Chapter 1

Voronoi Diagram

Scene: *Sites*

Display shows black dots in the plane, that will be called *sites*. AND collection of sites represents input data for and search of and Voronoi diagram. In this scene, we will just see how to manipulate the set of sites.

Here and in most of the following scenes it is possible to create new set of sites using the button **New input,** its size can be set it the field net to the label **N=.** In scenes, where they are available, buttons **Zvti** and **Zmeni** can be used to zoom the image, which often brings an interesting new view to the configuration.

In the control bar there is also a choice ot the mouse function (used in some other scenes):)
Choice **Move diagram:** Using the mouse, it is possible to move the plane, in which sites and the Voronoi diagram are drawn.
Choice **Move site:** Using the mouse, it is possible to catch a site and drag it arbitrarily in the window.
Choice **Add site:** Clicking into an empty space in the display creates a new site.
Choice **Delete site:** Delete an existing site by clicking onto it.

Scene: *Voronoi diagram*

This scene shows a Voronoi diagram of the selected set of sites. There is a region around any site in the form of a convex polygon, bounded by black lines. Some of boundary lines are line segments of bounded length, others are halflines.

Very exceptionally it can happen that the boundary is an infinite line (in this case the region is a halfplane) or 2 parallel line (in this case the region is an infinite stripe), but if there are more than 2 sites, these cases are very unlikely. The situations will be shown in the scene on degenerated examples.

Point of the plane aredivided into several categories:

- Interior of the region around a site s is composed of the points of the plane, that are closer to the site s than to any other site of the site set.
- Points that belong to the boundary of the regions correspondint to sites s_1 and s_2 are points that are equally close to s_1 and s_2 and closer to both s_1 and s_2 than to any other site. Such points belong to dividing line segments, halflines or lines, but there are not their end-points.
- Finally the diagram contains points that belong in the same time to boundaries of regions of three different sites s_1, s_2 and s_3. Such points have the same distance from sites s_1, s_2 and s_3, which is smaller that the distance to any other site. Such points are end-points of the boundary line segments and halflines.
- I general, there can be even points x having four or more nearest sites. However, such a case is very unlikely for a randomly constructed set of sites. A point x would belong to a boundary of at least four regions (similarly as the Four Corners point on the boundary of Arizona, Colorado, New Mexico and Utahu in the USA) and all of its nearest sites would belong to the same circle vit the center x. Such and other degenerated cases are discussed in detail later.

Points tha belong to boundary of three or more regions are called *Voronoi points*.
Change the set of sites by moving, adding and deleting of sites and observe changes of the diagram.

Scene: *Algorithm animation*

This applet shows, how a Voronoi diagram is constructed by an algorithm that was described by S. Fortune. In the next two scenes the algorithm will be presented using a 3D drawing. To avoid possible misunderstanding concerning the meaning of "up" and "down" the upper part of the display will be called "north" and the lower part of the display will be "south", using a convention common in the cartography. The terms "upper" and "lower" will be used in the 3D representation only, where I assume that the situation is observed from above, and hence "upper" will mean "closer to the observer" and "lower" will mean "farther from the observer."

Fortune's algorithm is a "sweep line" algorithm. After certain time a line appears in the display, which "sweeps" the plane from south to north and, at each moment, it shows a Voronoi diagram of site that have already been reached by the sweep line - more precisely that part of the diagram that will not change any more during the computation and will stay as a part of the resulting diagramu of the whole set of sites. The sweep line is not explicitly drawn and it is given by a moving boundary between the light and the dark of the plane.

A computation of the algorithm can be stopped by the button **Halt and** restarted using the button **Forward** or run backwards using the button **Backward.** It is also possible to jump to the end of the computation using the button **Finish** or go back to the beginning using the button **Reset** and then to start the algorithm again using the button **Forward** or **Backward.** Sweep line can also be moved using the mouse: push the mouse button outside of the control bar; the sweep line jumps to meet the point of the mouse push and remains attached to the cursor when the mouse button remains to be pressed.

There is a strange red surved line south of the sweep line. Due to its shape, the red cureve is usually called a "beach line" in the anglo-saxon literature. The Voronoi diagram is drawn in the region *south* of the beach line. This is because (as shown below) that the shape of the resulting diagram in the region south of the beach line is given only by sites that have already been "swept" by the sweepline, while the form of the diagram in the region between the beach line and the sweep line can still the sweep line. Of course, note that nothing of the diagram could be determined in the region north of the sweep line.

When observed more closely, it can be found that the beach line is made of several arcs (it will be shown later that the arcs are parabolic arcs) and the points, where the arsc meet, draw the Voronoi diagram. The reason for this fact will become obvious in the following scenes.

