Voronoi diagram in the plane

Scene: *Sites*

Black circles in the screen window represent points of the plane that will be called *sites*. The collection of sites is the input information for Voronoi diagram construction. Here and in most othere scenes a new set of sites can be created using the button $\boxed{\text{New input}}$; the size of a set can be set in the field $\boxed{N=}$.

If available in a particular scene, the buttons $\boxed{\text{Zoom-in}}$ and $\boxed{\text{Zoom-out}}$ may be used to zoom a configuration.

The set of sites can also be modified by a mouse if a suitable mouse function is selected in the control bar:

- if the mouse function choice in the control bar is set to $\boxed{\text{Move site}}$, any site can be dragged by a mouse;
- if $\boxed{\text{Insert site}}$ is selected, a mouse click to any point other than a site creates a new site; and
- if $\boxed{\text{Delete site}}$ is selected than a site is deleted when clicked by a mouse.

These features are even more useful when the Voronoi diagram is displayed. In certain scenes the mouse function is pre-determined and can not be changed; the choice control element is not displayed.

Scene: *Voronoi diagram*

The scene shows a Voronoi diagram of the set of sites. Each site is surrounded by a convex region bounded by black line segments and/or halflines. Each region corresponds to one of the sites, and all the points in one region are closer to the corresponding site than to any other site.

In exceptional cases a boundary can also be represented by a line (in this case the region is not a polygon, but a half-space) or by two parallel lines (and the region is an infinite stripe), but such cases are very unlikely if the number of sites is grater than 2.

Let $\rho(x, y)$ denote the Euclidean distance of two points x and y of the plane.

Points of the plane can be divided into several categories:

- the interior points of the region around a site s are points x of the plane such that $\rho(x, s) < \rho(x, s')$ for any site s' different from s;

- the points on the boundary of regions of two sites s_1 and s_2 are points x of the plane such that $\rho(x, s_1) = \rho(x, s_2) < \rho(x, s')$ for any site s' different from s_1, s_2. Such points belong to segments and halflines of the Voronoi diagram that are not their end-points.

- the points on the boundary of regions of three sites s_1, s_2, and s_3 are points x of the plane such that $\rho(x, s_1) = \rho(x, s_2) = \rho(x, s_3) < \rho(x, s')$ for any site s' different from s_1, s_2, and s_3. In such points three line segments or half-lines of the Voronoi diagram meet.

- In general, one can also find points that have four of more equidistant nearest sites. However, such points (that look like a Four Corners Point on the boundary of Arizona, Colorado, New Mexico and Utah in USA) are not very frequent, because their nearest sites would belong to the same circle and it is known that a random set of points of the plane is not likely to involve such quadruples. (This and other degenerated cases will be analyzed later)

Try to modify the site set by inserting, deleting or moving sites and watch how the diagram changes.

Scene: *Algorithm animation*

The applet explains an algorithm that constructs a Voronoi diagram using a method discovered by S. Fortune. Let us note that there are other algorithm to solve the problem. An incremental algorithm starts with a Voronoi diagram of a small subset of sites (e.g., three leftmost points), adds sites one by one and modifies the diagram at each step. A "divide et impera" algorithm splits the sites set to two approximately equal parts by a vertical line, creates Voronoi diagrams of boths parts and then combines them together to obtain a Voronoi diagram of the whole set. Fortune's algorithm was selected for this presentation, because it is based on a very elegant algorithmic idea.

The algorithm will be explain not only in the plane, but also in a 3D embedding. In order to avoid any possible misunderstanding, the "upper" and the "lower" part of the screen will be respectively referred to as "nothern" or "southern", similarly for directions. The words "up" and "down" will be used only in 3D case (and will basically denote the direction orthogonal to the screen plane).

Fortune's algorithm is so called *sweep line* algorithm. Start animation by clicking the button Go on. After a while a line appears in the screen window, which "sweeps" the plane from south to nord. The line is not drawn explicitly, but it is determined by the boundary between the dark and the light part of the plane. We assume that sites become known to the algorithm only when swept by the line. It is therefore possible to draw only a part of Voronoi diagram - the part that can be determined using only known sites and will not change when other sites are swept.

This is why the sweep line is followed by a red curve that is often called a *beachline*. As it will be explained later, Voronoi diagram in the region between the green sweep line and the red beachline can still be affected by yet unknown sites somewhere north of the sweep line. This is why nothing is drawn in this region. On the other hand, the region south of the beachline is "secure" and is completely dominated by sites that are already known. Thus, Voronoi diagram in its ultimate form appears in the southern part of the screen window.

