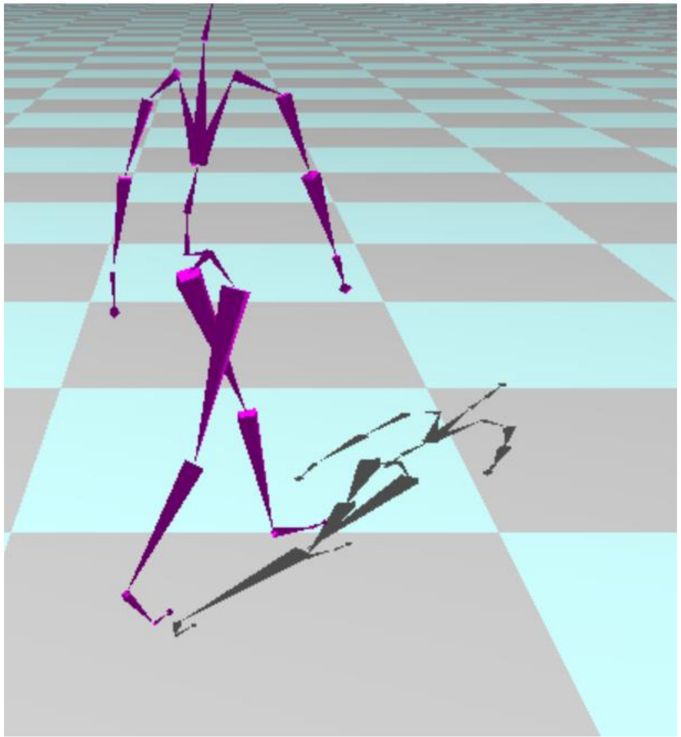


Walk.bvh



Extra Features

- Planar shadows: Use a projection transformation (ref COSC363 slides) to generate shadows on the floor plane.

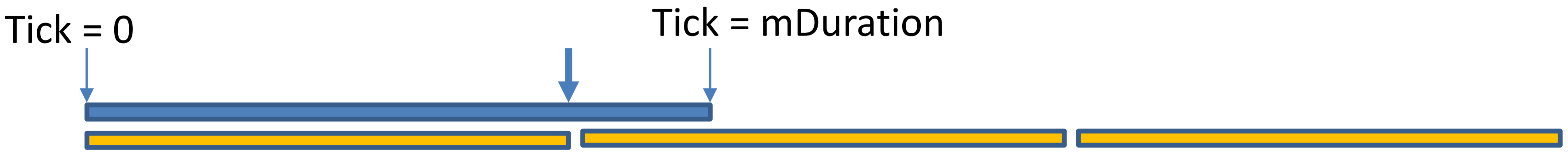


Extra Features

- Tracking movement: The position of the skeleton in world space is given by either of the following:
  - The root channel’s position key
    - `(scene->mAnimations[0]->mChannels[0]->mPositionKeys[k]).mValue`
  - The translation component of the root node’s transformation matrix
    - `scene-> mRootNode->mTransformation`