Scene: *Cones*

This scene and the next one bring an alternative view of the situation that has been described in the previous scenes. The view can make it much easier to understand the algorithmic idea of the Fortune's algorithm, but I have found that certain readers have difficulties to understand it. Since the scenes are not absolutely necessary for the explanation of the algorithm, it is possible to skip tham and possibly go back later, even though I think 3D view makes the understanding easier and deeper.

The 3D scenes are computationally intensive, and hence there might be certain delays on older computers.

This scene shows the same situation as the previous one, but represented in three-dimensional space. The planes involving all the input sites is embedded into the space in such a way that it is horizontal and observed from above. we assume that is observed from a big distance (perhaps from the infinity) and consequently effects of perspective are suppressed and the display shows a parallel vertical projection into the plane of input sites.

Every site is the top of a cone with a vertical axis and shaded as if there was a source of the light on the left. The surface lines of the cones have a declination 45 degrees. What we get is a mountain range, where particular mountains have the same regular shape and their tops (our sites) are in the same altitude. Viewed fromanother cone.

When the mouse moves up end down having the left button pressed, the mountains start to incline - observe them from a side view. Moving the mouse left or right rotates the mountain. (Only part of the mountain is shown - its cutout that was visible in the window when observing it vertically; during tests most users claimed this method as easier to be understood). Clicking the button **Vertical** it is possible to restore the original vertical position.

[Sorry: *Triangles explained in the following paragraphs of the scene are not yet implemented*]

Click now by the right button of the mouse to any point of the surface of the cone moutains. A rectangular equilateral triangle appears with different colors of its sides (if you see just a red line or if the triangle is too narrow rotate the mountains to see the triangle better). A green hypotenuse of the triangle shows a descent from the top of the cone that bears the selected point on its surface, along the spdov line of the cone to the selected point. A red triangle base is the projection of the green hypotenuse into the horizontal site plany. Finally a yellow height side shows how deep is the selected point below the site plane. The triangle is rectangular, because

the yellow height is perpendicular to the site plane and hence perpendicular to the red base as well, and since the declination of cone surface lines is 45 degrees, the triangle is also equilateral.

Move the cursor having the right mouse button pressed to see the triangle in different positions, possibly moving the mountains to a different position to make the triangle better visible (having the left mouse button pressed). Finally let the mountains move back to the initial position (e.g., using the button **Vertical)** to verifz that what can bee seen in this position as a line segmant connecting the top of the cone and the selected point is in fact the projection of their true connection in the space.

You have perhaps noticed that if the selected point belongs to an intersection of two cones, two triangles appear (and three or even more triangles can appear in Voronoi points of the diagram). Put the selected point to a visible intersection of two different cone surfaces and rotate the mountains so the triangles are well visible. Both triangles are rectangular and equilateral ant thez have a common yellow side. As a consequence, the triangles are identical and hence their red sides have the same length. Put now the mountains again to their initial position and observe them vertically. Now, red triangle bases give distances of the selected point, as they appear in their projections to the site plane. This proves what, I guess, is already clear to anybody: the projections of the visible cone intersections is exactlz the Voronoi diagram of the set of the input sites (cone tops).

To hide the triangles, uncheck the checkbox **Triangle.** If **Triangle** is checked again, it is necessary to click somewhere by the red mouse button to have the triangle displayed again.

Scene: *Sweep plane*

Let us now imagine that the mountains of the previous scene is "swept", but by a green plane rather than by line. The plane declination is 45 degrees (only rectangular section of the plane is shown). Parts of the mountains that are *below* the plane are blue (from light to dark blue), while parts above the sweep plane are gray (again from light to dark). The intersection of the green plane anf the cone surfaces is red.

Push button **Ahead** and observe the plane movement. The sweep plane slowly uncovers parts of the cones. Wait until the plane is approximately in the middle of the mountains and push the button **Halt.** After a thorough inspection of the way the mountains are cut by the sweep plane (possibly rotating the mountains), push the button **Vertical.** The mountains return to the initial position known from the previous scenes, when the cones are observed vertically from above in the direction parallel with their axis.

When the mountains are viewed in their initial position, the intersection of the sweep plane with the horizontal site plane represents the sweep line from the previous 2D scene. You obviously see that the intersection of the sweep plane and the visible cone surfaces (or, more precisely, its projection to the site plane) is the beach line that was shown in the original 2D animation of the algorithm. Points that are south of the beach line in the 2D scenes are projections of points of the visible parts of the cone surfaces that are *above* the sweep plane in the 3D scene. Points that are north of the beach line in the 2D scenes are projections of points of the visible parts of the cone surfaces that are *below* the sweep plane in the 3D scene.

