Lecture 2 (CGAL)

Panos Giannopoulos, Dror Atariah
AG TI

WS 2012/13
- New CGAL release 4.1
- Do the exercises!
- Campus Management?
Outline

Numerics, Number types, Kernels

Convex Hull

Exercise 2
**CGAL Numerical Issues**

```cpp
typedef CGAL::Cartesian<NT> Kernel;
NT sqrt2 = sqrt(NT(2));

Kernel::Point_2 p(0,0), q(sqrt2, sqrt2);
Kernel::Circle_2 C(p, 4);

assert(C.has_on_boundary(q));
```

- OK if NT supports exact sqrt.
- Assertion violation otherwise.

**Note:** `assert()` will only run in debug mode
Use: `-DCMAKE_BUILD_TYPE=Debug` with `cmake` (2nd run)
Example
CGAL Pre-defined Cartesian Kernels

- Support construction of points from `double` Cartesian coordinates.
- Support exact geometric predicates.
- Handle geometric constructions differently:
  - `CGAL::Exact_predicates_inexact_constructions_kernel`
    - Geometric constructions may be inexact due to round-off errors.
    - It is however more efficient and sufficient for most CGAL algorithms.
  - `CGAL::Exact_predicates_exact_constructions_kernel`
  - `CGAL::Exact_predicates_exact_constructions_kernel_with_sqrt`
    - Its number type supports the exact square-root operation.
Computing the Orientation

- imperative style

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;

typedef Kernel::Orientation_2 Orientation_2;

int main()
{
    Point_2 p(0, 0), q(10, 3), r(12, 19);
    return (CGAL::orientation(q, p, r) == CGAL::LEFT_TURN) ? 0 : 1;
}
```

- precative style

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;
typedef Kernel::Orientation_2 Orientation_2;

int main()
{
    Kernel kernel;
    Orientation_2 orientation = kernel.orientation_2_object();

    Point_2 p(0, 0), q(10, 3), r(12, 19);
    return (orientation(q, p, r) == CGAL::LEFT_TURN) ? 0 : 1;
}
```
Orientation1
Orientation2
Convex Hull Terms and Definitions

**Definition (convex hull)**

The **convex hull** of a set of points $P \subseteq \mathbb{R}^d$, denoted as $\text{conv}(P)$, is the smallest (inclusionwise) convex set containing $P$.

When an elastic band stretched open to encompass the input points is released, it assumes the shape of the convex hull.

- $n$ — the number of input points.
- $h$ — the number of points in the hull.

**Time complexities of convex hull computation:**

- Optimal, output sensitive: $O(n \log h)$. [Chan96]
- QuickHull (expected): $O(n \log n)$. [BDH96]
Convex Hull Terms and Definitions

Definition (convex hull)

The **convex hull** of a set of points $P \subseteq \mathbb{R}^d$, denoted as $\text{conv}(P)$, is the smallest (inclusionwise) convex set containing $P$.

When an elastic band stretched open to encompass the input points is released, it assumes the shape of the convex hull.

$n$ — the number of input points.

$h$ — the number of points in the hull.

- Time complexities of convex hull computation:
  - Optimal, output sensitive: $O(n \log h)$. [Chan96]
  - QuickHull (expected): $O(n \log n)$. [BDH96]
Convex Hull Properties

- A subset $S \subseteq \mathbb{R}^d$ is convex $\iff$ the line segment $pq \in S$ for any two points $p, q \in S$.
- The convex hull of a set $S$ is the smallest convex set containing $S$.
- The convex hull of a set of points $P$ is a convex polygon with vertices in $P$.

- Input: set of points $P$ (or objects).
- Output: the convex hull $S \subseteq P$ of $P$. 
**CGAL Convex Hull**

```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/convex_hull_2.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;

int main() {
    CGAL::set_ascii_mode(std::cin);
    CGAL::set_ascii_mode(std::cout);
    std::istream_iterator<Point_2> in_start(std::cin);
    std::istream_iterator<Point_2> in_end;
    std::ostream_iterator<Point_2> out(std::cout, "\n");
    CGAL::convex_hull_2(in_start, in_end, out);
    return 0;
}
```

```text
p1
p2
p3
p4
p5
```

*Computational Geometry Algorithm Library 55*
The edge \( pq \) is visible from \( r \) \iff \( \text{orientation}(p, q, r) < 0 \)