The animation can be stopped by the button HALT, restarted in both direction by buttons MOVE NORTH and MOVE SOUTH (Move up/down in the present applet; will be corrected).

Observe that the beachline consists of many arcs (as we will see later, they are parts of paraboles) and the joins of arcs draw the diagram. The reason will be explained later.

Scene: *Cones*

This scene and the next one present an alternative view of building Voronoi diagram. Even though it is not necessary for explaining Fortune's algorithm, it brings an intuition that could help to understand the underlying algorithmic idea.

Visualization in this scene is based on an embedding of the planar collection of sites into the 3D space. Drawing and updating the image is computationally very intensive and therefore there could be delays when a slower computer is used.

As it was already mentioned, a planar collection of sites is embedded into the 3D space. Imagine that the plane containing sites is horizontal, but observed from above. Each site is an apex of a circular conical surface (cone), which is shown as if it is under light comming from the west, the slope of cones is 45 degrees.

What we see is therefore a hilly landscape, where all hills are regular shape volcanos extending to the same altitude. When looking from above, we do not see whole cones, because the cones partially hide each other. Only that part of each cone is visible that is not hidden by another cone. Let us note that no perspective is used to draw the landscape (that looks as if viewed from a point in the infinity).

The landscape can be rotated in the space using left, right, up, and down arrows of the keyboard and also by mouse movement when the left mouse button is pushed. However, if the Ctrl key is pressed simultaneously, the direction keys do not case rotations, but shifts if the image, and the same is true for a mouse is the right button is pressed instead of the left one. The BackSpace key returns the mountain into the original position.

Look at the visible intersections of cones that represent bottom lines of valeys of the landscape. If the landscape is in a general position, we see that the intersections are arcs (and if you have elementary backgroung in analytical geometry, you certainly know that they are sections of hyperboles). However, if the BackSpace is pressed and the landscape is displayed in the original position, viewed from above in a direction that is parallel to axis of the cones, the intersections arc now look as if they were line segments and/or halflines. Moreover, it is clear that the intersections represent the Voronoi diagram of the given collection of sites.

To explain the latter observation, look first at any point x on the surface of some cone. Let us s be the site defining the cone and denote by x_0 the verticalprojection of the point x into the horizontal plane containing sites (cone apexes). The triangle xx_0s is a right-angled isosceles triangle, because the angle xx_0s is a right angle and the size of the angle x_0sx is 45 degrees. Hence the "horizontal distance" of points x a s, i.e. the distance of s and the projection x_0 of x into the horizontal site plane, is the same as the difference of altitudes of the point x and the site s. The "horizontal distance of two points is also the distance of their images, how they apper on the screen if the landscape is observed vertically.

Now, let us consider any point x that belongs to an arc, which is an intersection of cones represented by sites s_1 and s_2 and it is not hidden by any other cone, when observed from above. Since the sites s_1 and s_2 are in the same altitude, the altitude difference between x and s_1 is the same as the altitude difference between x and s_2. It follows from the previous paragraph that the differences of the "horizontal distance" from x to s_1 and from x to s_2 are the same. In other words, the projection x_0 of the point x into the site plane has the same distance from s_1 and from s_2. Moreover, if s_3 is any other site and y is the point of the cone of the site s_3, which projects vertically also to the point x_0, it must be below x (or to coincied with x in the extreme case when x is on the surface of the cone of s_3 as well), because we assume that x is not hidden by any other cone. Hence the distance of x_0 and s_3, which is equal to the distance of y from the site plane, is greater or equal to the distance of x_0 from s_1 and s_2.

Scene: *Sweep plane*

Now, imagine that the landscape is "swept" by a plane that is inclined to the south under 45 degrees, and moves from the south to the nord. Let the plane moving (click either the button Go on or the button Go back and rotate the landscape to different positions to understand better how the plane moves. In fact, only a part of the sweep plane is shown ... ? Imagine the plane as made of bluish glass - the part of the landscape that is below the glass of the sweep plane is colored using different ? odstin of blue, while remaining (directly visibly) remains colored by neutral colors from white through gray to black.

The intersection of the sweep plane and the horizontal plane containing all sites is displayed as a ? green line. Moreover the intersection of the sweep plane with the landscape surface is drawn as a red curve.

Now return the landscape into its original position by pressing the BackSpace key. It is clear that the green intersection of the sweep plane and the site plane becomes the sweep line of the 2D scenes, while the red intersection of the sweep plane and the landscape surface becomes the beachline as it was shown in the 2D scenes.