The first moment, when the sweep plane touches a cone of some site, is the moment, when the plane touches the top of the cone (i.e., a site). In the same time the plane touches a cone in a surface halfline that goes from the top down to the south.

Now it is clear, how the sweep line is constructed: it is the projection of the intersection of the sweep plane and visible parts of the cone surfaces. Since the sweep plane is parallel to a surface halfline of a cone, the intersections of the sweep place and cone surfaces is a parabola and ist projecton to the site plane is a parabola as well. This shows that the beach line in 2D scenes is really composed of parabolic arcs.

Valeys of the cone mountains are points, the projections of which form boundary segments of regions of a Voronoi diagram in 2D. These points are also points, where visible intersections of the cone surfaces meet the sweep plane. This is why meeting points of arcs of the beach line in 2D reprezentaci "draw" the boundaries of Voronoi regions: all are projections of points of valey bottoms.

Scene: *Single parabola*

In previous scene we explained the meaning of the beach line using a 3D view of the configuration. In this and the following scenes we explain the meaning of the beach line once more, but using its planar drawing. We will again meet cone sections, in particular parabolas, but not as an intersection of a cone and a plane, but as a set of points equidistant from a point and a line.

As I have already explained above, the algorithm tries to construct a part of the Voronoi diagramu that can be determined using just knowledge of sites, that have been swept (or at least touched) by the sweep line.

Such sites will be called *known sites*. Sites that are still in the dark north region (north of the sweep line) are *unknown sites*.

The figure shows all sites and the sweep line. Click on known site. The site is emphasized and a pink region apears. The region is upper bounded by a parabolic curve. The boundary is a set of all points, that have the same distance from the selected site and the sweep line (in its actual position). It is known that such a curve is a parabola that has a *focus* in the selected site; the sweep line is the *directrix* of the parabola. The parabola is open southward and the points of the pink region are the points that are closer to the focus than to the directrix, while points that are not pink and do not belong to the parabola are (regardless of their position with respect to the sweep line) closer to the directrix of the parabola (i.e., the sweep line) than to the focus.

Note that the point of the parabola that is north of the selected site is the point that is in the middle of the distance from the focus to the directrix.

Choose another known site; again you obtain a parabola and a pink region of points closer to the selected focus than to the directrix. The directrix (sweep line) can be moved by the buttons of the control bar as in the previous scenes. A movement stops when the selected site is hit by the sweep line (to prevent its passing to the region of unknown sites). The sweep line can not be dragged by the mouse to avoid colision with selecting sites by the mouse.

Observe changes of a parabola shape when the sweep line, representing the directrix of the parabola, moves. If the directrix is close to the focus, the parabola is very narrow, but opens as the directrix moves northward.

A special situation occurs when the focus (i.e., the selected site) belongs to the directrix. In this case the set of points that are equidistant from the focus and the directrix is not a parabola, but a line passing through the focus and perpendicular to the directrix. This case is also important for the computation according to the Fortune's algorithm and occurs during so called site event. In the present setting we are only interested in a part of the line (a halfline) that starts in the focus and extends to the south. We can view of this halfline as a degenerated parabole of the zero width. The above mentioned pink region is now void; when the focus belongs to the directrix, no point can be closer to the focus than to the directrix.

Note that the degenerated situation of the preceeding paragraph can be explained in 3D setting as well: in this case the sweep plane includes the top vertex of a cone (a site) and touches the cone surface in a single halfline, while later it intersects the surface in a proper parabola.

Scene: *Parabolae together*

In this scene all parabolae of known sites, as explained in the previous scene, are shouwn in the same time. The pink region in the display is the union of the pink regions of all known sites. The pink region is the set of points of the plane that are closer to one of the known sites than to the sweep line.

Red points are points of a parabola of some known site that are not elements of the pink region of any other known site. Such points are points such that the distance to the nearest known site is the same as the distance to the sweep line. It is clear that this set is the beach line of the previous scenes.

Remaining points north of the beach line are points that are closer to the sweep line than to the nearest known site. Note that in this case it is possible that the sweep line hides an unknown site that is closer than any of the known sites. This is why the form of the Voronoi diagram can not be determined in the region between the beach line and the sweep line, assuming that we are not allowed to use information about unknown sites.

[Sorry: *The scene "Beachline" that is still in the applet was deleted from the guide*]

Scene: *Events*

The shape of the beach line always changes as it moves to the north, but the number of arcs and their foci change only infrequently. Momnts when the beach line substantially changes are called *events*. This scene animates sweeping the sites using buttons **Upwards** and **Dawnwards** similarly as in the third scene, but the movement automatically stops not only when the button **Halt** is pressed, but also when an event occurs.