The edge \( pq \) is weakly visible from \( r \) \iff \( \text{orientation}(p, q, r) \leq 0 \)

Maintain the current convex hull \( S \) of a set of points seen so far

1. Initialize \( S \) to the counter-clockwise sequence \( \{a, b, c\} \subset P \)
2. Remove \( a, b, \) and \( c \) from \( P \)
3. \textbf{for all} \( r \in P \) \textbf{do}
4. \hspace{1em} \textbf{if} there is an edge \( e \) visible from \( r \) \textbf{then}
5. \hspace{2em} Compute the sequence of edges, \( \{v_{i+1}, \ldots, v_{j-1}v_j\} \), weakly visible from \( r \)
6. \hspace{2em} Replace the sequence \( \{v_{i+1}, \ldots, v_{j-1}\} \) by \( r \)

The sequence of edges weakly visible from \( r \), \( \{v_{i}v_{i+1}, \ldots, v_{j-1}v_j\} \), is a consecutive chain
Incremental Convex Hull

- The edge $pq$ is visible from $r$ \iff orientation(p, q, r) < 0
- The edge $pq$ is weakly visible from $r$ \iff orientation(p, q, r) \leq 0

Maintain the current convex hull $S$ of a set of points seen so far

1. Initialize $S$ to the counter-clockwise sequence $\{a, b, c\} \subset P$
2. Remove $a$, $b$, and $c$ from $P$
3. for all $r \in P$ do
   4. if there is an edge $e$ visible from $r$ then
   5. Compute the sequence of edges, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, weakly visible from $r$
   6. Replace the sequence $\{v_{i+1}, \ldots, v_{j-1}\}$ by $r$

- The sequence of edges weakly visible from $r$, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, is a consecutive chain
Incremental Convex Hull

- The edge $pq$ is visible from $r$ if
  $\iff \text{orientation}(p, q, r) < 0$
- The edge $pq$ is weakly visible from $r$ if
  $\iff \text{orientation}(p, q, r) \leq 0$

Maintain the current convex hull $S$ of a set of points seen so far

1. Initialize $S$ to the counter-clockwise sequence $\{a, b, c\} \subset P$
2. Remove $a, b, c$ from $P$
3. for all $r \in P$ do
4.     if there is an edge $e$ visible from $r$ then
5.     Compute the sequence of edges, $\{v_iv_{i+1}, \ldots, v_{j-1}v_j\}$, weakly visible from $r$
6.     Replace the sequence $\{v_{i+1}, \ldots, v_{j-1}\}$ by $r$

- The sequence of edges weakly visible from $r$, $\{v_iv_{i+1}, \ldots, v_{j-1}v_j\}$, is a consecutive chain
Convex Hull
(E. Fogel)

Incremental Convex Hull

- The edge $pq$ is visible from $r$ if $\text{orientation}(p, q, r) < 0$
- The edge $pq$ is weakly visible from $r$ if $\text{orientation}(p, q, r) \leq 0$

Maintain the current convex hull $S$ of a set of points seen so far

1. Initialize $S$ to the counter-clockwise sequence $\{a, b, c\} \subset P$
2. Remove $a$, $b$, and $c$ from $P$
3. for all $r \in P$ do
   4. if there is an edge $e$ visible from $r$ then
      5. Compute the sequence of edges, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, weakly visible from $r$
      6. Replace the sequence $\{v_{i+1}, \ldots, v_{j-1}\}$ by $r$

- The sequence of edges weakly visible from $r$, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, is a consecutive chain
Incremental Convex Hull

- The edge $pq$ is visible from $r$ if 
  \[ \text{orientation}(p, q, r) < 0 \]
- The edge $pq$ is weakly visible from $r$ if 
  \[ \text{orientation}(p, q, r) \leq 0 \]

Maintain the current convex hull $S$ of a set of points seen so far

1. Initialize $S$ to the counter-clockwise sequence $\{a, b, c\} \subset P$
2. Remove $a$, $b$, and $c$ from $P$
3. for all $r \in P$ do
   4. if there is an edge $e$ visible from $r$ then
   5. Compute the sequence of edges, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, weakly visible from $r$
   6. Replace the sequence $\{v_{i+1}, \ldots, v_{j-1}\}$ by $r$

- The sequence of edges weakly visible from $r$, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, is a consecutive chain
Incremental Convex Hull

