Snell's law and light traveling along the shortest path

Carlos Lara

September 18th 2006

Supervisors : Mario A. Lopez and Petr Votchevosjky

Abstract

the problem to be analyzed follows: Given a starting point s, an ending point t and a set of n Weighted Faces (or regions) in a 2-dimensional space, find the best path from s to t, where the length of the path is defined as the weighted sum of the Euclidean length of the sub paths inside each region. Let α_f be the weight of f, with $\alpha_f > 0$. Then walking through each face will have different cost depending on the α_n associated with each face.

The weighted region problem is solved by [MP91] locally by the use of the Snell's law. This law determines the refraction of light, the phenomenon of the change in direction experienced by a beam of light when it crosses the frontier between one transparent medium into another (for example from air into water, or into glass). The solution given by Mitchell and Papadimitriou in 1991 is analyzed as it sets an analogy between shortest paths and the optical path (light's path). However this optics analogy has its limits, when a path hits a region in some specific ways, the results cannot be predicted by Snell's Law. In fact Snell's law can be violated $O(n)$ times, where n is the number of faces (example provided by P.h.D. student, Sada Narayanappa). Finally, A simulation of Snell's law is provided in order to visualize what a direction a light beam would successively take given a set of user-defined polygons, that have user-defined weights.

Contents

Introduction

Since 1996, GPS (Global Positioning Systems) have been available to civilians as well as military providing a great tool to know one's exact geographical position in the planet at any given time. Much software takes advantage of features offered by GPS to direct you step by step to your destination, in an urbanized area. What if one gets lost in a non-urbanized area, and has to choose from crossing a river or going through a far away bridge, would these pieces of software, for instance automotive navigation system be able to direct you in that case ?

The answer is no, and the reason being that input and output of problem changes. In fact, both input and output sets increase dramatically. First of all, automotive navigation systems have a discreet solution set, as roads in a region is a small discreet set. However, if you happen to be walking in a mountain, there are infinitely many ways to get from point A to point B. if this was the only difference in the output set, solving shortest paths in non-urbanized areas would be easier than urbanized areas, as the solution would always be the straight line. However there is a difference in the input set as well. Input set for automotive navigation includes a set of streets. When traveling in an urbanized area, if the Euclidian distance of two paths is the same then the units of time spent traveling along those paths are the same. However in a non-urbanized area, the actual length of two paths that share the same Euclidean distance, can differ enormously, as traveling along grass, a river, concrete, dust, or a mountain require different effort, and different traveling time.

The purpose of this thesis project is to analyze how *The Weighted Region Problem* is solved, to explain some of its limits and provide a way to visualize these limits. For this task, we will define and illustrate the weighted region problem; then we will explain how mathematics and optics are involved in solving a geographical problem, namely using Fermat's principle that light travels along the shortest path. This gives enough background to survey the solution given to WRP by Mitchell and Papadimitriou. Can a geographical/geometrical problem be solved by optics and laws of refraction? How does light know from which point to which point we want to go? After that we will explain some issues about how the problem is being solved, namely using optics to find shortest paths. Finally we provide a way to simulate the path that light would take on a user-defined polygonal meshes.

The problem to be approached follows. Given a starting point s, an ending point t and

a set of n Weighted Regions $f_0, f_1, ..., f_n$ in a 2-dimensional space, we need to find the best path from s to t, where the length of the path is defined as the weighted sum of the Euclidean length of the sub paths inside each region. Let α_f be the weight of f, with $\alpha_f > 0$. Then the cost of traveling in a region is a function of the distance traveled and the weight of that region. Common sense tells us that as an area becomes expensive, the path should avoid going through there, and if it must, minimize the amount traveled on this area. As a result, when a path enters cheaper or more expensive regions, its direction changes. This change will be studied through out this paper.

Here is a graphical interpretation of the problem with the simplest set of inputs.

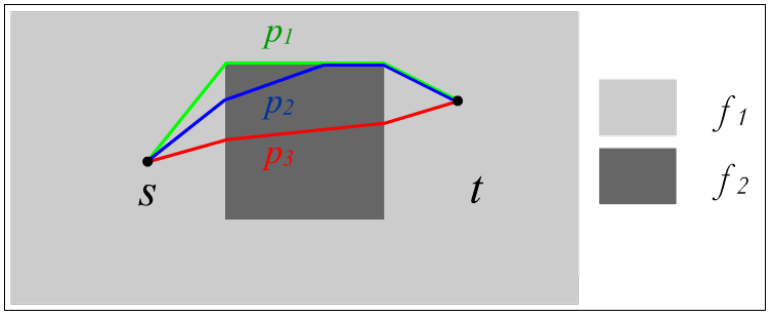

Figure 1: Simple visualization of WRP

Depending on the weight of the Region f and f', the solution paths will differ depending on these conditions :
- if $\alpha_{f'} \gg \alpha_f$, then $Sol = p1$, the path will aim for a line boarding f_2 .
- if $\alpha_{f'} \leq \alpha_f$, then $Sol = p3$, the path will strive for a straight line.
- if $\alpha_{f'} > \alpha_f$, then $Sol \simeq p2$, the path will target a middle ground between the two earlier solutions.

The weighted faces could be identified to lakes, woods, or road and many others; in this context, the problem has many applications such as finding the best way to get to any point while walking on a non-urbanized area, or finding the closest water fountain or lumber area while camping. It can also be applied in other situations such as building a highway, railway or any other transport system located in vertically uneven terrain (the weight of the region will increase as the slope increases).

1 History on Snell's Law

1.1 Snell's Law

The weighted region problem is solved locally by the use of the Snell's law which determines the refraction of light, the phenomenon of the change in direction experienced by a beam of light when it crosses the frontier between one transparent medium into another (for example from air into water, or into glass). This change of direction is directly connected with the change in velocity experienced by light from medium to medium. The change in velocity is due to the fact that light travels slower in denser mediums, and faster in less dense mediums. Mathematical interpretation of refraction goes back to Ptolemy 2nd c A.C., and later Johannes Kepler in 1611, and Thomas Harriot (1560 - 1621). It was not until 1621 that the Dutch physicist Willebrord Snel (1580 - 1626) discovered the precise law of refraction (without publishing)[NAHIN04]:

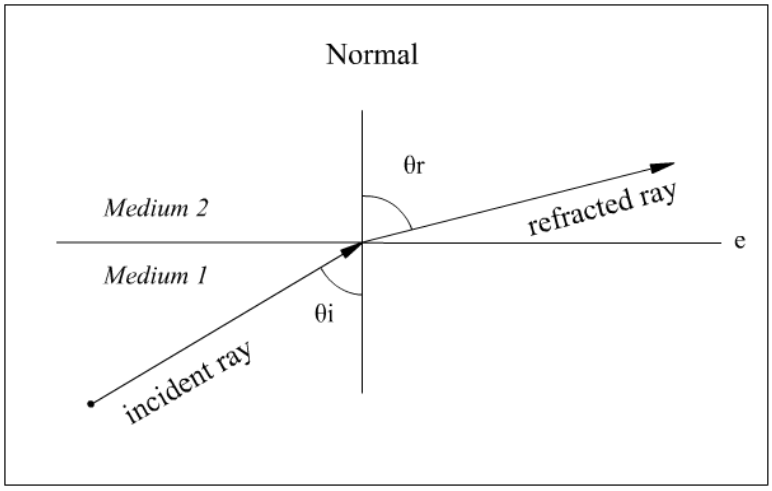

Figure 2: Snell's law and light refraction [NAHIN04]

According to Snel , $\frac{sin(\Theta_i)}{sin(\Theta_r)} = constant$, where the angle of incidence Θ_i is the counterclockwise angle between the incident ray and the normal at the region boundary and the angle of refraction Θ_r is the counterclockwise angle between the refracted ray and the normal. the *constant* is a function of the indices of refraction n_1, n_2. The index of refraction (or refractive index) of a material is the factor by which the phase velocity of electromagnetic radiation is slowed in that material, relative to its velocity in a vacuum. It is a function of the material's relative permittivity (ability to transmit an electric field) and permeability (how a material is affected by magnetic fields).

This law means that as light's velocity decreases, it veers towards the normal, therefore, if the velocity of light in *Medium2* is smaller than the velocity of light in *Medium1*, then the angle of incidence is also smaller than the angle of refraction (*see Fig 2*).

The *constant* was first determined incorrectly by Descartes in *laDioptrique* (1630). It is the first published derivation of the law of refraction; The constant that Descartes had derivated was actually incorrect. In fact Descartes had derived the exact opposite of the actual solution: $constant = \frac{v'}{v}$; therefore he was not far, but not correct . This would mean that as light slows down from medium to medium , light would veer away from the normal. However it was seen above that light does the exact opposite, it veers towards the normal when entering denser medium. To bring his results to something coherent, he concluded that light speeds up as it penetrates through a denser medium. Fermat was very unimpressed with Descartes derivation, in fact he commented: "of all the infinite ways [to analyze the motion of light] the author [Descartes] has taken only that one which serves him for his conclusion; he has thereby accommodated his means to his end, and we know as little about the subject as we did before."[NAHIN04] Nowadays speed of light can actually be computed experimentally so Fermat has been proven correct empirically but in the days of Descartes and Fermat, the speed of light in different media could only be determined theoretically. Fermat was puzzled about how light can speed up on a denser media so he was determined to prove the contrary [NAHIN04].

1.2 Snell's law and Fermat's least time principle

Fermat generalized the known fact that light reflects according to a minimum-path-length criterion and argued that the path that light takes is, for both reflection and refraction, the path of minimum time. If light is always in the same medium, in the case of reflection, the path of minimum length in space and time turn out to be the same. However in refraction, the paths are different. In this case, the actual path that light takes is the path of less time.

Fermat used the observation that on extrema (minimum or maximum), the value of a function does not change much as x changes by a small value. So if we want to find $x*$ so that x is an extrema, then we can set :

$f(x^* + \varepsilon) \approx f(x^*)$ and then, $f(x^* + \varepsilon) - f(x^*) \approx 0$ as ε becomes infinitely small.

more formally, Fermat was hinting at the modern version of the derivative, which is :

if $[\lim_{x \to x^*} f(x) + \varepsilon - f(x) = 0]$, then $x = x^*$ is an extrema for $f(x)$

In our instance we wish to minimize the distance traveled from (0,-y0) to (x1,y1) (*see Fig3*):

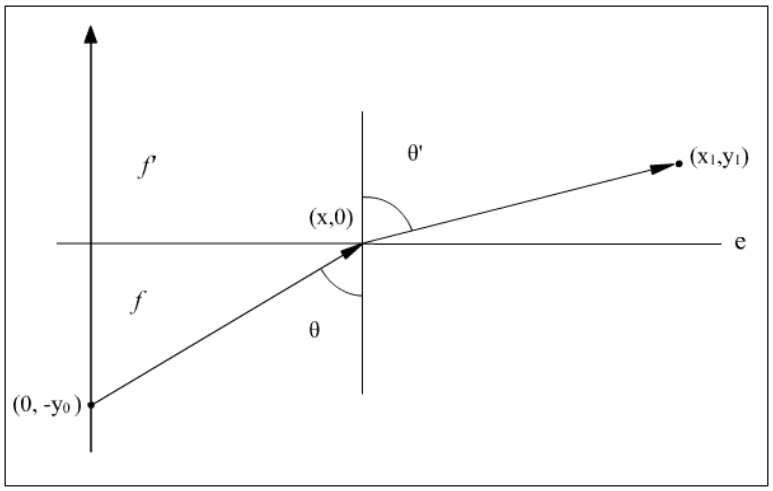

Figure 3: Snell's Law of refraction and shortest paths [MP91]

$f(x) = \frac{1}{v}\sqrt{x^2+y_0^2} + \frac{1}{v'}\sqrt{(x_1-x)^2+y_1^2}$

Fermat actually solved this problem without using the modern concept of derivatives, but hinting closely at it.

The detail of his calculations can be seen at [NAHIN04] and are quite lengthy and unprecise, as they use some approximations ($\sqrt{1+u} \simeq 1 + \frac{1}{2}u$ *for small values of* u). We will solve this by setting $f'(x) = 0$

First note that $\alpha_f = 1/v$, as the weight of face f varies inversely proportional to the velocity of light in face f and also note that $\sin\theta = \frac{x}{\sqrt{x^2+y_0^2}}$ and $\sin\theta' = \frac{(x-x_1)}{\sqrt{(x_1-x)^2+y_1^2}}$

$F(x) = \alpha_f\sqrt{x^2+y_0^2} + \alpha_{f'}\sqrt{(x_1-x)^2+y_1^2}$ and $F'(x) = \alpha_f\frac{2x}{2\sqrt{x^2+y_0^2}} + \alpha_{f'}\frac{-2(x-x_1)}{2\sqrt{(x_1-x)^2+y_1^2}}$

$F'(x) = \alpha_f\frac{x}{\sqrt{x^2+y_0^2}} - \alpha_{f'}\frac{(x-x_1)}{\sqrt{(x_1-x)^2+y_1^2}} \Rightarrow F'(x) = \alpha_f\sin\theta - \alpha_{f'}\sin\theta$

so, $F'(x) = 0 \Rightarrow \alpha_f\sin\theta = \alpha_{f'}\sin\theta' \Rightarrow \frac{\sin\theta}{\sin\theta'} = \frac{\alpha_{f'}}{\alpha_f} = \frac{v}{v'} = constant$!

The exact opposite of what Descartes had derivated !