Let the sweep line move to the north and observe events that breaks the computation. There are two cases:

- **Site event:** The sweep line hit another site (which explains the name of the event type). A new but degenerated "arc" appeared in the beach line. In fact, the "arc" is a line segment that is a part of the halfline representing e degenerated parabola of the new known site that belogs to the sweep line in the moment of the site event. The "arc" cuts into two parts an arc that is below the new know site on the original beach line.

- Circle event: One of the arcs of the beach line has been "oppressed" by two neighboring arcs, its width decreased and the arc completely disappeared in the moment of an event. (The name of this type of event will be explained later).

Occasionally some other types of events can occur. Certain among them can be viewed as a simultaneous occurence of two or more simple events; e.g. erasing of two or more contiguous or non-contiguous arcs in the same time. In the case of a site event, it can happen that a vertical halfline starting in a new know site hits the beach line not in the middle of an arc, but in the point where two arcs meet. The latter event must be dealt with in a special way. Such configurations are discussed in a special scene later.

Play with the applet and observe situations that occur during events. I recommend observing configurations that appear just before or after an event (in particular before a circle event and after a site event).

Scene: *Site event*

The scene is similar to the previous one, but the sweep line halts at site events only.

Observe what happens in the moment of a site event and immediately after it. Before a site event nothing signals that a new event occurs. The situation north of the sweep line is unknown and there is no possibility to find that the line aproaches an unknown site.

In the moment of an event a new site appears on the sweep line. A new degenerated parabole is included into the list of arcs of the beach line. Assuming a usual site event, the halfline of the degenerated parabola cuts certain arc of the beach line into two parts and is inserted between them. The cut point has the same distance to the newly discovered site on the sweep line and to the focus of the parabolic arc that has been cut. This is why a new segment of the Voronoi diagram starts in the cut point. At the moment of a site event the new Voronoi segment is represented by a single point, the cut point.

Now, using the mouse, move the sweep line very slowly to the north. The degenerated parabola becomes an extremely narrow proper parabola. However, the parabola opens very quickly. Intersection of the branches of the parabola with the west and the east parts of the cut arc of the original beach line move swiftly in the opposite directions and draw a new segment of the Voronoi diagram. It is obvious that the intersection points move along the line which is the axis of the line segment connecting the new know site and the focus of the arc that was split.

When writing a code for a computer, it is convenient to assume that the new segment of the Voronoi diagram is not drawn in two directions, but that they are two new segment, starting in the point where a degenerated parabola of the new site cut the arc. The new segments are both drawn unidirectionally by intersection points of the branches of the parabola having the new site as a focus with the arc that has been split. Activate the checkbox **Halflines** to illustrate this approach. The two segments are drawn as arrows having an explicitly marked starting point.

Scene: *Circle event*

Roughly speaking, a circle event is a decay of certain arc of the beach line, when its neighbors touch in the point where the arce has disappeared. It is possible to say that the arc died due to expansion of its neighbors.

Unlike in the case of a site event, a circle event can be forecasted sometimes long time before. However, it can happen that a planned circle event would not occur, because another event unplanned event takes place and changes the situation in such a way that the planned event is meaningless (e.g., it can happen than an aggressive neighbor arc is eliminated before it kills the squeezed arc).

Let the sweep line move (the buttons **Forward** or **Backward)** until it halts in the moment of a circle event, and then move it slightly back.

Now, we observe the situation just before a circle event that is going to eliminate a short arc α having the focus s_α, which is squeezed by neighboring arcs β and γ having respective foci s_β and s_γ. The lines that represent axis of the line segments $s_\alpha - s_\beta$ and $s_\alpha - s_\gamma$ are lines that guide the meeting points of the respective arc pairs (α, β) and (α, γ). The lines intersect in the point in which the arc α disappears soon.

[Sorry: *The following interaction is not yet fully functional in the applet*] Let the sweep line to move again; the circle event happens. At this moment activate the checkbox **Parabolae.** Two full parabolae appear that are determined by the sites s_β and s_γ, and that involve the arcs β and γ, the killers of the arc α. The intersection point, where the arc α disappeared, is also on the parabola with the focus s_α. The parabola is represented as well (but using the green color), because the arc α is still in fact a part of the beach line, even though it degenerated to a single point. Hence, the point c, where the arc α disappeared, belongs in the same time to three parabolae with the foci α, β and γ.