Note that there might exist still unknown (=unswept) mountain cone that will later become visible in also in the region between the sweep line and the beachline when the landscape is observed vertically. This is why we are not yet able to predict in a reliable way the form of Voronoi diagram in this region.

On the other hand, the landscape surface that we see in the region south of the beachline is already fully visible, the sweep line is hidden under mountains, and hence Voronoi diagram is know an will not change when the sweep plane moves further nordwards, possibly revealing new sites.

The intersection of the sweep plane and the horizontal plane of sites gives the sweep line of the previous 2D scenes.

It is important that the first moment when the sweep plane touches a cone of certain site is the moment when it touches the site, which represents the top of the cone; in this moment it also touches a halfline that starts at the top site and goes southwards along the surface of the cone. This means that points that are on the perimeter of a Voronoi polygon of a certain site (that all belong to the cone in the 3D representation) appear in the moment when the site becomes known (= is swept by the sweep plane).

It is also clear, how the beachline in 2D image is constructed: as a projection of the intersection of the sweep line and the landscape, it is composed of parts of intersections of the sweep plane with cones. Since the sweep plane is always parallel in the southward surface halfline of the cone, it intersects any cone in a parabole, and a projection of a parabole is a parabole as well. This means that the beachline is composed of parabolic arcs.

In the moment when the sweep plane touces a cone for the first time, the intersection parabole is degenerated into a halfline that starts in the top site ang goes southward, but soon it becomes a parabole that is firs very narrow, but opens as the sweep plane continues to the nord.

Parabolic arcs that belong to neighboring cones meet, of course, in a point that belongs to the intersetcion of the cones. As already explained, cone intersections appear in the 2D projection as points of Voronoi segments and halflines that separate site polygons. This is why meeting points of beachline arcs "draw" the Voronoi diagram.

Scene: *Single parabole*

Let us first remind (as it will be proved in the school) that the set of points of the plane that have thes same distance from a given *focal* point and a given *directing?* line form a parabole.

In this scene we move back to 2D visualization and we will reconsider meaning of the beachline. Again we will deal with parabole, but this time not as an itersection of a plane and a cone, but as a set of points that have the same distance from a given point and a line.

As mentioned above, the algorithm tries to construct a part of Voronoi diagram that can be determined using knowledge of sites that have already been swept (or at least touched) by the sweep line. Such sites will be called *known sites*, remaining sites that have not yet been encountred by the sweep line are *unknown sites*.

The scene shows all sites; known and unknown sites are distinguished by color, and one know site is emphasised (by a right mouse button click, another known site can be selected). The sweep line is shown as well, but the beach line does not appear. On the other hand, there is another red curve - a parabole of points that have the same distance from the selected site and from the sweep line. The region below the parabole is light pink and its interior consists of points that are closer to the selected site than to the sweep line. The sweep line can be moved in the same way is in the previous scenes by a mouse (left button), buttons Go on and Go back and by up and down arrow keys of the keyboard, but with restriction that the selected site must remain known (i.e. south of the sweep line).

Observe how the parabole changes when the sweep line moves and gets wider if the line moves to nord and narrower if the site gets closer to the sweep line.

Using the right mouse button click to select another site and to observe another parabole.

It is also possible to select more sites if a new site is clicked when the Ctrl key is down, and to see how pink regions of paraboles overlap.

Let us note that this scene can be viewed in 3D as well. In this case the degenerated situation, when the sweep line contains the selected site, means that the sweep plane touches the selected cone, while later, when the plane moves to the nord, the cone is intersected in a non-trivial way.

Scene: *Paraboles together*

This scene shows a situation that could have been generated in the previous scene by selecting all sites: paraboles of all known sites are displayed in the same time together with their corresponding pink regions.

Note that the union of the pink regions contains the points of the plane such that their nearest known site is closer than the sweep line.

The upper envelop of the pink region consists of arc that are parts of paraboles of known sites, and represents a collection of points x such that the distance of x to the nearest site is exactly the same as the distance from x to the sweep line.

Red points are those that belong to the parabole of at leas one known site, but do not belong to a pink region of the parabole of any known site. These points have the distance to the nearest known site the same as the distance to the sweep line. It is clear that the collection of the red points is the beach line of the previous scenes.

Pink points are those that are closer to certain known site than to the sweep line, which means that are closer to certain known site than to any unknown (unswept) site. It is therefore possible to determine a region of the Voronoi diagram which involves the point; it is the Voronoi region of the nearest (known) site.

