Program #3 Path Length **11.25%**  
Due: 11/09/2016  
Assignment:

In computer graphics, the camera is often moved through a scene to generate a "fly-through" animation. In order that the animation appears smooth, the camera must trace out the same distance between each frame. This is no problem if the camera is moving in a straight line. However, if the camera trajectory is complicated, moving the camera equal distances between frames can be difficult.

In general, the camera path can be specified by three unrelated equations, an equation to determine the x-coordinate of the camera, the y-coordinate, and the z-coordinate. Each of these equations depends on a single parameter, **u**. Thus, even though the camera is moving through three dimensions, it traces out a 1D curve in 3D space.

To find the length of an arbitrary 1D path, you first need the "instantaneous velocity" (the velocity at a specific point on the path). Rather than time, however, the velocity is obtained by how fast the position changes with a change in the parameter u. If the camera's x-coordinate is the only coordinate that changes with a change in u, the instantaneous velocity is dx(u)/du. In addition, since you are interested in the path length and not the location for this computation, you only need the magnitude of the velocity, |dx(u)/du|. If all three coordinates change with a change in u, the magnitude of the instantaneous velocity is determined by:  
√(dx(u)/du)2 + (dy(u)/du)2 + (dz(u)/du)2, where the entire expression is under the radical

To get the length of the path from the starting u value (ul) to the ending u value (uh), you need to integrate the above instantaneous velocity from ul to uh:  
length s = ∫uluh √(dx/du)2 + (dy/du)2 + (dz/du)2 du

Thus, if the animator knows ul (where the camera is right now), the animator must determine **uh** so that the distance traveled by the camera between ul and uh is a **specified distance s**, possibly the same distance traveled by the camera in each of the other frames of the animation. Once uh is known, the location of the camera can be determined by plugging uh into x(u), y(u), and z(u).

The first problem is the fact that for arbitrary functions of the parameter u, the above integral cannot be determined analytically. Further, it is difficult to identify the correct value for uh so that the integral gives the specified path length s. Therefore, there are three numerical parts to this assignment.

**Numerical Differentiation**

Use the **centered difference** approximation for the first derivative to find dx(u)/du, dy(u)/du, and dz(u)/du.

**Numerical Integration**

Use **Romberg integration to level 5 (Romberg.cpp)** to compute the path length (evaluate the above integral) between ul and uh for arbitrary functions x(u), y(u), and z(u).

* double accurateRomberg(Function\* f, double\* a, double\* b, int level, int numVars)

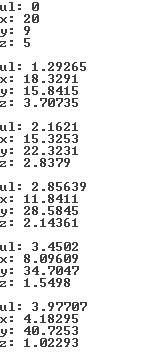
**Root Finding**

Use the **modified secant method (ModifiedSecant.cpp)** to iteratively refine uh so that the path length is the user specified path length s.

* double\* modifiedSecant(Function\* f, double x, double d, double tol, int max)

**Camera Location for Each Animation Frame**

Repeat for each frame of the animation, where uh becomes ul in the next frame.



The above image shows my result for the PathLengthDriver.cpp example provided below. The location of the camera for each frame of the animation is shown. Each iteration advances the camera along the path by 7.2.

* C++ files:
  + [**prog3.zip**](http://mboshart.dyndns.org/~mboshart/3020programs/prog3.zip)