This implies that the distances from the intersection point c to all three sites s_α, s_β and s_γ are the same as the distance from the point c to the sweep line. It follows that the circle with the center c and passing through s_α passes through the sites s_β and s_γ as well. Moreover, in the moment of the site event the sweep line touches the upper point of above mentioned circle. To show the circle check the checkbox **Show circle.**

Now, it is already clear how to forecast a circle event that kills an arc α having neighbors β and γ (unless invalidated by an earlier event):

1. draw a circle passing through sites s_α, s_β and s_γ; the arc α is supposed to disappear when the center of the circle is in a point that will be hit by the beach line in the future;
2. if the arc α disappears, it is eliminated in the center of the circle;
3. if the arc α disappears, the event occurs in the moment when the sweep line touche the uppermost point of the circle, i.e., when the y-coordinate of the points of the (horizontal) sweep line is equal to the y-coordinate of the circle center plus r, where r is the radius of the circle.

A role of a circle when forecasting a circle event explains the name of the event.

[Sorry: *The following three scenes are not yet implemented*]

Scene: *Circle event - planning*

When the computation is interupted, select the choice **One circle** and click certain arc of the beach line. If it appears that the selected arc would disappear as a result of a future circle event, a circle appears that passes through the foci of the selected arc and its neighbors in the beach line. The circle can be understood as a forecast of a future circle event.

However, if the selected arc expands as long as the sweep line mves to the north (i.e., the distance of its end-points gets larger), no circle appears.

When selecting the choice **All circles,** the circles of all expected future circle events appear in the same time. The figure is usually a bit messy, but it gives an idea of how many circle events are planned.

Scene: *Failed circle event*

This scene shows several model configurations where the emphasized arc of the beach line is supposed to disappear, but finally the planned event does not occur. The corresponding circle is shown as well.

In the example 1 the emphasized central arc is cut by a parabola of a site that appears at the moment of an earlier site event.

In the examples 2 and 3 the neighbors of the emphasized central arc are eliminated by earlier site events. However, one can argue that these two cases mean that the original circle events are preserved in a modified form - a neighbor arc that have been cut during a site event is replaced by its part that is adjacent with the emphasized central arc. Note that the original neighbor and its part have the same focus, and therefore the replacement event has the same circle and hence it happens in the same time and the arc is planned to disappear in the same place.

In the examples 4 and 5 the neighbors of the emphasized central arc are eliminated by earlier circle events. In this case the originally planned event is replaced by another circle event, too. However, unlike in the previous case, the replacement is substantial - a replacement neighbor arc has a different focus and hence the circle of the event is quite different.

In the examples 6 and 7 an earlier site event occurs and its degenerated parabola hit the beach line in the points where the emphasized central arc of the planned event meets with its neighbor. Once again, the central arc gets a substantially different neighbor arc having a different focus. Thus, the planned circle event is replaced by a circle event that has a different circle.

Scene: *Event calendar*

In order to schedule events, the algorithm uses an *event calendar*. In the previous scenes we have implicitly assumed that the computation is performed on-line and the sites become known only when the sweep line encounters them. However, if solved off-line, a complete set of sites is given at the beginning of the computation. In such a case all sites can be inserted into the event calendar according to y-coordinates of sites, which tell us when the sweep line hits them.

Circle events are inserted when a change occurs on the beach line such that it is possible to expect that a circle event will occur in the future. Data describing a single circle event involve an information on foci of 3 arcs,

and the center and the radius of the corresponding circle. A data record is stored according to the y-coordinate of the upper point of the circle (i.e., the y-coordinate of the circle center plus the circle radius), which gives the position of the sweep line in the moment of the corresponding event.

In the display, the calendar is represented by black and magenta items drawn in the right part of the window. Black items correspond to site events and a nib on the left side of an item gives the y-coordinate of the corresponding site (and hence it determines the position of the sweep line in the moment when the event occurs). When clicked, a calendar item is emphasized together with the corresponding site. Tha same happens when a site is clicked.

Magenta items correspond to circle events; the set of planned circle events change during the computation. A nib of an item determines the y-coordinate of the upper point of the corresponding circle (and hence it determines the position of the sweep line in the moment when the event occurs). When clicked, a calendar item is emphasized together with the central arc of the event, and the event circle is shown. The same happens when the central arc of the event is clicked.

As already explained, certain planned circle events are erased from the event calendar before they occur during an earlier site or circle event. On the other hand, certain new circle events must be planned during events. Thus, when an event occurs, both events to be erased and new events to be planned are displayed. The former ones have temporarily gray color, the latter ones are green. Of course, after the event the gray calendar items disappear and the green ones get their normal magenta color of circle events.

It happens quite often that events are planned so that the corresponding calendar items overlap and the calendar becomes messy. In such a case it is possible to deactivate the checkbox **Exact position** and the calendar items are re-arranged vertically so that they do not overlap. However, in this case the vertical position of the items doesn't correspond to the position of the sweep line in the moment of the event (and hence nibs on the left side of items are not shown). The correspondence of events with sites or circles and arcs, activated by mouse clicking, is preserved.