Remaining points south of the sweep line are those that are closer to the sweep line than to the nearest known site. Since an unknown (unswept) site can be hidden just behind the sweep line (i.e. north of it, but in a very short distance) it might happen that such a point would belong to the Voronoi region of yet unswept site, and therefore its Voronoi region can not be determined from knowledge of already swept sites.

Scene: *Beachline*

As it has already been mentioned, points on the beach line and south of it are those, for which their position in the final Voronoi diagram could be determined from the knowledge of sites that have been swept so far. Since the beach line moves northwards, a point is classified at the moment when it appears at the beach line. Later it would appear south of it and its membership in Voronoi regions does not change. Hence, let us analyze positions of points of the beach line. Two possibilities exist:

- A point x is *inside* of some parabolic arc. In this case x belongs to a parabole of certain known site s and for any other known sites it is north of their paraboles. This means that

- the distance from x to s is smaller than the distance from x to any other known site, and
- the distance from x to s is the same as the distance from x to the sweep line and therefore greater than the distance from x to any unknown site.

Together, these two statements imply that s is the unique nearest site of x, and therefore x belongs to the interior of the region of the site s.

- x is a joint point of two neighbouring arcs of the beach line that are respectively determined by paraboles of sites s_1 and s_2. In this case, if s is any other known site, then either x is north of the parabole of s or it also might be on that parabole. Therefore

  - the distance from x to both s_1 and s_2 is smaller or equal to the distance from x to any other known site, and
  - the distance from x to s is the same as the distance from x to the sweep line and therefore greater than the distance from x to any unknown site.

  This means that x belongs both to the boundary of the Voronoi region of s_1 and to the boundary of the Voronoi region of s_2 and it also could belong to the boundary of the Voronoi region of some other known site(s). Hence x is a part of certain boundary segment(s) of the Voronoi diagram.

This implies that points of boundary segments of the Voronoi diagram are exactly those points that happen to become joints of arcs of the beach line at certain moment of the computation. This is why joints of arcs "draw the Voronoi diagram".

Let us note that if x is a joint point of two beach line arcs, there could exist more than two nearest sites. This typically happens during so called circle event (see later), if a point belongs to a circle determined by three different sites such that there is no other site in the interior of the circle. Since any three non-colinear sites determine uniquely a circle, such a situation is relatively frequent. Exceptionally, a point can belong to more than three paraboles of sites. Such degenerated situation will be dealt with later.

Scene: *Events*

As the beach line moves north, its shape continuously change, and therefore a computer program can not keep track of all immediate forms of the beach line, because this would mean to collect an infinite amount of information. Fortunately, this is not necessary. What we only need to get is a list of the boundary line segments and halflines of the Voronoi diagram, and not too much information is needed to describe such elements:

- an infinite halfline is described by its origin and direction;
- a finite line segment is described by its two end points;

It is therefore sufficient just to know positions of beach line joints at moments when certain boundary line segment or halfline begins or ends to be drawn. Such moments are called *events*.

Note that the actual form of the beach line always change, but the number of arcs and their directing sites change only when an *event* occurs. The present scene makes it possible to animate movement of the sweep line to the north or south using buttons Forward and Backward in the same way as in the previous scenes, but the motion automatically stops when an event occurs. Start forward movement of the sweep line to see that there are two types of event (the place where an event occurs is marked by a green dot):

- **Site event:** The sweep line just hit another site (this explains the term used to describe the event). A new "arc" appeared in the beach line: the "arc", connecting the new known site with the point of the beach line that is exactly south of it, is in fact a line segment that is a part of the halfline starting in the site an directed southwards, and the halfline can be considered as a degenerated parabole of the new known site. (Note that this halfline is in fact the collection of points on the sweep line and south of it that have the same distance to the new site and to the sweep line).

  The "arc" cuts generally into two parts the arc of the beach line that is south of it. In this way one arc of the beach line is replaced by its two parts with the new "arc" in between. The number of arcs of the beach line increases by two.

  In exceptional case the "arc" meets the beach line in the interarc joint. In such a case no arc is cut ant the new degenerated arc is simply inserted between two arcs of the previous beach line.

  Note that immediately after a site event the degenerated parabole becomes a true parabole, though it is very narrow at the beginning, but getting wider as the sweep line moves.

- Circle event One of the arcs of the beach line have been "sqeezed to zero width" by two neighbouring arcs, which means that one arc of the beach line disappeared. The name of the event will be explained later.

I recommend to follow a form of the beach line both just after a site event and just before a circle event (e.g. by dragging the sweep line slowly by a mouse) to see how a degenerated parabole becomes a true parabole in the case of a site event and how an arc disappears in the case of a circle event.

