Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
кафедра общей математики

Дипломная работа

Быстрый алгоритм построение тени набора многогранных поверхностей на прямоугольный участок плоскости.

Выполнил: М.М.Казбеков 509 гр.
Научный руководитель: к.ф.-м.н., доцент В.В. Сазонов

Москва 2013

Введение.

Проблемы решаемые этим алгоритмом могут быть использованы для планирования энергопотребления искуственными спутниками Земли. Математическая модель прогноза выработки электроэнергии солнечными батареями (далее СБ) малого лабароторного модуля (далее МЛМ) представляет собой прикладное программно-математическое обеспечение, предназначенное для оценки поступающей электроэнергии от солнечных батарей МЛМ Российского Сегмента (РС) Международной Космической Станций (МКС) на заданном временном интервале с учетом движения МКС по орбите, движения МКС относительно центра масс, положения подвижных элементов и состава МКС на момент прогнозирования.

Это программное обеспечение состоит из набора следующих модулей:

1. Геометрической модели МКС.

2. Отыскания освещенных участков поверхности солнечных батарей (СБ) МЛМ РС МКС.

3. Упрощенного математического моделирования работы СБ МЛМ РС МКС.

Далее приводится описание работы каждого из пяти указанных модулей.

Модуль геометрической модели МКС позволяет создавать трехмерную геометрическую модель МКС, состоящую из набора многогранников с треугольными гранями. Трехмерная модель используется для визуализации и для определения освещенных участков поверхности СБ МЛМ.

При отыскании освещенных участков СБ МЛМ используются следующие допущения:

1. Учитывается только прямое освещение от Солнца.

2. Солнце расположено бесконечно далеко от Земли и освещение от Солнца можно считать плоскопараллельным световым потоком.

Модуль определения освещенных участков СБ МЛМ использует геометрическую модель МКС для своей работы. От Солнца части СБ могут заслонять элементы конструкции станции. Освещенные части СБ получаются дополнением к частям СБ, затеняемых элементами конструкции станции. Таким образом, для того, чтобы отыскать освещенные участки СБ нужно найти тень от

части конструкции, которая находится между плоскостью СБ и Солнцем на плоскость обрабатываемой СБ, а потом произвести отсечение найденной тени по границе СБ.

Одной из геометрических задач, решаемых этим модулем, является задача построения геометрической тени объекта при освещении параллельным световым потоком. Для работы алгоритма построения геометрической тени необходимо задать следующие данные:

- теневую плоскость – плоскость, на которую надо построить тень;
- направление распространения света;
- трехмерную геометрическую модель объекта.

Объект задается набором многогранников с треугольными гранями. Заданные многогранники могут пересекаться с плоскостью, на которую отбрасывается тень.

При фиксированном положении источника параллельного света можно грани объекта разделить на два класса:

- лицевые – грани, нормаль которым составляет острый угол с направлением распространения на света;
- нелицевые – грани, нормаль которым составляет тупой угол с направлением распространения света.

При такой классификации граней ребра можно разделить следующим образом:

- лицевые – соединяют лицевые две грани,
- нелицевые – соединяют две нелицевые грани,
- контурные – соединяют лицевую и нелицевую грани.

Контурные ребра образуют так называемые контурные циклы, проекции этих циклов на теневую плоскость образуют границы геометрической тени. Каждое элементарное тело, из которых состоит модель МКС, имеет свой набор контурных циклов. Общая тень получается объединением всех многоугольников – проекций контурных циклов на теневую плоскость. При построении геометрической тени все объекты отсекаются теневой плоскостью и берутся те часть объектов, которые

лежат в одной полуплоскость с источником света (Солнцем).

СБ МЛМ представляет собой набор прямоугольных панелей – солнечных элементов. Геометрическая модель СБ – набор плоских прямоугольников.

Для отыскания затененной части СБ надо:

1. Построить геометрическую тень МКС на плоскость обрабатываемой СБ.

2. Произвести отсечение геометрической тени по набору прямоугольных областей, моделирующих солнечные элементы СБ.

Для поиска объединений и пересечений многоугольников на плоскости используется линейно-узловой алгоритм на основе алгоритма, предложенного А.В. Скворцовым [5].

Далее площадь освещенной части солнечного элемента СБ получается вычитанием площади затененной части и общей площади данного солнечного элемента.

Математическое моделирование работы солнечных элементов является задачей вычислительно сложной и громоздкой для реализации. В общем случае требуется учитывать множество факторов: прямое солнечное излучение, отраженный солнечный свет от поверхности станции, от атмосферы Земли, от Луны, текущую электрическую нагрузку цепи, в которую входит рассматриваемый солнечный элемент [6].

На этапе создания прототипа предполагается использовать упрощенную модель выработки электроэнергии солнечными элементами, входящими в состав СБ МЛМ РС МКС.

Считаем, что мощность, вырабатываемой солнечной энергии одним элементом СБ может быть вычислена по формуле:

$P = kS(n, l)$, где k – некоторый коэффициент, S – площадь освещенной части элемента СБ, n – внешняя нормаль к поверхности элемента СБ, l – направление на Солнце.

Коэффициент k не является постоянным, так как со временем характеристики работы СБ изменяются, происходит так называемая деградация. Но изменение

происходит медленно, потому можно считать эту величину константой при построении прогноза выработки электроэнергии СБ МЛМ МКС на несколько суток. Этот коэффициент может быть определен на основе экспериментов и уточняться по данным телеметрической информации, поступающей с борта.