- The edge $pq$ is visible from $r$ if $\text{orientation}(p, q, r) < 0$
- The edge $pq$ is weakly visible from $r$ if $\text{orientation}(p, q, r) \leq 0$

Maintain the current convex hull $S$ of a set of points seen so far:

1. Initialize $S$ to the counter-clockwise sequence \{a, b, c\} $\subset P$
2. Remove a, b, and c from $P$
3. for all $r \in P$ do
   4. if there is an edge $e$ visible from $r$ then
   5. Compute the sequence of edges, \{vi vi+1, ..., vj−1vj\}, weakly visible from $r$
   6. Replace the sequence \{vi+1, ..., vj−1\} by $r$

- The sequence of edges weakly visible from $r$, \{vi vi+1, ..., vj−1vj\}, is a consecutive chain
Incremental Convex Hull

- The edge $pq$ is visible from $r$ if $\text{orientation}(p, q, r) < 0$.
- The edge $pq$ is weakly visible from $r$ if $\text{orientation}(p, q, r) \leq 0$.

Maintain the current convex hull $S$ of a set of points seen so far:

1. Initialize $S$ to the counter-clockwise sequence $\{a, b, c\} \subset P$.
2. Remove $a$, $b$, and $c$ from $P$.
3. For all $r \in P$ do:
   4. If there is an edge $e$ visible from $r$ then:
   5. Compute the sequence of edges, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, weakly visible from $r$.
   6. Replace the sequence $\{v_{i+1}, \ldots, v_{j-1}\}$ by $r$.

- The sequence of edges weakly visible from $r$, $\{v_i v_{i+1}, \ldots, v_{j-1} v_j\}$, is a consecutive chain.
Wrong Incremental Convex Hull

\( p_1 = (24.00000000000005, 24.000000000000053) \)
\( p_2 = (24.0, 6.0) \)
\( p_3 = (54.85, 6.0) \)
\( p_4 = (54.85000000000357, 61.00000000000121) \)
\( p_5 = (24.000000000000068, 24.000000000000071) \)

- \( (p_1, p_2, p_3, p_4) \) form a convex quadrilateral.
- \( p_5 \) is truly inside this quadrilateral.
- \( \text{orientation}^*(p_4, p_1, p_5) < 0. \)

[KMP+08] Computational Geometry Algorithm Library 63
Wrong Incremental Convex Hull

\[ p_1 = (24.00000000000005, 24.000000000000053) \]
\[ p_2 = (24.0, 6.0) \]
\[ p_3 = (54.85, 6.0) \]
\[ p_4 = (54.85000000000357, 61.00000000000121) \]
\[ p_5 = (24.0000000000068, 24.0000000000071) \]
\[ p_6 = (6, 6) \]

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^*(p_4, p_1, p_5) < 0.\)

\[ [\text{KMP}^+08] \]
Wrong Incremental Convex Hull

\( p_1 = (24.00000000000005, 24.000000000000053) \)
\( p_2 = (24.0, 6.0) \)
\( p_3 = (54.85, 6.0) \)
\( p_4 = (54.8500000000000357, 61.000000000000121) \)
\( p_5 = (24.000000000000068, 24.000000000000071) \)
\( p_6 = (6, 6) \)

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^*(p_4, p_1, p_5) < 0.\)
- \(\text{orientation}^*(p_4, p_5, p_6) < 0.\)
- \(\text{orientation}^*(p_5, p_1, p_6) > 0.\)
- \(\text{orientation}^*(p_1, p_2, p_6) < 0.\)

[KMP+08]
Wrong Incremental Convex Hull

\( p_1 = (24.00000000000005, 24.000000000000053) \)
\( p_2 = (24.0, 6.0) \)
\( p_3 = (54.85, 6.0) \)
\( p_4 = (54.850000000000037, 61.0000000000000121) \)
\( p_5 = (24.000000000000068, 24.000000000000071) \)
\( p_6 = (6, 6) \)

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^*(p_4, p_1, p_5) < 0.\)
- \(\text{orientation}^*(p_4, p_5, p_6) < 0.\)
- \(\text{orientation}^*(p_5, p_1, p_6) > 0.\)
- \(\text{orientation}^*(p_1, p_2, p_6) < 0.\)