Exceptionally another types of event occur, but they can be viewed as a simultaneous occurence of two or more events described above, e.g., simultaneous disappearing of two or more arcs that in certain cases disappear in the same point of the plane, which can simultaneously be a point when a new degenerated arc is inserted into the beach line. Such configurations will be dealt with at the end of the chapter.

Scene: *Site event*

The scene is similar as the previous one, but the sweep line stops only when a site event occurs.

Let us follow a particular site event; it is advantageous to choose a moment when the arc cut by the degenerated new arc is sufficiently wide and the cutting point is not too close from either end points of the arc.

Immediately before the event nothing suggests that an event would appear soon, because we assume that sites north of the sweep line are not known and information about their existence and position become available only when hit by the sweep line.

At the moment of the event a new site appears at the sweep line (at exceptional cases more sites could be hit at the same time). A new degenerated arc appears; assuming a non-degenerated event, the degenerated arc cuts an existing arc of the beach line and is inserted between them. The intersection of the two arcs is a point that has the same distance to the new site and the site corresponding to the cut arc. This is why the intersection point is a location where a new boundary segment of the Voronoi diagram starts to appear. At the moment of the event, the intersection point is the only point of the the segment.

Using a mouse, drag the sweep line slowly to the north. A degenerated parabole becomes a true parabole, very narrow at the beginning, but getting wider quickly. Intersections of the two branches of the parabole with the intersected arc of the beach line, i.e. joint points of the new arc with the west and the east parts of the original arc, move quickly in opposite directions and draw a new boundary segment of the Voronoi diagram. Thus, a segment is drawn "from the middle". Note that points of the segment must have the same distance from the new known site and the site associated with the intersected arc, and therefore they belong to the axis of the segment connected these two sites. This fact determines easily the direction of the new Voronoi segment.

For technical reasons, it is convenient to understand the new Voronoi segment that is drawn in two opposite directions from an interior point as two segments that are drawn from the intersection point of the event's degenerated arc and the intersected arc, and have opposite directions. Check the checkbox $\boxed{\text{Halflines}}$ to show drawing of any subsegment as starting in an end point of the segment, continuing in a particular direction and either ending at another point of the plane if the segment is finite or continuing forever in the case of an infinite halfline.

Scene: *Degenerated site event*

This scene shows an exceptional site event when a degenerated arc reaches the beach line in a joint of two arcs. In this case the new degenerated arc are inserted between two arcs; the segment drawn by the joint of two previously neighbouring arcs ends in the location of the event and two new arcs start in the same place - the arcs drawn by joints of the new arc with both neighbouring arcs. In this way two new line segments do not begin "in the middle of nowhere", but at their en points and they do not have opposite directions.

Scene: *Circle event*

Roughly speaking, a circle event is a moment when one arc of the beachline disappears and its left and right neighbors would meet to become connected. In this scene, automatic animation stops only at circle events.

Unlike in the case of site event, a circle event can be forecast. Let a circle event happen, and then move the sweep line slightly back. When the event occurs, an arc α associated with a site s_α disappear being squeezed to zero width by neighboring arcs β and γ associated with respective sites s_β and s_γ. Straight lines that are axis' of segments $s_\alpha - s_\beta$ and $s_\alpha - s_\gamma$, respectively, that ar guiding lines of joins of pairs of arcs (α, β) and (α, γ), respectively, intersect in a point that will soon be hit by the beach line and in that moment the end points of the arc α would meet and the arc itself disappears.

Click the arc that is going to disappear by a mouse: a circle (see below) appears (and for a moment avoid clicking the screen outside the control bar). Then, let the event to occur again in the screen by clicking the button Forward. In the moment of the event the arc α disappears and the neighboring arcs β and γ meet. It is not a coincidence that this occurs in the point which is the center x of the circle shown in the screen: at the moment of the event, the center x belongs both to the arc β and γ. Moreover, is still belongs to the disappeared arc α (which can also be considered as still present, but having zero length), just in a neighborhood of the point x the parabole determined by the site s_α, containing the arc α, is covered from the north by paraboles of s_α and s_β, respectively. Check the checkbox Paraboles to display full paraboles of all sites, not only their segments that are involved in the beach line. You will see all three paraboles corresponding to arcs α, β and γ pass through the center of the circle.

This means that the distance from x to the sweep line is equal to the distance from x to s_α (because x belongs to α), to the distance from x to s_β (because x belongs to β), and to the distance from x to s_γ (because x belongs to γ). As a consequence, all three sites s_α, s_β, and s_γ belong to the same circle with the center x - the circle shown in the screen window - and, which is also very important, the sweep line touches the circle at its northernmost point.

