COMP 371 – Computer Graphics

Assignment 1: Graphics Pipeline and Simple Animation

Due: **Wednesday, July 10th 2019 , End of Day**
Worth: 10% of your final grade

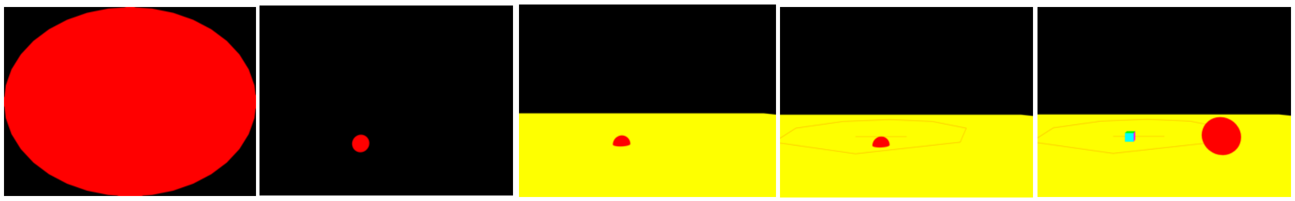
Topics

- World Transform (Translation, Rotation, Scaling)
- View Transform (Camera positioning, Change of basis)
- Draw Calls in OpenGL (Line drawing)
- Animation of Models along Key Frames (Animation Keys)

Non-Programming Assignment [2 points]
(You can do it by hand with readable hand writing, and scan it with your assignment, or you can do it with a Word processor and submit it in PDF or DOCX format).

- Explain the World Transform and derive the 4x4 Matrix which includes Translation, Rotation and Scaling. Show that the order of transformations is important **[1 point]**
- Explain the View Transform and derive the 4x4 Matrix from the Camera position and $\overrightarrow{lookAt}$ vector **[1 point]**

Programming Assignment [8 points]



Look in Code for // @TODO

Images above are what you should get if you properly implement each one of the tasks.

Images from left to right: initial framework (1st), after implementing the ViewProjection transform (2nd), after implementing the World Transform (3rd), after drawing the animation keys (4th), after animating the cube along the key frames (5th).

Tasks to Do:

- View Transform **[2 point]**
  - In the Camera class, implements the ViewProjection transform, the projection transform is already implemented, and the view transform is implemented in the Static Camera class
- World Transform – Translate, Rotate, Scale Cube and Sphere Models **[2 points]**
  - Set the World Transform in the Model class based on the position, rotation and scaling of the Model loaded from a Scene file
- Drawing Animation Key Positions **[2 points]**
  - In the Path class, the draw function is empty. Implement the draw call by using a GL_LINE_LOOP primitive. The vertex buffer is already created, the shader is set to the PathLine Shader. You only need to set the shader inputs and constants and implement the draw call.
- Animating models between animation keys (aka waypoints or key frames) **[2 points]**
  - If a model contains an animation, the model should animate along the animation keys of that animation. Update the position of models every frame towards the target key position. When you reach the target key, you should start aiming to the next key position in the animation. After you reach the last animation key, you aim for the first.

Scene Representation

- Data driven Framework
  - The framework will load scene into the virtual world through .scene files
  - You can find CoordinateSystem.scene and AnimatedScene.scene in the Scene folder
- Scene file specifications
  - The character # at the beginning of a line specifies a comment
  - Cube models are loaded from lines containing [Cube]
  - Sphere models are loaded from lines containing [Sphere]
  - Each model can have a

▪ Position:	position = x y z	(x, y and z are numeric values)
▪ Scaling:	scaling = x y z	(x, y and z are numeric values)
▪ Rotation:	rotation = axisX axisY axisZ angleInDegrees	
▪ Name:	name = "name of the model"	
▪ Animation:	name = "name of the animation the model will follow"	

  - Check the scene files for examples

Programming Framework

- <u>Main</u>
  - Initialize/Shutdown the program
  - Implement the Main Loop
- <u>Event Manager</u>
  - Create Window
  - Setup OpenGL Context
  - Calculate frame time
  - Handle mouse and keyboard inputs
- <u>Renderer</u>
  - Set OpenGL states
  - Load shaders
  - OpenGL code that should be executed at the beginning/end of a frame
- <u>World</u>
  - Contain all the Models and Cameras in the Virtual World
  - Update and Draw all of these objects
  - You can press [1] and [2] to change the Camera
  - You can press [9] and [10] to change the current Shader
- <u>Camera</u>
  - Base class to set a virtual Camera
  - Interface for the view and projection transforms
- <u>Static Camera</u>
  - Non moving Camera looking towards a look at point
- <u>Model</u>
  - Base class for all the Model objects, update and draw them
- <u>Cube Model</u>
  - Create cube of size specified in the constructor (default 1x1x1)
  - Creates vertex buffer and draw the cube at the Origin
- <u>Sphere Model</u>
  - Create sphere of size specified in the constructor (default 1x1x1)
  - Creates vertex buffer and draw the sphere at the Origin
- Animation
  - Contains a list of animation keys.
  - Draw a line between the animation key positions (for you to implement)
  - Interpolate world transform according to the current time and matching animation keys (each animation key has a timestamp) (for you to implement).

Submission Guidelines

- You need to submit your assignment on Moodle by due date, before midnight.
- Late submission penalty is 5% per hour late.
- For the non-programming assignment, you will need to scan your readable hand written derivations, or include the PDF or DOCX word processed file.
- For the programming assignment, you should delete the Bin and Build folders before submitting your completed assignment.
- You must also print, sign, scan and include the Concordia's Honor Code available on http://www.concordia.ca/content/dam/ginacody/docs/Expectations-of-Originality-Feb14-2012.pdf
- Everything must be submitted within a single ZIP file on Moodle.
- Anything missing from the Submission will result in grade deductions.