[KMP+08] Computational Geometry Algorithm Library 66
Wrong Incremental Convex Hull

\[
\begin{align*}
  p_1 &= (24.00000000000005, 24.000000000000053) \\
  p_2 &= (24.0, 6.0) \\
  p_3 &= (54.85, 6.0) \\
  p_4 &= (54.850000000000357, 61.00000000000121) \\
  p_5 &= (24.00000000000068, 24.00000000000071) \\
  p_6 &= (6, 6)
\end{align*}
\]

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
  - \(\text{orientation}^*(p_4, p_1, p_5) < 0\).
  - \(\text{orientation}^*(p_4, p_5, p_6) < 0\).
  - \(\text{orientation}^*(p_5, p_1, p_6) > 0\).
  - \(\text{orientation}^*(p_1, p_2, p_6) < 0\).

\[\text{[KMP}^+08\]\]
Wrong Incremental Convex Hull

\[ p_1 = (24.00000000000005, 24.000000000000053) \]
\[ p_2 = (24.0, 6.0) \]
\[ p_3 = (54.85, 6.0) \]
\[ p_4 = (54.85000000000357, 61.00000000000121) \]
\[ p_5 = (24.000000000000068, 24.000000000000071) \]
\[ p_6 = (6, 6) \]

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^\ast(p_4, p_1, p_5) < 0.\)
- \(\text{orientation}^\ast(p_4, p_5, p_6) < 0.\)
- \(\text{orientation}^\ast(p_5, p_1, p_6) > 0.\)
- \(\text{orientation}^\ast(p_1, p_2, p_6) < 0.\)

[KMP+08] Computational Geometry Algorithm Library 68
Wrong Incremental Convex Hull

\[ p_1 = (24.00000000000005, 24.000000000000053) \]
\[ p_2 = (24.0, 6.0) \]
\[ p_3 = (54.85, 6.0) \]
\[ p_4 = (54.850000000000357, 61.0000000000000121) \]
\[ p_5 = (24.000000000000068, 24.000000000000071) \]
\[ p_6 = (6, 6) \]

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^*(p_4, p_1, p_5) < 0.\)
- \(\text{orientation}^*(p_4, p_5, p_6) < 0.\)
- \(\text{orientation}^*(p_5, p_1, p_6) > 0.\)
- \(\text{orientation}^*(p_1, p_2, p_6) < 0.\)
Wrong Incremental Convex Hull

\( p_1 = (24.00000000000005, 24.000000000000053) \)
\( p_2 = (24.0, 6.0) \)
\( p_3 = (54.85, 6.0) \)
\( p_4 = (54.850000000000037, 61.000000000000121) \)
\( p_5 = (24.000000000000068, 24.000000000000071) \)
\( p_6 = (6, 6) \)

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- \(\text{orientation}^\star(p_4, p_1, p_5) < 0.\)
- \(\text{orientation}^\star(p_4, p_5, p_6) < 0.\)
- \(\text{orientation}^\star(p_5, p_1, p_6) > 0.\)
- \(\text{orientation}^\star(p_1, p_2, p_6) < 0.\)
Wrong Incremental Convex Hull

\[ p_1 = (24.00000000000005, 24.000000000000053) \]
\[ p_2 = (24.0, 6.0) \]
\[ p_3 = (54.85, 6.0) \]
\[ p_4 = (54.850000000000357, 61.000000000000121) \]
\[ p_5 = (24.000000000000068, 24.000000000000071) \]
\[ p_6 = (6, 6) \]

- \((p_1, p_2, p_3, p_4)\) form a convex quadrilateral.
- \(p_5\) is truly inside this quadrilateral.
- orientation\((p_1, p_2, p_6)\) < 0.
- orientation\((p_4, p_5, p_6)\) < 0.
- orientation\((p_5, p_1, p_6)\) > 0.
- orientation\((p_1, p_2, p_6)\) < 0.
Graham’s Scan Convex Hull

Andrew’s variant of Graham’s scan algorithm

---

Compute the convex hull \( S \) of a set of points \( P \)

Sort the points in lexicographic order, resulting in the sequence \( p_1, \ldots, p_n \)

\[
S_{\text{upper}} \leftarrow \{p_1, p_2\}
\]

for \( i \leftarrow 3 \) to \( n \)

while \( |S_{\text{upper}}| > 2 \) & !rightturn\((q, r, p_i)\), \( q \) and \( r \) are the last points of \( S_{\text{upper}} \)

\[
S_{\text{upper}} \leftarrow S_{\text{upper}} \setminus \{r\}
\]

\[
S_{\text{upper}} \leftarrow S_{\text{upper}} \cup \{p_i\}
\]