This gives a simple way to predict a future circle event: given three consecutive arcs β, α, and γ of the beach line, try to draw a circle that is uniquely determined by sites s_β, s_α, and s_γ corresponding to arcs. If a circle event of disappearing of α is concievable (the guiding lines of joins of arc pairs $\beta - \alpha$ and $\alpha - \gamma$ intersect north of the beach line), the location where the arc α would disappear is the center of the circle, and the event would occur in the moment when the sweep line would touch the circle at its northenmost point or, in other words, it the y-coordinate of the sweep plane is equal to the y-coordinate of the center of the circle plus the radius of the circle.

Therefore the program maintains a dynamic calendar of events. Site events are fixed and correpond to sites, and they are ordered according to y-coordinate of sites that give information about the moment when they are hit by the sweep line. New anticipated circle events could be created after any change of the set of arcs of the beach line and inserted into the calendar according to the y-coordinate of the northernost point of the corresponding circle, which determines a position of the sweep line in the moment when the event occurs. In certain cases, as a result of certain event, it might happen that scheduled circle event is removed from the calendar queue befor it is executed, since it relates to an arc that ceased to exist during the event. It is therefore necessary to associate any arc that is anticipated to disappear as a result of a circle event with a pointer to the event in order to find easily a calendar record to be deleted if an arc is destroyed.

As a variant, an obsolete circle event is not deleted from the calendar queue immediately, but simply marked as invalid. Execution of an invalid event consists in simple deleting the record from the calendar queue without performing any other action.

Scene: *Arc search*

If the site set is large and wide, it can easily happen that the beach line contains a large number of arcs. In such a case a site event execution can be slowed down, because determining an arc that would be cut by the (degenerated) parabole of the newly swept site involves examination of a large number of arcs.

The simplest way of finding such an arc is to scan all arcs of the beach line and to examine for each arc if it is south of the new known site by comparing the x-coordinates of the arc with the x-coordinate of the site. However, this method is quite slow and should not be used for large site sets.

It is more convenient to build a tree data structure above the set of arcs of the beach line as illustrated in this scene. Each leaf of the tree (a green dot) are associated with an arc of the beach line that is below the leaf. Each internal (i.e. non-leaf) node (a blue dot) has two children and corresponds to a joint of two consecutive arcs that is below it.

Each leaf involves information about the focal site of the corresponding arc, while an internal node of the tree involves information about focal sites of the two arc that meet in the corresponding arc joint (in such a way that it is possible to find which site is a focal site of the west (east, resp.) arc).

Search for an arc south of the new site begins in the root of the tree and continues as follows: when in an internal node v and given a position of the sweep line (y-coordinate), which is the directrix of site paraboles, equation of paraboles involving the arcs that meet in the beach line joint corresponding to v are determined; comparing the equations gives a quadratic equation that gives two intersections of the paraboles as solutions and an appropriate one of the two intersections is considered (more details will be given in school). If the x-coordinate of the new site is smaller (greater, resp.) then the x-coordinate of the intersection of the paraboles, the search continues in the left (right, resp.) child of the node. If the x-coordinates of these two points are equal, a degenerated site event occurs and the arc corresponding to the new site would be inserted between the arcs that meet in the joint related to the internal node v.
when the search arrives to a leaf v, the arc south of the new site is the arc correspondint to the leaf v.

The tree must be updated at any event:

- if a site event occurs, then certain arc α is cut during the event and replaced in the beach line by a sequence three arcs α_1, β and α_2, where α_1 and α_2 are parts of the original arc α, while β is a degenerated arc of the new site. The tree above the beachline can be updated as follows:

the tree leaf corresponding to the original arc α is replaced by a 5-node tree with 3 leafs v_{α_1}, v_β, and v_{α_2}, corresponding to respective arcs α_1, β, and α_2 and two internal nodes $v_{\alpha_1,\beta}$, v_{β,α_2} corresponding to respective joins $v_{\alpha_1-\beta}$, $\beta - \alpha_2$, and, e.g., $v_{\alpha_1,\beta}$ is the root of the substitution tree with a left child v_{α_1} and a right child v_{β,α_2}, while v_{β,α_2} has a left child v_β a right child v_{α_2}; moreover it is necessary to update arc information in internal nodes corresponding to the joints of the original arc α with its left and right neighbors (if present), because these two internal nodes do not any more refer to non-existing arc α, but to respective parts α_1 and α_2;