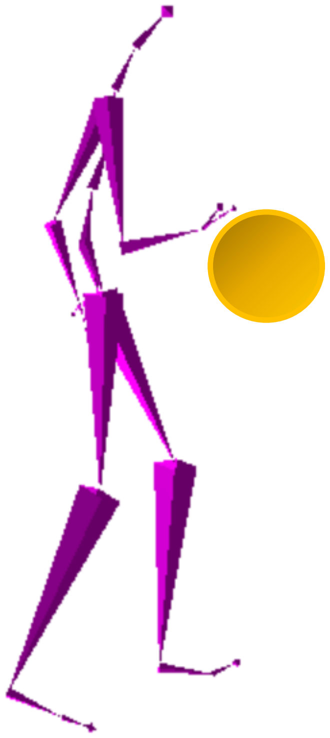
Extra Features

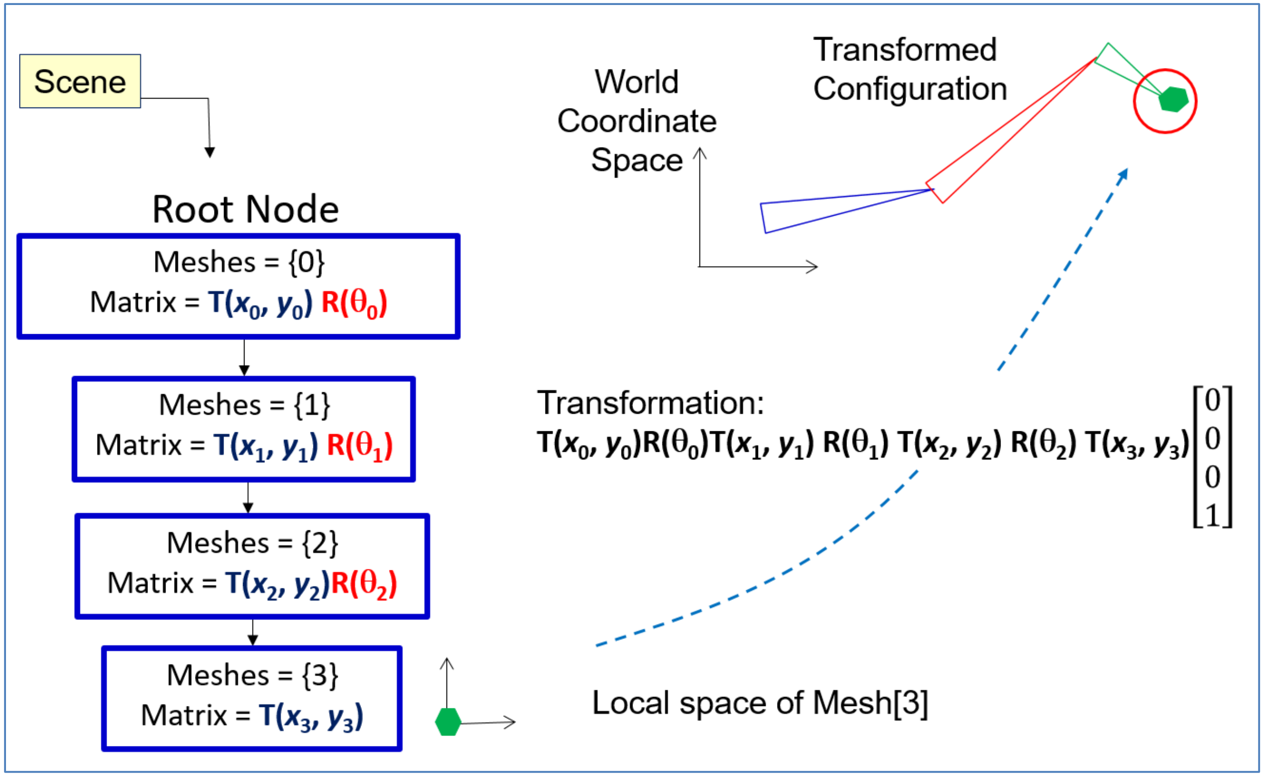
- Camera view distance and orientation
- Physics based motion
- Animation looping
  - Select a keyframe from the end of the animation segment that has key rotation angles similar to the first key frame.
  - When the skeleton reaches the end of a segment, adjust the root node's position with an offset so that the skeleton's first keyframe is displayed at that position.



Extra Features

- Hand/foot position tracking





Scene

Root Node

Meshes = {0}
Matrix = $\mathbf{T}(x_0, y_0)\ \mathbf{R}(\theta_0)$

Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)\mathbf{R}(\theta_2)$

Meshes = {3}
Matrix = $\mathbf{T}(x_3, y_3)$

World Coordinate Space

Transformed Configuration

Transformation:

$$\mathbf{T}(x_0, y_0)\mathbf{R}(\theta_0)\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)\ \mathbf{T}(x_2, y_2)\ \mathbf{R}(\theta_2)\ \mathbf{T}(x_3, y_3)\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Local space of Mesh[3]