---
Andrew’s variant of Graham’s scan algorithm

---

Compute the convex hull $S$ of a set of points $P$

Sort the points in lexicographic order, resulting in the sequence $p_1, \ldots, p_n$

$S_{\text{upper}} \leftarrow \{p_1, p_2\}$

for $i \leftarrow 3$ to $n$

    while $|S_{\text{upper}}| > 2$ & !rightturn($q, r, p_i$), $q$ and $r$ are the last points of $S_{\text{upper}}$
        $S_{\text{upper}} \leftarrow S_{\text{upper}} \setminus \{r\}$
        $S_{\text{upper}} \leftarrow S_{\text{upper}} \cup \{p_i\}$

$S_{\text{lower}} \leftarrow \{p_n, p_{n-1}\}$

for $i \leftarrow n-2$ downto 1

    while $|S_{\text{lower}}| > 2$ & !rightturn($q, r, p_i$), $q$ and $r$ are the last points of $S_{\text{lower}}$
        $S_{\text{lower}} \leftarrow S_{\text{lower}} \setminus \{r\}$
        $S_{\text{lower}} \leftarrow S_{\text{lower}} \cup \{p_i\}$

Remove the last points from $S_{\text{lower}}$ and $S_{\text{upper}}$

$S \leftarrow S_{\text{upper}} \cup S_{\text{lower}}$
The convex hull of a set of \( n \) points in the plane can be computed in \( O(n \log n) \) time.
Theorem (Convex Hull)

The convex hull of a set of $n$ points in the plane can be computed in $O(n \log n)$ time.
Graham’s Scan Convex Hull

Theorem (Convex Hull)

The convex hull of a set of $n$ points in the plane can be computed in $O(n \log n)$ time.
The convex hull of a set of $n$ points in the plane can be computed in $O(n \log n)$ time.
Graham’s Scan Convex Hull

Theorem (Convex Hull)

The convex hull of a set of n points in the plane can be computed in $O(n \log n)$ time.
Graham’s Scan Convex Hull

Theorem (Convex Hull)

The convex hull of a set of \( n \) points in the plane can be computed in \( O(n \log n) \) time.
The convex hull of a set of \( n \) points in the plane can be computed in \( O(n \log n) \) time.
The convex hull of a set of $n$ points in the plane can be computed in $O(n \log n)$ time.
Convex Hull

Graham’s Scan Convex Hull

Theorem (Convex Hull)

The convex hull of a set of n points in the plane can be computed in \(O(n \log n)\) time.

FU Berlin, Lecture 2 (CGAL), WS 2012/13
Graham’s Scan Convex Hull

**Theorem (Convex Hull)**

The convex hull of a set of $n$ points in the plane can be computed in $O(n \log n)$ time.
CGAL Convex-Hull Implementations

- Given $n$ points and $h$ extreme points
  - CGAL::ch_akl_toussaint() $O(n \log n)$
  - CGAL::ch_bykat() $O(nh)$
  - CGAL::ch_eddy() $O(nh)$
  - CGAL::ch_graham_andrew() $O(n \log n)$
  - CGAL::ch_jarvis() $O(nh)$
  - CGAL::ch_melkman() $O(n)$ (simple polygon)
Upper Convex Hull

```cpp
#include <iostream>
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;

int main()
{
    Kernel kernel;
    std::vector<Kernel::Point_2> in;
    std::vector<Kernel::Point_2> out;
    upper_hull(in.begin(), in.end(), std::back_inserter(out), kernel);
    return 0;
}
```

- Using Random Access Iterator.  
  [upper_convex_hull_1.cpp]
  - Provides both increment and decrement.
  - Constant-time methods for moving forward and backward in arbitrary-sized steps ('[ ]' operator).
- Maintaining the current point separately.
  [upper_convex_hull_2.cpp]
- Maintaining an iterator that points to the current point separately.
  [upper_convex_hull_3.cpp]
- More generic — can be used for computing the lower hull.
Convex-Hull Bibliography

C. Bradford Barber, David P. Dobkin, and Hannu T. Huhdanpaa
The Quickhull algorithm for convex hulls.

Timothy M. Chan
Optimal output-sensitive convex hull algorithms in two and three dimensions.

Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee K. Yap.
Classroom Examples of Robustness Problems in Geometric Computations.
Convex Hull

Numerics

Kernels
Exercise 2

1.