- if a degenerated site event occurs and the arc β of the new site is inserted between two arcs α and γ, then a 3-node subtree replaces the internal node $v_{\alpha,\beta}$ corresponding to the join of α and γ; the tree has a leaf v_β, corresponding to the new arc β and two internal nodes $v_{\alpha,\beta}$, $v_{\beta,\gamma}$, corresponding to joins of β with neighbors, and, e.g., $v_{\alpha,\beta}$ is the root of the substitution subtree, its left child is the former left child of $v_{\alpha,\beta}$, its right child is $v_{\beta,\gamma}$, and the left child of $v_{\beta,\gamma}$ is the leaf v_β, while its right child is the former right child of $v_{\alpha,\beta}$;

- if a circle event occurs, then certain arc β with neighbor arcs α and γ is removed from the beach line; in this case, the leaf v_β is a child of either the node v_α corresponding to α or the node v_γ corresponding to γ. In this case the leaf α and its parent are deleted using binary search tree methods (note that after deletion of the leaf α, its parent becomes a node with only one child), while the other of nodes v_α, v_γ is updated to represent the newly created join of α and γ.

It can happen that the tree built over the beach line becomes unbalances and searching in the tree is inefficient. This can be avoided by balancing the tree after any insertion and deletion using any appropriate method, e.g., considering the tree as an AVL-tree or red-and-black tree, see the chapter on trees.

Scene: *Nearest site search*

The main application of Voronoi diagram is to find a nearest site of a given point x of the plane. It is frequant that the set of sites is fixed and meny nearest neighbor queries are processed. In this case it is convenient to pre-process the information about the site set in such a way that single queries are executed faster.

A standard pre-processing first computes the Voronoi diagram of the site set, which is then used as follows:
First draw vertical lines through all Voronoi points - choose Vertical lines in the control bar to display the lines in the screen window. The lines partition the plane into vertical stripes of different width.

Voronoi segments cut stripes into trapezoidal regions (in many cases a trapezoid degenerates to a triangle, but we will use a general name “trapezoid” for any region). Select One stripe in the control bar and click different stripes to see the partition, or select All stripes to see a complete partition of the plane.

It is clear that all points in the interior have the same and unique nearest site, similarly for any line segment that is a side of a trapezoid its points different from the end points have the same and unique pair of the nearest neighbors. Finally there is just a limited number of points that are vertices of trapeziods, and the information about the set of the nearest sites (usually two or three) can be determined separately for any such point.

Given a point x of the plane, a nearest neighbor query for x is processed as follows:
first, the stripe containing x is determined by halving: if stripes scanned from left to right are $s_1, \ldots, s_k$, $k > 1$, then the middle stripe (i.e. s_ℓ, where $\ell = k/2$) is considered first, and it is easy to determine whether x is left to the stripe, inside it or right of it. If the point was found inside the stripe, the search is over, otherwise the number of stripes that could still contain the point was reduced by at least one half, and repeating the method resursively, the stripe can be found in time proportional to the binary logarithm of the number of stripes.
considering trapezoidal regions within the stripe containing x and using again halving, the region containing x can be found in time proportional to the number of trapezoids within the stripe. The answer to the query is found in the information that was associated with the trapezoid during pre-processing. Note that a query scheme for each stripe is different.

It can also happen that the point x belongs to a vertical interstripe line or to a boundary between two trapezoids in the same stripe. It is easy to implement the above search in such a way that it determines a boundary line segment or a vertex of a trapezoid that contains the point.

Scene: *Delaunay triangulation*

Delaunay triangulation of a set of finite points of the plane (sites) is a planar dual graph to the Voronoi diagram of the site set. Roughly speaking, two sites are connected if and only if their Voronoi regions have a common face (one common point is not sufficient).

Using checkboxes Voronoi and Delaunay it is possible to display or hide separately both the Voronoi diagram and the Delaunay triangulation.

Faces of the Delaunay triangulation correspond in 1-1 way to Voronoi vertices: if s_1, s_2, and s_3 are three nearest sites of a Voronoi vertex x, then s_1, s_2, and s_3 form one of the triangular sites of the Delaunay triangulation. Note that x need not belong to the triangle $\Delta(s_1, s_2, s_3)$.

Perhaps you remarked that a dual graph to Voronoi diagram is a true triangulation only if there are not 4 co-circular sites such that the interior of the circle determined by the sites contains no other site. If there is a "four corner" Voronoi vertex, its nearest sites (that all belong to a circle with the center in the vertex and not containing any other site) form a face of the Delaunay "triangulation" that has four sides, similarly for five or more "corner' vertex.