2 Case study : Vector approach to WRP (Mitchel and Papadimitriou)

There are two known methods to solve this problem. The first one is a vector method designed by Mitchell and Papadimitriou, the second one is a raster method; The later only gives an approximation of the solution and not the optimal path: A grid is super imposed to the plane and weights are assigned to each cell. Then the Dijkstra graph search algorithm is used to find the optimal path. The precision of raster-based solutions depends on how well the grid first the regions (of course smaller cells carry increasing precision) and on the number of directions where the point can move. [BEMMELEN93]. According to [MP91] the problem with such an approach is that it could require very fine grids to capture the content of a simple map, and it creates a bias as the movement is limited to four, eight or sixteen directions. The precision on the vector based solution is actually set by the user, by setting a precision variable.

We will analyze the Vector method proposed by [MP91] as it uses the Snell's law to direct the paths along a polygonal subdivision. This lets the algorithm use a continuous paradigm. Particularly, the continuous Dijkstra technique. This technique was first developed by Mitchell [MMP87] to solve the Discrete Geodesic Problem. This is a very similar but simpler problem, as the difference lies in the input: all weights of the polygonal subdivision are the same.

2.1 Input and conventions

the input of the problem is a plane that is subdivided into polygonal regions forming a polygonal subdivision ϑ. Each polygon is associated with a positive integer weight α which specifies the "cost per unit distance" of traveling in that region. Without loss of generality, the polygons are triangles, and these are computed using a CDT (Constrained Delaunay Triangulation). The CDT is composed by a set of triangles or faces, edges, and vertices which parameters are specified in integers. Similar to a face, an edge e also has a positive integer weight α_e. The notion of length is determined by a weighted Euclidean metric. The weighted length of a line that joins two points x and y is the product $\alpha_f.|xy|$ where $|xy|$ is the distance between x and y, and f is the medium (either a face or an edge). If e is the undirected edge shared by faces f and f', noted $e = f \cap f'$, then $\alpha_e = min(\alpha_f, \alpha_{f'})$ or $\alpha_e < min(\alpha_f, \alpha_{f'})$. In the first case, traveling along the edge is as traveling inside the cheaper region. In the latter case, traveling on the edge can actually be cheaper than traveling inside

either faces. This can be associated to traveling on a road between grassland and mountains. An error tolerance is also set by the user [MP91]. An error tolerance ϵ is also set by the user. As ϵ takes smaller values, the path will be improved however the computation time will be increased. The error can therefore get arbitrarily small as it is set by the user, and not by the algorithm itself (as the raster method).

Question: Build a structure (a *shortest path map*) that allows one to compute a $\epsilon-optimal$ path from s to any query point t. The path will consist of a set of geodesic paths which are paths that are locally optimal. Assume $\alpha_e = min(\alpha_f, \alpha_{f'})$ for edge $e = f \cap f'$, then a geodesic path that passes through the interior of e must bend at the edge according to Snell's Law of Refraction of light, which in this case would give : $\alpha_f sin\theta = \alpha_{f'} sin\theta'$, where θ and θ' are the angles of incidence and refraction (respectively). Note that solving for θ' gives $\theta' = arcsin[sin\theta.\frac{\alpha f}{\alpha f'}]$.
According to Fermat's least principle light seeks the path of minimum time. The index of refraction n_f for a region is proportional to the weight of the face α_f and inversely proportional to the speed with which light in that region. (as seen above $n_f = \frac{1}{v_f} = \alpha_f$. [MP91] The proof is a single variable calculus problem, first attacked by Pierre de Fermat. The function :
$f(x) = \frac{1}{v}\sqrt{x^2 + y_0^2} + \frac{1}{v'}\sqrt{(x_1 - x)^2 + y_1^2}$
gives the length of the path. We therefore wish to minimize this function, setting the derivative to 0. These same calculations are detailed in section 1.2.

2.2 Critical angles and vertices

The optics analogy has its limits, when a path hits a region in some specific ways. The results cannot be predicted by Snell's Law in these cases.

if $e = f \cap f'$, then the critical angle is $\Theta_c(e) = \Theta_c(f, f')$. if $\frac{\alpha_f}{\alpha_{f'}}.sin\Theta > 1$, then taking the inverse sinus to get the angle of refraction does not make sense, as sine is function that only maps [-1,1] on y-axis. The angle at which $\frac{\alpha_f}{\alpha_{f'}}.sin\Theta = 1$ is the critical angle at edge e. Note that a critical angle can exist only if $\frac{\alpha_f}{\alpha_{f'}} > 1$, which is when $\alpha_f > \alpha_{f'}$. In such case, if the angle of incidence at e is the critical angle a ray of light will travel along edge e rather that going inside f'. Also if the angle of incidence at e is greater than the critical angle, the light ray will be totally reflected. Note that when looking for the shortest path, any such internal reflection is not possible, the optics analogy breaks in this case.

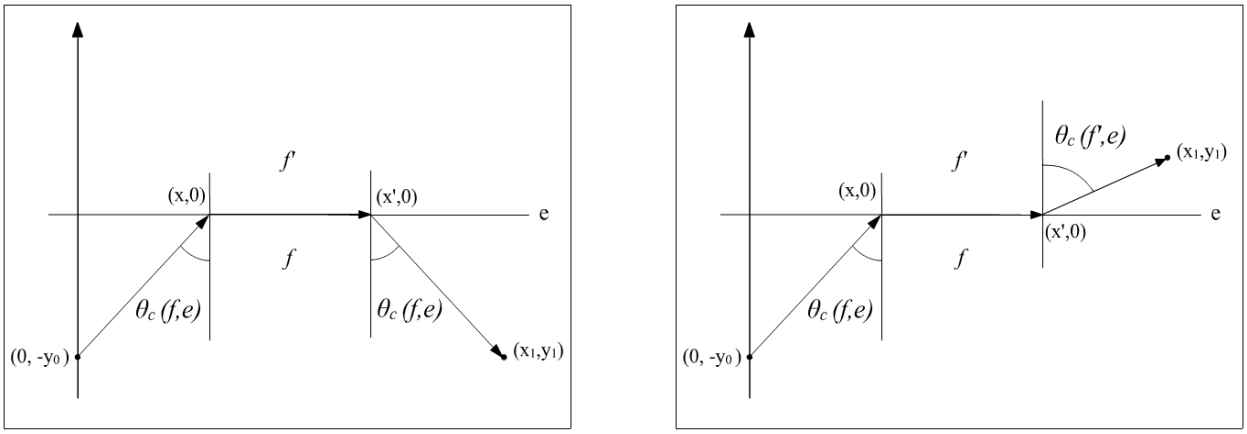

Figure 4: Path is critically used ($left$) and Path is critically reflected ($right$) [MP91]

2.3 Local Optimality Criterion

Generally, when the path hits an edge, it follows the following:

Local optimality criterion:

if $\Theta < \Theta_c$, *the path will refract according to Snell's law, namely* $\Theta' = arcsin(sin\Theta.\frac{\alpha_f}{\alpha_{f'}})$

if $\Theta = \Theta_c$, *then path p will travel along edge e for some positive amount, then:*

- *if* $[e = min(\alpha f, \alpha f')]$

the critical angle is $\Theta_c(e) = \Theta_c(f, f') = \arcsin\frac{\alpha_{f\,\prime}}{\alpha_f}$, *The path p is said to be critically reflected : it exits back into face f at an incoming critical angle* Θ_c *(see Fig 4).*

- *if* $[e < min(f, f\,')]$ *the critical angle* $\Theta_c(e) = \Theta_c(f, e) = \arcsin(\frac{\alpha_e}{\alpha_f})$. *The path p has critically used edge e: p exits into face f' at an outgoing critical angle* $\Theta_c(f', e) = \arcsin(\frac{\alpha_e}{\alpha_{f'}})$ *(see Fig 4).*

In both cases the segment $xx' = p \cap e$ is said to be a *critical segment* or *shared segment* of p along e. also x is called the critical point of entry to e and x' is the critical point of exit from e. The problem is that although the angle of exit can be defined, $|xx'|$ cannot. The path can use e for an arbitrary distance and then leave at any point x' $\in e$. Moreover, nothing is said in this local optimality criterion about p hitting an edge on an angle $\theta > \theta_c$ or about p hitting e in an end point of e.

A new path search must be started when the path hits a vertex or an edge at a critical angle, [MP91] proves that there can only a limited number of such events. Between any two consecutive vertices on p, there is at most one critical point of entry to an edge e, and at most one critical point of exit from an edge e'. The proof is addressed in [MP91] and is quite extensive so it is not addressed here. This property will be used to establish the complexity

of the algorithm, as it provides a relation between vertices and critical points.

The proof given by [MP91] uses the fact that the weighted length of the path is given by a function that goes from point s to t, and is strictly convex. Then the function possesses a unique global minimum, and any local minimum must be global. This remark is used to show how ray tracing can be done to find a shortest path from s to t.

2.4 Algorithm

An optimal path as a piecewise linear path that goes through alternating sequence of vertices, edge sequences and shared segments such that the path obeys the *Local Optimality Criterion* when hitting an edge. A geodesic path can be described as a list of vertices, a list of crossing points (where the path follows Snell's law) and critical points, with the constraint that in between consecutive vertices, there can only be one critical point of entry to an edge e and its respective critical point of exit.

Only certain points of this wavefront are recorded, namely the events that occur when the path hits an edge at a vertex or at a critical angle. These events are kept in a priority queue sorted on their distance to the source. These events can be seen as the output of the algorithm as they efficiently trace back the shortest path from any point in the triangulation to the source [BEMMELEN93].

Events are points associated to some candidate interval of optimality labeled with the best known distance back to the source. Intervals of optimality for a root r with respect to (e, f) describe the subset of paths for which the shortest path from r to $e \in f$ have the same sequence of edges. Its extent $[a, b]$ is a sub-segment of e where a and b are the end-points of the interval on e. In their algorithm each face-edge pair will be subdivided into different intervals (as many as $O(n)$). At the end of the algorithm, Each edge-face pair has associated with it a sorted list of its intervals of optimality such that none of them overlap (the union of their extents is the list of end points). Each interval has a frontier point c this is the closest point to r on the subsegment $[a, b]$.

$Main()$
A wavefront is originated at the source point, and will initially be propagated circularly through the triar
Select the next event from the priority queue, propagate the wavefront through the corresponding interva
- c is not a vertex ,
$I_1 = Project(I)$ $doInsert - Interval(I_1, c)$ - c is a vertex, for every face f containing c,
for every edge in e in f
create candidate Interval I $doInsertInterval(I, c)$

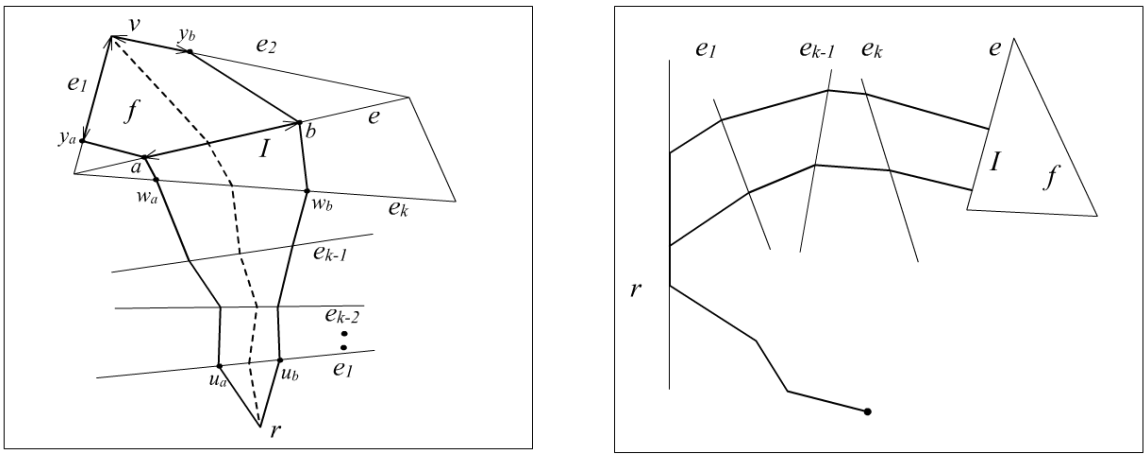

Figure 5: Interval of optimality with root r, to face pair (f, e) $(left)$, Interval of optimality with a critical root r $(right)$ [MP91]

$Project(I)$

Propagate I onto edge $e1$ (and $e2$).

leta_1,b_1 be the subsegment of e_1 that lies within I.

let $[a, y_a]$ be the left most refracted subsegment of I, and $[b, y_b]$ be the right most refracted sub

if $y_a \notin e_1$

then $a_1 = b_1 = \emptyset$.

if both$\{y_a, y_b\} \in e_1$

then the propagation of I ontoe1 is also trivial, $a_1 = y_a$ and $b_1 = y_b$.

Otherwise (namely $y_a \in e_1$ and $y_b \in e_2$), the interval must be split. do $FindPoints(I, v)$, whic

The procedure FindPoint is very computationally intensive as it involves a binary search to backtrack the shortest path from v to r ($seedottedlineinFig5$) and find the first bend point.

Note that every time a vertex is hit, or a critical angle is found, new intervals must be created and propagated. The propagation of an interval is the process of finding the cone or the belt which defines the interval of optimality (*see* $Fig5$) [MP91] - section 5.

[MP91] - section 5.