Scene: *Degenerated events*

This scene shows 8 different site configurations that generate degenerated events. Configuration are selected by choosing **Example xx** in the control bar. After having chosen an example, use the button **Forward** to animate a computation that halts at all events.

Example 0 shows two southernmost sites. Start an animation. When the sweep line arrives to the sites, the beach line is represented by two separated parallel halflines oriented to the south, but not intersecting. After having moved an infinitesimally small distance to the north, two halflines become two proper intersecting parabolae. Their *unique* intersection arrives quickly from the infinity as long as the sweep line slowly moves to the north. This is the only case, when a halfline that makes a part of the Voronoi diagram is drawn "from the infinity". When writing a code, this case must be handled separately. In Example 0 drawing the halfline is stopped by a circle event shortly after the sweep line hits the third site of the configuration.

In Example 1 when the sweep line hits the northernmost site, two events occur in the same time: a site event corresponding to the site and a circle event. These two events are independent and could be processed one after another in an arbitrary order.

In Example 2, a site event corresponding to the northermost site is of a special type. A halfline (degenerated parabola) going south from the new site hits the beach line in an intersection of two arcs. In such a case the segment that has been drawn by the intersection ends and two new Voronoi diagram segments start. The segments are drawn by intersections of a parabolic art of the new site with two arcs of the original beach line. This event must be handled in a special way.

Example 3 is similar, but the site event occur simultaneously with a circle event affecting one of the involved arcs. In this case the circle event should be processed before the site event.

In Example 4 two circle event occur simultaneously and two adjacent arcs disappear. The events can be processed in an arbitrary order.

In Example 5 all sites belong to one line. Such a configuration that must be processed separately results in a Voronoi diagram that involves $n-1$ parallel lines (where n is the number of sites) that divide the plane into two halfplanes separated by $n-2$ stripes of infinite length.

Example 6 is similar, but all sites have the same y-coordinates. Their Voronoi diagram is again represented by $n-1$ parallel lines that delimit two halfplanes and $n-2$ stripes. This configuration that generates Example 0, must be handled separately and in a way different from the previous example.

Finely Example 7 brings nothing new, but it exhibits simultaneously 9 events that eliminate 9 adjacent arcs simultaneously with a site event, the degenerated parabola of it hits the point, where all arcs disappeared. Even

though it looks quite complex, the combined event is processed by sequentially processing 9 circle events in any order, followed by the site event.

Scene: *Arc search*

[Sorry: *In the present scene the tree is correctly displayed, but after the last amendment of the code a searching in the tree is not working*]

When a site event occurss, it is necessary to determine an arc of the beach line that is below the new site. Even though complex calculation is necessary to find arc intersection points, each intersection is evaluate in constant time. However, the beach line generally involves a large number of arcs (proportional to the number of sites in the worst case) and hence if arcs are inspected left to right, it takes long time to find the proper arc.

If the computational time is important, it is advantageous to build a binary search tree over the beach line to speed up the arc search.

Leafs of the tree (orange dots) represent beach line arcs (each leaf is placed above the corresponding arc) and internal nodes of the tree (gray dots) correspond to meeting points of adjacent arcs (an internal node of the tree is always drawn above the corresponding meeting point of arcs). Any internal node includes an information about foci of arcs, that meet in the point below the node (distinguishing the focus of the left arc from the focus of the right arc).

Edges of the tree are green. The correspondence of tree nodes and elements of the beach line is marked by vertical lines of orange and gray colors (the lines are not parts of the tree; they just illustrate a logical correspondence of the tree nodes and beach line elements). Vertical lines and the tree itself could be hidden using a proper choice in the control bar.

It is necessary to mention that the shape of the tree is not important for correct function of the algorithm. Arcs of the beach line determine the set ot tree leafs, but it is not important how the structure of internal nodes of the tree is built. However, in order to get a fast computation, it is advantageous when the tree is well balanced.

Click the button **Forward.** After some while, an even occurs and the computation halts.

When a site event occured, standard animation buttons appear and you can use them to animate searching for an arc below the site. The search starts in the root and a token moves as follows:

if the token is in an internal node of the tree, then the node corresponds to a meeting point of two adjacent arcs of the beach line. If the x-coordinate of the arc meeting point below the node is smaller than the x-coordinate of the site with the green background that has been found the sweep line then the token moves to the *right*, if it is larger, the token moves to the *left*. In a rare case when the x-coordinates are equal, a special case occured (see, e.g., Example 2 of the scene on degenerated events).

The token eventually reaches a leaf of the tree, and the leaf corresponds to the arc below the event site that we have been looking for.

