Shortest path finder proposal

I want to calculate the best path to take when walking to class, or when hiking up a mountain. The shortest absolute path between two points is obviously a straight line, but that path is usually not possible because of buildings or steep slopes. Also, a short, jagged path is not necessarily the best path because it takes some energy to change direction quickly. I plan to make a program that will calculate the best path to take, given an environment and some parameters such as maximum allowed slope and desired speed.

Data description

World	
Data members	
type and name	*note*
Points surface[][]	A 2D array of point objects describing the surface of the environment, such as a mountain
Member Functions	
Constructor: World(ifstream)	On creation, read in environment from a file. Data verification also happens here.
Curve: bestPath(tolerance)	Iterate math to find best path within certain tolerance based on surface properties

Point	
Data members	
float height	Height of surface at current point
char texture	Texture or roughness of surface determines how easy it is to walk on. This could also contain special cases such as start/end points and walls.

Curve	
Data members	
string Equation	Math representation of a curve
Member Functions	
Points approximateToPoints()	Approximate equation to the set of points closest to it, in order to make it easier to display
float findWork()	Find Work done to travel that curve, see Calculate best path on page 2
float equationFromPoints()	Approximate an equation from a set of connected points

Procedure description

1. Read in environment:
    - **File input**: the environment will be externally stored in a file consisting of a grid of points representing the surface of the environment.
    - Open this file containing the environment
    - Verify validity of file
    - Extract relevant info, including start/end points, obstacles, surface, and user parameters. These values are stored into the World class.
2. Calculate best path
    - Generate a guess curve to test
    - Find Work:
        i. Find length of the curve $L=\int\sqrt{1+(f'(x))^2}dx$
        ii. Find vertical slope (in 3D space) at each position of the curve
        iii. Find centripetal acceleration at each point on the curve (2nd derivative of curve)
        iv. Calculate work required to travel that curve $W=\int_0^L aFdx$
    - Iterate these steps to find the curve of least work
3. Output the path.
    - Display the path onto the inputted environment (possibly in an SFML graphics window).
    - Print the equation of the curve.

Concerns

Finding the best path requires generating an initial guess curve, and that the guesses approach the best curve. Calculating the Work for each curve itself is also a fairly complex math equation. Additionally, the input environment file may have to be very complex to be able to hold enough data to get a useful result. I will try to find a prebuilt topographic map from which to extract this data.