2.5 Complexity

The overall time for this algorithm is claimed to be $O(n^4)$ on the number of events because there can be at most $O(n^3)$ intervals of optimality per edge-face pair [MP91] -section7. They prove this by computing the upper bound of times the edge is encountered on shortest paths

that have no vertices or critical points; and then computing the times each first encounter can hit a critical point or vertex therefore producing a new interval of optimality. An edge e can be crossed $O(n)$ times by a path p that has no critical roots if it is shaped in a spiral-like manner. Their proof that the number of critical roots is quadratic is that such a path can intersect any edge in a critical point of entry, thus creating another interval of optimality as many times as it intersects that edge. Since there are $O(n)$ vertices, each can be a root to each one of them $O(n)$ critical points of entry to edge e. [MP91] - section 7. Therefore:

$O(n)edges.\ O(n)\frac{intersections}{edge}.\ O(n)\frac{vertices}{intersection}.\ O(n)\frac{criticalpointsofentry}{vertex}$

3 Geometric Issues

3.1 Modern version of Fermat's principle

The original version of Fermat's principle of least time states: "The actual path between two points taken by a beam of light is the one which is traversed in the least time." However this is not always correct. The modern version of the principle states that "the path a light beam follows is a stationary path". This means that slight variations in path leaves the travel time unchanged. If the length of the path is given by $f(x)$ then a beam of light will follow a path of length $f(a)$ *s.t.* $f'(a) = 0$. From calculus, it is known that the point $(a, f(a))$ can be either a local minimum, a local maximum or a sadle point. For example, when a function is an odd power, there is not local minimum or maximum, only a saddle point. A real life example of a point of inflection occurs when light is reflected off of an elliptical mirrored surface. The path that light takes from the center of the ellipse s back to s is either minimal or maximal.

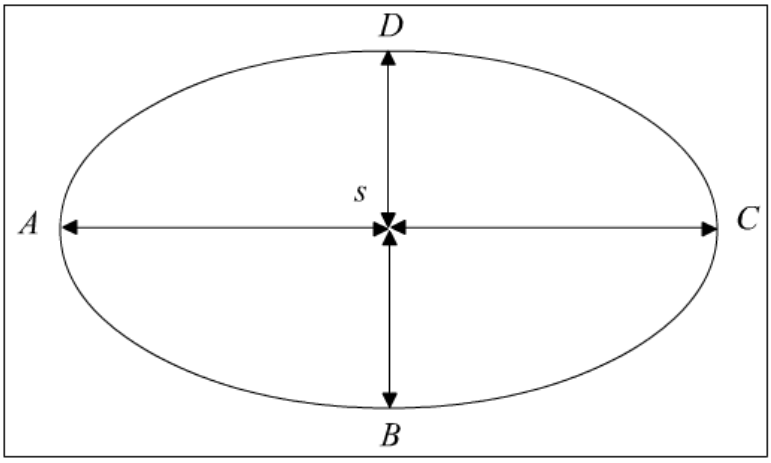

Figure 6: Light using paths of minimal time (through B and D) *AND* maximal time (through A and C to get from s back to s [NAHIN04]

Most physicist omit the details of fermat's principle and assume that other physicists understand that both maxima and minima are possible. Some call it, "principle of least or greatest time" or also "principle of stationary action" [PERLICK56]. Some books that study Fermat's principle in depth explain in in terms of this modern version. [PERLICK56] defines Fermat's principle as the following : "Fix two points in space, consider all possibilities to go from one to another. Among all these 'trial curves', the actual light rays are then the local extrema and saddle points of a certain functional which is called the optical path length." Other books such as [BERGETHON85] define Fermat's principle less carefully : "If the speed

varies as light travels on a path from one medium to another, the light will bend to the path that leads to the shortest time. [...] The overall superposition of light waves leads to no net wave arriving except from the shortest distance because all of the neighbors annihilate each other." Computer scientists solve the WRP using fermat's principle and Snell's law, Snell's law only holds locally, and does not seem like the best approach to finding shortest paths.

3.2 Snell Path, Critical Path and Shortest Path

Note that snell's law of refraction determines a change of direction in the path based on a ratio velocities of light in the different mediums and on an angle of incidence. The angle of refraction is by no means a function of the goal t but of the conditions found. Consider two points $\{s, t\} \in f$ that are to the left and to the right of a face f', and $\alpha_f < \alpha_{f'}$, then the direction of light traveling from s to t is treated exactly the same in both of these configurations :

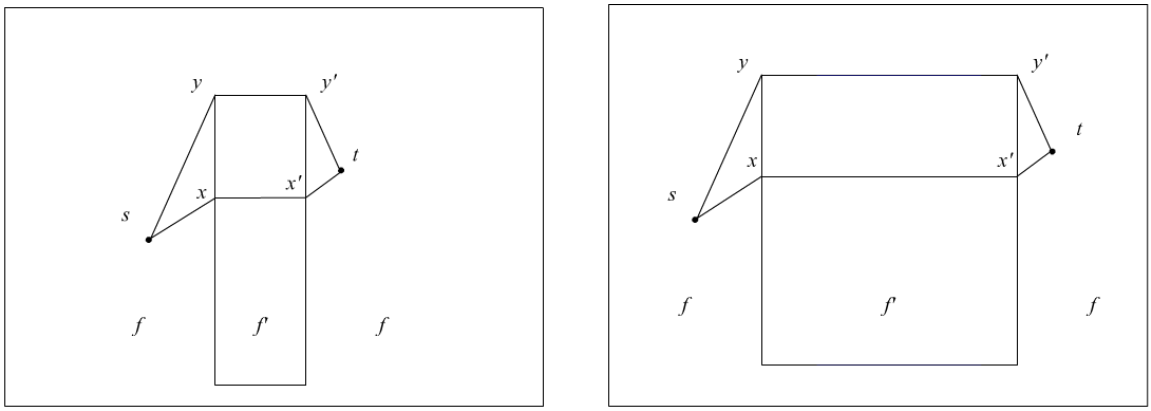

Figure 7: two different configurations, same Snell path

Maintaining equal weights, and changing face dimensions can produce completely different shortest paths where Snell's law would take the same one.

Let's compute the candidate paths for shortest paths for both of these configurations with $\alpha_f = 1$, and $\alpha_{f'} = 3$. we also have $|sy| = |y't| = 6 * \alpha_f = 1$ and $|sx| = |x't| = 2\alpha_f = 2$. then :
$p(a) = |sy| + |yy'| + |y't| = 12 + |yy'| = 12 + 1 * \alpha_f = 13$
and $p(b) = |sx| + |xx'| + |x't| = 4 + |xx'| = 4 + 1 * \alpha_{f'} = 7$

to the right, the setting is similar : $p(a) = 12 + |yy'| = 12 + 8 * \alpha_f = 20$
and $p(b) = 4 + |xx'| = 4 + 8 * \alpha_{f'} = 24$

In both cases Snell's law would act in the same way, maybe it would choose the path *syy't* or *sxx't*, however in any case, it will always choose the same path. let us refer to the path that goes from s to t and minimizes distances as a $SnellPath(st)$ if non of its bending violates Snell's law, and $Critical(st)$ if it does violate Snell's law at least once.

it is not an easy task to generalize when to use a $SnellPath$ or a $CriticalPath$. Let's assume that there are only two faces similar to $Fig7$, every configurations of weights will give a $SnellPath$ and a $CriticalPath$ however it is not at all clear which one we could use. The number of possible configurations is infinite, even with only two faces.