This scene is presented just as an information, no details concerning, construction, properties and applications of Delaunay triangulation are not given.

Scene: *Original Fortune's algorithm*

The original Steve Fortune's formulation of the algorithm was different, but both ways are just the same 3D algorithm as explained above, viewed under different angles.

The scene shows again 3D "mountains" viewed vertically. Animate the computation - it is seen as it was explained above: intersection of cones, even though they are hyperbolic curves, look like line segments representing segments of the Voronoi diagram.

Now replace the choice $\boxed{\text{Vertical}}$ in the control bar by $\boxed{\text{Fortune}}$: the mountains are seen under -45 degrees. If it is difficult for you to understand what happend, select $\boxed{\text{General view}}$, set $\boxed{\text{Angle}}$ to 0 to see the mountains vertically (the field sets the angle between a vertical line and the direction of observation of the mountains), then change the angle in steps from 0 to -45 and observe how the image of the mountains chages.

The direction we see the mountains is parallel to the southern slopes of cones and therefore parallel with the sweep plane (assuming parallel projection, i.e. the observer in infinity). This angle of view has one advantage and one disadvantage.

The advantage is that we observe the mountains in the direction parallel with the sweep plane, and therefore the sweep plane projects to a single line. Since both the sweep line and the beach line belong to the sweep plane, their projection are identical. In other words, we do not see separate sweep and beach lines, but just a single straight line. The "dead space" between the sweep line and the beach line in the vertical 2D projection is void. In this way Fortune's algorithm is a true sweep line algorithm, which draws Voronoi diagram immediately at all swept portion of the plane.

On the other hand, Voronoi segments do not any longer look as straight line segments and Voronoi regions are not polygons, but distorted in a strange way. It is important that any site now appears as the lowest point of its region, and therefore it is not necessary to start drawing a region prior its site has been swept (which was the case in the vertical projection).

It could seem that the original Fortune's algorithm is computationally too complex due to non-linear shape of Voronoi segments. However, only information we need are positions of Voronoi points and information which pairs are directly connected. This information is easily obtained during processint of site and circle events, and the the appearance of Voronoi segments under the projection is not important. Hence, it is sufficient to transform Voronoi points to vertical projection at the end of the computation by means of a simple linear transformation, and to connect pairs of transformed Voronoi points by line segments.

Scene: *Degenerated cases*

This scene shows several interesting degenerated configurations; choose any of examples using the choice in the control bar and animate the computation (it stops at all events).

Most of examples are configurations when several site and/or circle events occur in the same time and give Voronoi diagram that is composed of finite line segments and halflines. This means that there two or more events scheduled to the same position of the sweep line. The algorithm does not provide any particular way to process such multiple events and process the events on-by-one to get the correct result (but the applet does not make it possible to show intermediate results of such a computation).

However, there are several examples that deviates from what was stated in the previous paragraph.

If all sites belong to the same straight line (and only in this case) the Voronoi diagram is represented by a collection of parallel straight lines. Example 1 shows such a case, when the line containing the sites is *not* horizontal (see below). Note that no modificaton of the algorithm is necessary to handle this case. Repeat the example again with the checkbox Halfline checked to see that lines are drawn "from an interior point" as two halflines of opposite orientation that are drawn from their end-point.

Examples 2, 3, and 4 are the most exceptional case, which need amending the algoritm to handle the case. In Example 2 there are two southernmost sites (i.e. two sites have the same y-coordinate that is smaller than the y-coordinate of any other sites). In such a case the region of the two sites are separated by a line segment that is a part of a straight line going from south to north and is infinite on its southern side. Unlike any other halfline drawn by the algorithm in any situation, this halfline is not drawn from its end-point in the direction to the infinity, but in an exactly opposite way: when the sweep line hits both southern sites, the paraboles corresponding to the sites are degenerated to two parallel halflines starting in the sites and oriented exactly to south. If the sweepline moves infinitesimally small distance northwards, both halflines "open" to true paraboles that intersect in a finite point, which means that an infinite part of the halfline separating the southern site has already been drawn in an originally unbounded speed. This possibility must be handled in a way different from what was explained above.

Example 3 is similar to Example 2, but there are four southernmost sites. In Example 4 all sites have the same y-coordinate and hence the Voronoi diagram consists of several parallel straight lines that are not drawn from "middle" points as in Example 1, but from the "southern infinity".

As already mentioned above the remaining examples are just configurations, where at some moment (i.e. position of the sweep line) several site events and/or degenerated site events and/or circle events occur simultaneously.

Laboratory

School