When the arc is found, it is necessary to update the tree by splitting the arc below the site into two parts and inserting a new arc (at the moment degenerated) between them, using the button **Update.** This operation increases both the number of arcs of the beach line and the number of leafs of the tree by two (one arc split, one added). This also induces increase of the number of internal nodes of the tree by two. After the arc below site is found and the tree updated, click the button **Finish** or **Reset** to continue (the later one undoes the changes made in the tree).

Of course, if a special site event occurs (the site exactly above the meeting point of arcs), only one arc, one tree leaf and one tree internal node are added.

When using a tree, the number of steps that have to be performed during an arc search is proportional to the depth of the tree rather than the number of arcs of the beach line (i.e., the number of leafs of the tree).

In the optimal case the tree depth is proportional to the logarithm of the number of arcs of the beach line; it is obvious that a substantial speed-up of the arc search is obtained. If an unbalanced tree is used, it can happen that the tree depth becomes much higher than the optimal value. In order to prevent this, it is possible to use some kind of balanced trees, e.g., an AVL-tree or a red-black tree, see Capter 2. In the present scene balancing of a tree is not implemented, and a simple BVS is used.

Scene: *Meeting of beach line arcs*

[Sorry: *This is a new scene that has not yet been implemented in the applet*]

This scene is a small technical detour. It shows how one can find coordinates of a point, where two adjacent beach line arcs meet. The scene shows the sweep line, given as a boundary of a light and a dark region of the

window and two sites that have respectively red and green color. Let us suppose that we know numeric values of the position of the sweep line and both sites.

As it has already been said, an equation of a parabola with a given focus and a given directrix can be determined quite easily: if $F = [f_x, f_y]$ is the focus, d is the y-coordinate of the sweep line that is north of the focus (i.e., $f_y < d$, and $P = [x, y]$ is a general point of the parabole, then
$d - y$ is the distance of P from the directrix,
$\sqrt{(x - f_x)^2 + (y - f_y)^2}$ is the distance of P from F,
and hence the equation of the parabola is

$$(d - y)^2 = (x - f_x)^2 + (y - f_y)^2,$$

which gives

$$y = \frac{-x^2 + 2f_x x - (f_x^2 + f_y^2 - d^2)}{2(d - f_y)}.$$

Since we have two sites, we have also two paraboles and two equations. Equality of right hand sides of equations give a quadratic equation for the x-coordinate of the intersection, and y-coordinate is computed from the above equation. The quadratic equation has generally two solutions and it is a bit complicated to determine the righ one.

When the scene is opened, the y-coordinates of both red and green sites are the same. In this case both parabolae are open in the same way and the quadratic equation has just a single solution (of multiplicity 2). There is no problem to determine the meeting point of arcs.

Now use the mouse to move the red site to the north (increase its y-coordinate). The parabola with the red focus gets narrower and cuts a section of the parabola of the green focus and a red arc in inserted into the space obtained in thic way. In this case the beach line consists of a green (unbounded) arc, a red arc and another green arc. If we know that two adjacent arcs of the beach line are green and red, then if a green arc is on the west of the intersection point, and a red arc is on the east, then the intersection is given by the smaller solution of the quadratic equation. On the other hand, if red is on the west and green is on the east, we have to use the larger solution. Recall that red corresponds to the focus which is more to the north, green is southern.

Change the position of sites so that northern site is green. In this case the situation is quite opposite. An arc of the southern site on the west (east, resp.) corresponds to the smaller (larger, resp.) solution of the quadratic equation.

Scene: *Nearest point*

The fundamental application of Voronoi diagram is a problem of finding the nearest element from a given set of sites for a given point x of the plane. A set of sites is often fixed, but a large number of queries for the nearest sites must be solved. In such a case it is convenient to pre-process the site set in such a way that nearest site queries are performed as quickly as possible.

It is recommended to pre-process the site set as follows:

First, construct a Voronoi diagram of the site set as shown in the display.

Then draw vertical lines through end-points of all segments of the Voronoi diagramu. Choose **Vertical lines** in the control bar to show the lines.

The lines divide the plane into vertical stripes, and the stripes are cut by segments of the Voronoi diagramu into trapezoidal regions (in some cases a trapezoid degenerates to a triangle, but we will use always the term trapezoid).

Now, choose **One stripe** to emphasize partition of one vertical stripe; click stripes and observe how they are partitioned to regions. Finally choose **All stripes** to color all trapezoidal regions.

The way how trapezoidal regions were constructed is such that any region is *whole* a part of a Voronoi region of certain site, and hence all internal points of any trapezoid have the same nearest site. The same site is also the nearest site of points of the trapezoid perimeter, but is is not generally the unique nearest site.