A path violating Snell's law once or twice is not too much trouble, as it the shortest paths can be broken up into $O(1)$ number of shortest paths. The problem arises when the number of faces that follow this configuration is $O(n)$.

3.3 Example provided by P.h.D student , University of Denver

In this example, the triangular subdivision has $O(n)$ triangles and the the Snell's law is violated $O(n)$ times. let α_1 be the weight of $Area1$ (two lower faces), let α_3 be the weight of $Area3$ top triangles, and let α_2 be the weight of $Area2$ the rest of the triangles. let p be the path in bold, the shortest path from s to t in this configuration. Also, let S_v be the number of times that the path does not follow the law of refraction on the border of two faces.

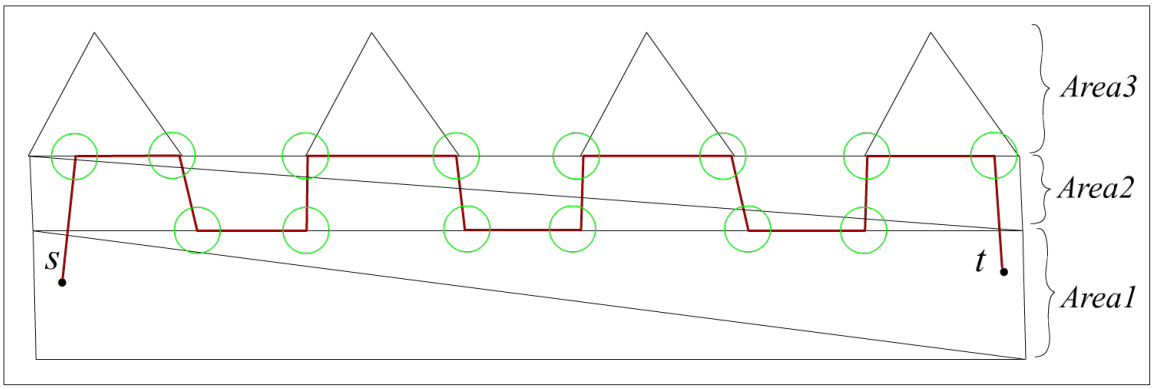

Figure 8: Example from Sada Narayanappa : Snell's Law violated $O(n)$ times

In order for the shortest path from s to t to follow (or closely follow) the path in bold we need the following conditions to be met :

- $\frac{\alpha_3}{\alpha_2} \approx 0 \Rightarrow$ the path will strive to pass just inside the top triangles
- $\alpha_3 < \alpha_1 < \alpha_2$
- The triangles that compose $Area2$ need to be very thin, in other words the height of the

rectangle composing the faces of $Area2$ needs be very small.

It is easy to show that there are $O(n)$ top triangles, where n is the number of faces or triangles in this subdivision. Let n_k be the number of faces with weight α_k in the previous example. then, $n = n_1 + n_2 + n_3 = 2 + 2 + n3 \Rightarrow n_3 = n - 4 = O(n)$. It is also easy to demonstrate that each triangle has a critical point of entry and a critical point of exit on its lower edge, that edge is said to be critically reflected by the path. Note that for every time the edge of a top triangle is critically reflected (excluding the triangle that is closest to t), there is one segment in the edge that borders $Area1$ and $Area2$ that is also critically reflected. In this case there is also one point of entry and one point of exit. The path has these critical points as it tries to travel in $Area1$ and $Area3$ when possible. Each time there is a critical point (of entry or exit) Snell's law is violated therefore, we can establish a relation between the number of triangles and S_v: $S_v = 4n_3 - 2 = O(n)$. the exact number of times snell's law is violated is $4 * (n - 4) - 2$.

3.4 How does light know the shortest path, at the beginning of the path ?

Fermat's principle is said to be meta-physical as its scope goes way beyond mathematics and physics. The obvious question is how does light, or nature for that matter, knows at the start of the path, what sub paths will result in minimum effort (length or time). Quantum electrodynamics (which was not developed until 1900) explains this phenomenon: light doesn't know where it is going, and it doesn't always take the quickest path. According to QED light does not have to - it simply goes over every possible path, and the observer (at a particular location) simply detects the mathematical result of all wave functions added up (as a sum of all $lineintegrals$). The phenomena of light can be approximated by rules such as "light travels in straight lines" or "light travels along the shortest path". This is because when traveling distances that are large enough, the probability of light to take the shortest path is bigger than its probability of traveling along longest paths [**?**].

In Quantum electrodynamics, The probability of a fundamental event u is the absolute value of the square of its amplitude. These amplitudes are directions which are represented by vectors, or arrows. Each amplitude vector's direction corresponds to the amount of time it takes light to get from the source to the end point following a particular path. $P(e)$ can be represented as the path integral of a vector field function. All amplitude vectors are added up to get the amplitude of e, $P(e)$ ($see\ Fig11$). The major contribution to $P(e)$ occurs at

the straight line path, which is the path of least time. This is because the path integral is dominated by solutions which are stationary points. Close to stationary points, the direction of the amplitudes of similar paths will constructively interfere (or add up) with one another. Contrarily, paths that are far from being stationary cancel each other. These vectors will not vary much for stationary points, but will vary very much for similar paths that are far from stationary (*see* $Fig10$).

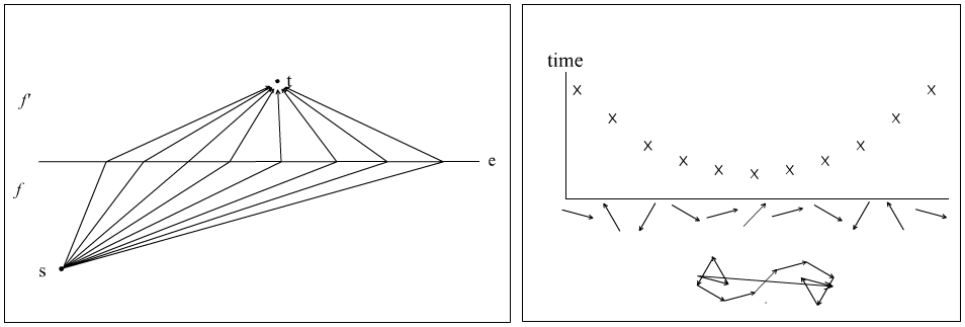

Figure 9: Computing path integrals : paths that light can take from s to t from left to right ($left$) integrating the vector amplitudes into one single arrow ($right$) [FEYNMAN85]

Note that the same reasoning is used by QED to show that light does not only travel in straight lines. The amplitude for light travel in straight lines is much bigger than to travel in curves. However the contribution made by nearly straight paths are also important. For example, note that a mirror that is too small does not work properly, light scatters in many directions because the core of neighboring nearly straight paths does not make it to and back from the mirror [**?**].