A line segment on the perimeter can be within a Voronoi region of some site or it can be a part of a segment of a Voronoi diagram that is an intersection of Voronoi regions of two sites. In both cases all points ot the interior of a side of a trapezoid have the same information about the nearest site(s).

Finally there is a finite list of points of the plane that are vertices of trapezoids. For each such vertex we determine easily two or more nearest sites..

The pre-processing finishes by associating trapezoids and their parts (interior, sides, vertices) with the information about the nearest site(s).

When processing a nearest site query for a give point ot the plane, it is sufficient to determine the trapezoid that includes the point and read the associated information.

In order to determine the trapezoid, a vertical stripe containing the input point is determined first. This task can advantageously be found by halving. Let $x_1 < \cdots < x_\ell$ be x-coordinates of vertical lines dividing the stripes. The x-coordinate of the query point is compared with $x_{\ell/2}$; depending of the outcome that tells us whether the query point is located left or right to the middle vertical line (or on it) the x-coordinate of the query point is compared with either $x_{\ell/4}$ or $x_{3\ell/4}$ etc., and after at most $\lceil \log_2 \ell \rceil$ steps the stripe including the query point is identified.

Then, within the determined stripe, a trapezoid including the query point is found using similar, but more complicated halving scheme; again, it is possible to identify the trapezoid in logarithmic number of steps.

Thus, if the site set is pre-processed in a proper way, each subsequent nearest site query is answered in logaritmic numbero of steps..

Scene: *Delaunay triangulation*

Un this scene I will show a Delaunay triangulation of a set of sites in the plane. The triangulation is a dual graph to the graph represented by a Voronoi disgram. Roughly speaking, two sites are connected by a blue line segment if and only if their Voronoi regions have a common side (one common point is not enough).

If four corner Voronoi points do not exist, then the blue lines of the Delaunay diagram divide the plane into triangular regions, and hence the diagram is called Delaunay triangulation.

Using checkboxes **Voronoi** and **Delaunay** it is possible to show and hide both diagrams. Show them in the same time and decide whether you would be able to draw a Delaunay triangulation if a Voronoi diagram is known.

This scene is just an information about an important notion; a construction, properties and applications of Delaunay triangulations are not discussed in details.

Scene: *Original Fortune's algorithm*

The article of Steve Fortune described algorithm for construction of a Voronoi diagramu in a different way. In order to show his approach, we will use 3D visualization once more. Mountains of cones is first show in the same way as when the scene "Cones" opened, that is vertical view. However, the mountains immediately start to incline, and the movement halts when the view angle is 45 stup, see a small head in the corner. This means that at the end we observe the mountains in the direction of the south surface lines of cones (from the infinity or from a very large distance to avoid efects of perspective).

The selected angle of view has one advantage and one disadvantage.

An advantage is that we observe the mountains in the direction that is parallel with the sweep plane that was shown in the scene "The sweep plane". Instrad of a plane, we see a single line, and all points of the plane are projected to the observed line. In particular, this means that the beach line is projected to the observed line as well, because it is an intersection of the sweep plane and cone surfaces.

Thus, in the original version of the Fortunes algorithm there is no separated beach line no dead region between the sweep line and the beach line, that was in the "swept" region, but still potentially influences by yet unknown sites. For this reason the original algorithm was a pure sweep *line* algorithm as it is known from many other problems in the computational geometry.

On the other hand segment and region boundaries of a Voronoi diagramu become represented by curves of degree 2. In the 3D visualization Voronoi segments, separating regions of sites, are not line segments, but hyperbolic arc, that are intersections of surfaces of two cones with parallel axis. They look like line segments only when observed vertically, but Fortunes direction of view is not vertical and hyperbolic curves do not project as line segments, but as proper curves. As a consequence Voronoi regions of sites are deformed in a special way.

It is important that the south surface lines of cones (that involve the cone top a site) look as points when observed according Fortune. Moreover and proto se kad site jev jako nejni point sv regioni. Hence a site looks as the southernmost point if its region, which is hit as soon as the sweep line touches the region. In this way, when the algorithm starts to draw a region of certain site, the information obout the site is available. On the other hand, when observed vertically, the sweep line traverses a large part of a Voronoi region before it encounters the corresponding site.

It might seem that the computation according to original algorithm is extremely difficult, because boundaries of Voronoi regions are complex curves, but in fact we need to know just end points of the segments that are determined during events, and hyperbolic curves of 3D situation are not needed. At the end of the computation all segment end-points are transformed in an easy way to vertical view and connect by line segments.

The view direction can be controlled by the mouse in the same way as previously, and the respective buttons **Vertical** and **Fortune** set the view angle to vertical/Fortune direction.