The size of these arrows or amplitude vectors is determined by P(A to B), which is a function of the difference of space and the difference in time. These vectors are added for each sub-event that is analyzed. $P(AtoB) = (X2 - X1)^2 - (T2 - T1)^2$ where $(T2 - T1)$ is the time that light takes to travel $(X2 - X1)$ units.

Note that for simplicity this equation considers space as 1-dimensional. If we consider a 3-dimensional space, the equation would still hold : $P(AtoB) = (X2 - X1)^2 + (Y2 - Y1)^2 + (Z2 - Z1)^2 - (T2 - T1)^2$ The major contribution to $P(AtoB)$ occurs at the conventional speed of light, namely when $(X2 - X1) = (T2 - T1)$. There is also a probability for light to go faster or slower that the speed of light. These amplitudes are small compared to the amplitude of light to go at conventional speed, however when distances are very short, other possibilities become of great importance and must be considered. The difference in time is not significant and therefore any path could be stationary. This phenomenon is called interference, and it is the existence of it that makes it necessary to add line integrals (or

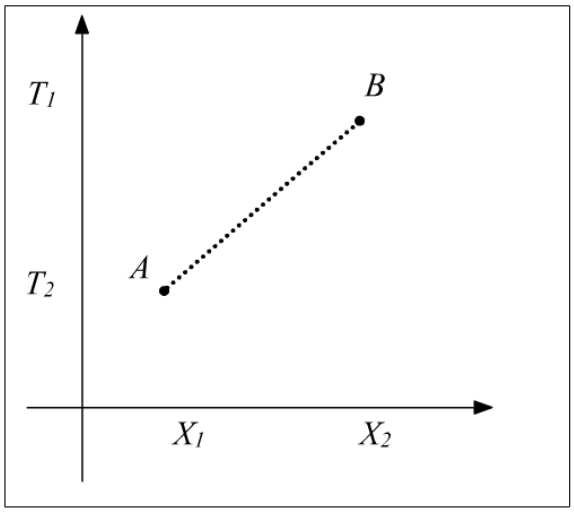

Figure 10: Light moving in space and time [FEYNMAN85]

path integrals) in order to predict where light will be [FEYNMAN85].

To summarize all these ideas, when time is least, it is also where time for nearby paths is nearly the same (which is when light travels at conventional speed of light, and also when light travels in straight lines). That is where the arrows or amplitude vectors point in nearly the same direction and add up to a substantial length. This is why in approximation we can speak of light only going where time is least.

Quantum electrodynamics shows that Fermat's principle is right, but only in approximation. Light does not only travel where time is least, and does not only travel in straight lines, and does not only travel at conventional speed of light. However as a whole, light seems to travel in straight lines and along the path of least time in the human eye, as it can only capture an approximation of light's behavior.

4 Simulation of light's path

4.1 Platform Choice

As seen in section 1, The algorithm used for solving WRP is actually taken from optics and assumes that a local shortest path is the one that would travel a ray of light. For that reason we are simulating the path taken by the light in a polygonal subdivision input by the user. As we will see in later, the program as we ambitioned needs a nice graphical user interface for it to be more visual and user-intuitive. One could set the coordinates of the initial point by taking integer x and y coordinates from the user; and also set the direction to an angle between 0 and 360; however this would not be very nice for the user, as it would be difficult to visualize where the point is going to land. We also needed our code to be portable, either via html or via a .exe application. The development platform was selected out of two choices: C++ using OpenGL API or Java (using the Java Graphics API). The latter was chosen for various reasons. Java is the most universal cross platform development language because a program's source code is compiled into an intermediate "byte-code" language. The "byte-code" is then executed by a Java Virtual Machine (Java interpreter) that was written for the particular hardware platform used by the user. As a result, generating an executable application for a java program is trivial.

I have previously used both platforms to develop different graphical applications. I used c++/OpenGL to create a solar system animation and also a clone of the well-known game "Tetris". I used Java and Borland's IDE for simulating a flight planning application. The latter choice provided with easier functions to program a friendly graphical user interface. Furthermore, java's pure object-oriented paradigm serves very well the purpose of the simulation software; this will be further explained in section 4.3. The only common downside about using java, is that since it is an interpretative language (compiles to intermediate language) as opposed to an imperative language (compiles to machine language), java tends to run slower than other languages such as C, C++. However the application we developed was not particularly computationally intensive, as the rendering will only include a set of polygons and a path which is basically a set of vectors traveling along the set of polygons.

4.2 Graphical User interface

WRP problem uses a triangulation for attacking the problem, however our aim is to provide a simulation of the path that light takes in a given polygonal subdivision, therefore we aim for more flexibility to the user, letting him create any sort of polygon that share any number

of vertices and edges. The creation of a polygon requires the user to input a given positive weight, this weight ranges from 0 - 255, and is visually represented by shades of gray : as weight gets larger, the color of the polygon gets darker. the rest of the space has an arbitrary weight which is set to 100.

It is important to be able to see the path of light within different subdivisions. For this purpose we provide a graphical user interface that permits necessary operations to set and manipulate an initial beam of light. After the user is done compiling the polygonal subdivision, the user should set the coordinates and start direction of the initial beam. left-click on any face of the subdivision origin vertex of the initial beam. Move the mouse to any direction the same face and left-click again to set the destination vertex of the initial beam. The beam is extra-sensitive to vertices and edges.

When a polygonal subdivision and an initial light beam are set, the user can either animate the path step by step or animate it entirely until it goes out of the triangulation. In the menu bar, clicking the step button, or hitting the accelerator key "Ctrl + D" will show the path that light would take until it hits the boundary of a region, where the refraction will most likely change the direction of the light path (unless of course, the weights of the triangles that share that edge are the same). Also, at the end of each path iteration, the state of the path will be updated in the bottom right panel. The statistics include variables such as the weights of the triangles that share the edges, the incident angle and the refraction angle. Note that the user can modify the triangulation by modifying the weights of any face at any given time. The user can also reset the initial point at any given time. This feature provides the user with increased flexibility when interacting with the subdivision and the light path.

4.3 The program's Structure

As said earlier, java's object oriented paradigm was crucial for the development of this application. The polygonal subdivision is stored as a Doubly Connected Edge list ($DCEL$) which keeps a record of every vertex (including a pointer to one edge that it belongs to), every edge (including pointers to its incident face, origin vertex, destination vertex, twin edge, next edge and previous edge), as well as every face, containing a pointer to any of its incident edges (the files containing these classes are vertex.java, edge.java, face.java, DConEdgeList.java). The path is stored in a class $SnellPath$ and the user interface is handled extending libraries from java's own gui libraries. (Snellframe.java and snellpanel.java).

The geometric classes serve as a project specific Api. the Java graphic libraries (java.awt

and java.awt.geom) provide an extensive library of methods and properties geometric shapes such as points, lines and polygons. Object oriented paradigm is used to make the class Edge and Polygon inherit methods from these java libraries. The class Edge inherit all methods from java.awt.geom.Line2D which is the double precision version of a line in java.awt.geom. It inherirts all methods and properties from it and can access methods such as *ptLineDist* (which gives the shortest distance a given point to this line. Also the class Face extends java.awt.Polygon). It can access methods such as *contains* (which identify if a given point is inside the polygon). An Edge stores its origin and destination, and also its *deltax* and *deltay* (the difference of x-coordinate and y-coordinate respectively). These are used to determine the incident angle and refraction angle, among others. Vertex implements a point with integer coordinates that is used for input points (polygon vertices, and initial path's input by user). The java.awt.geom class Point2D.Double is used for double precision when required. The project specific class is called SnellPath. This class holds all necessary data to compute the exit angle, this includes (see Fig. 12):

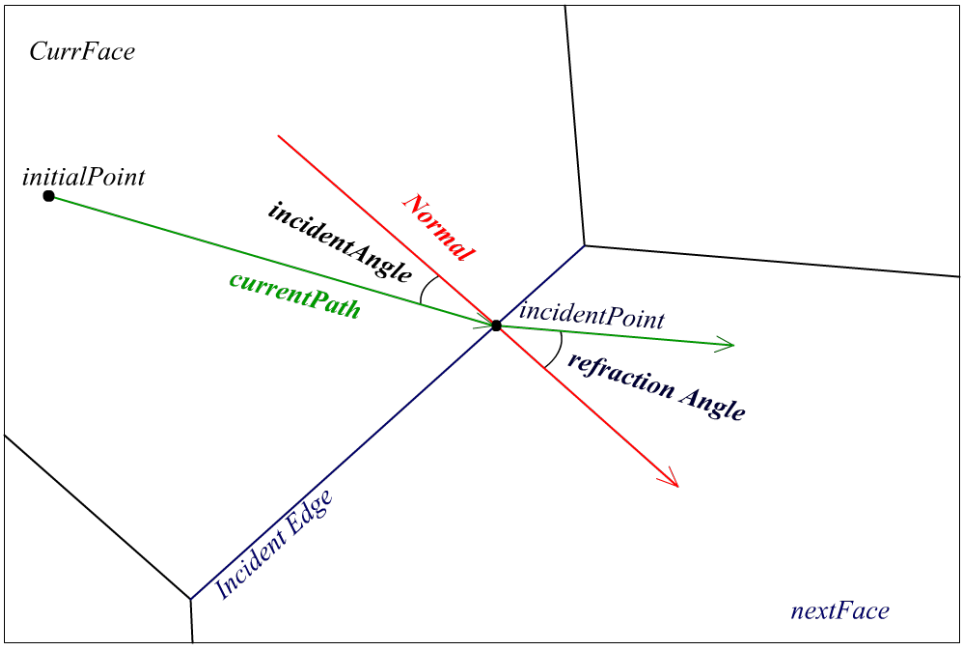

Figure 11: Snell path's class members

- a pointer to the polygonal subdivision - *dcel*
- a point of intersection - *incident Point*
- some edges :
    - an edge for the current Path being refracted - *currentPath*

- an array of edges that corresponds to the entire path taken so far - *pastPaths*

- the edge that the current path intersects - *incidentEdge*

- an edge that stores the normal to the incident edge - *normal*

- some angles :

- the sine of the incident angle obtained directly to avoid floating point errors - *sinAngle*

- an incident Angle and refraction Angle - *incidentAngle,refractionAngle*

4.4 Algorithm

- input:

- a polygonal subdivision S with vertices and weights set by the user

- an outside space with a predefined outside weight

- an initial beam of light defined by the user

- output:

- The path of light: represented by a series of paths refracted from different polygons

AnimatePath()

While(path is not outside canvas area) {

animateOnce().

}

AnimateOnce ()

setIncidentEdge()

setIncidentAngle()

setRefractionAngle()

updateCurrentPath()

set the *currentPath* = normal to *incidentEdge*

rotate *currentPath* counterclockwise by $refractionAngle$

normalize *currentPath*.

setIncidentEdge()

Compute intersection Points of each of the three lines with the *currentPath*.

Save the edge that has shortest distance to the Current path and that is its same direction.

Save respective point as $incidentPoint$.
set $nextFace$ = edge(incidentEdge.inTwin).inFace

$setIncidentAngle()$
set the Normal to the $incidentEdge$;
sin = crossProduct(normal,$currentPath$) / length($currentPath$)*length(normal)
$refractionAngle$ = arcsin (alpha($currentFace$) / alpha($nextFace$).sin);

Conclusion

[MP91] uses Snell's law and fermat's principle, When fermat doesn't work, [MP91] tricks the solution.

Light seems to follow fermat's principle, to the eye, although it was shown that light does not always travel in straight lines.

The principle of least action is puzzling, since instead of getting a result from initial conditions, one is given a start condition and an end condition and then find the path in between, as if nature knew where it was going.

The path integral is a good way to explain that nature does not know, it calculates the probability amplitude for a given process and the stationary points of the action mark the neighborhoods of the space for which the interference will add up constructively, yielding large probabilities.

References

[BEMMELEN93] Joost Van Bemmelen, Wilko Quak, Marcel Van Hekken, and Peter Van Oosterom (1993) :

'Vector vs. Raster-based Algorithms for Cross Country Movement,' Auto Carto 11 Proceedings. Minneapolis: ACSM-ASPRS, 1993.

[BERGETHON85] Peter R. Bergethon (1998) :

'The Physical Basis of Biochemistry,' Springer-Verlag, p.528.

[FEYNMAN85] Richard Feynman (1985) :

'QED The Strange Theory of Light and Matter,' Princeton University Press, p. 35-55,80-90.

[MAHONEY73] Michael Sean Mahoney (1973) :

'The Mathematical Career of Pierre De Fermat,' Princeton University Press, p.170-192.

[MP91] Mitchell and Papadimitriou (1991) :

'The Weighted Region Problem : Finding Shortest Paths Through a Weighted Planar Subdivision,' Journal of the Association of Computing Machinery.

[MMP87] Mitchell, Mount, and Papadimitriou (1987) :

'The Discrete Geodesic Problem,' Siam J Computing 16,4.

[NAHIN04] Paul J. Nahin (2004) :

'When Least Is Best,' Princeton University Press, p.101-135.

[PERLICK56] Volker Perlick (1956):

'Ray Optics, Fermat's Principle and Applications to General Relativity,' Springer-Verlag, p. 149-164;