Далее выработку электроэнергии за заданный прогнозный период можно вычислить следующим образом:

$W = \sum \int k_i S_i(t)(n_i(t), l)dt$, где N – количество солнечных элементов в составе СБ, $[t_{0,} t_1]$ – интервал моделирования, k_i – коэффициент i-го солнечного элемента, S_i – площадь освещенной части i-го солнечного элемента, n_i – внешняя нормаль поверхности i-го солнечного элемента.

Постановка задачи.

Рассматривается объект заданный набором многогранных поверхностей с треугольными гранями. Поверхности без самопересечений и участков самоналегания. Каждой грани поверхности объекта соответствует внешная нормаль. Выбирая внешний вектор нормали, мы задаем ориентацию поверхности. Ребро грани ориентировано так, что при движении вдоль ребра рассматриваемая грань располагается слева, если наблюдатель смотрит из конца вектора нормали. Требуется найти тень многогранной поверхности на прямоугольный участок плоскости при освещении плоско-параллельным световым потоком.

Используемые определения.

Будем называть *грань лицевой*, если её нормаль образует острый угол с направлением на источник света, и *не лицевой* в противном случае. Ребро будем называть *лицевым*, если оно принадлежит двум лицевым граням, *не лицевым* если оно принадлежит двум нелицевым граням, и *граничным* если оно соединяет лицевую и нелицевую грани. Грани называются *смежными* если они имеют общее ребро.

Будем считать, что две лицевые грани *связаны*, если можно переместиться с одной на другую, двигаясь только по лицевым граням поверхности. Под *лицевой поверхностью* будем понимать множество лицевых граней, каждая из которых связана со всеми другими гранями этого множества.

Будем называть вершину поверхности *точкой сборки*, если при обходе всех инцидентных ребер с переходом от ребра к ребру вдоль поверхности по часовой стрелке, сумма всех углов между проекциями последовательных ребер больше

2π. Вершина называется *особой*, если ей инцидентно более двух граничных ребер или если она является точкой сборки.

Способ решения задачи.

На область тени в плоскости влияет только освещенная часть поверхности. Так как если выбросим все неосвещенные участки объекта, от этого тень его не изменится. Поэтому, тень это проекция на плоскость освещенных участков поверхности. Так как проекция освещенных участков поверхности и объединение проекции лицевых поверхностей совпадают, то эту задачу можно решить следующим способом:

Предлагаемый алгоритм состоит из следующих важных этапов:

1) найти все граничные ребра
2) найти все лицевые поверхности, контурные циклы
3) найти проекции всех контурных циклов на участок плоскости, ограниченный прямоугольником.
4) отыскание тени
6) отсечение их с заданным прямоугольником.

Рассматриваемый объект содержит одну или несколько лицевых поверхностей. Граница лицевой поверхности состоит из граничных ребер, которые образуют цикл, который мы будем называть *контурным циклом*.

Ориентация каждого контурного цикла определяется ориентацией поверхности. Количество контурных циклов, ограничивающих лицевую поверхность, зависит от формы поверхности, его положения относительно источника света.

Внизу приведено различные взаимные расположения проекций контурных циклов. Каждому варианту взаимного расположения лицевых поверхностей соответствует вариант взаимного расположения их границ *(рис. 1)*.

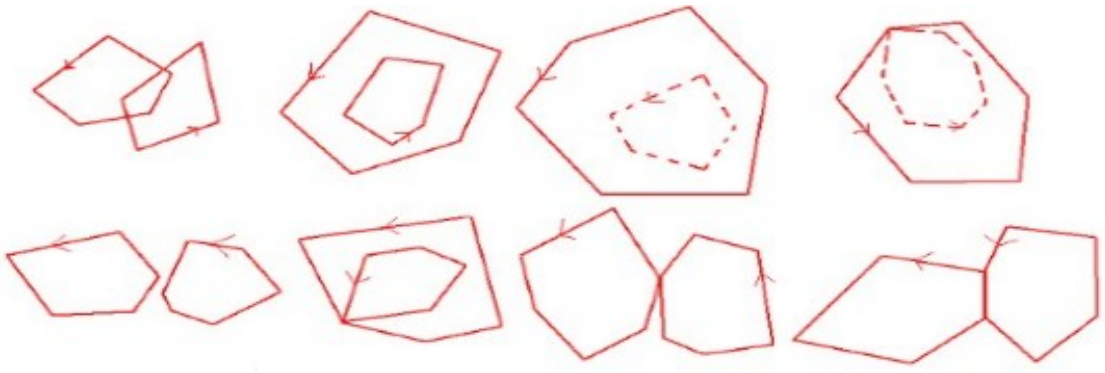


Рис. 1. Взаимное расположение контурных циклов

Нелицевые грани не освещены, поэтому из рассмотрения их следует

исключить. В основе предлагаемого алгоритма лежит алгоритм *удаления невидимых линий Аппеля* [1].

Организация входных данных.

Объект состоит из некоторого числа многогранных поверхностей, которые состоят из треугольных граней. Поверхность задается тремя массивами:

vertices — это массив вершин многогранника.

edges — это массив ребер многогранника.

faces — массив граней поверхности.

Каждая *вершина* задается тремя координатами, числами плавающей точкой. И каждой вершине однозначно сопоставлено натуральное число – индекс в массиве.

Каждая *грань* содержит:

Вектор нормали

3 индекса вершин

3 индекса ребер

Каждое *ребро* является ориентированным и представляет собой запись из шести полей.

Номера начальной и конечной вершины

Номера граней содержащих данное ребро, которые остаются слева и справа при движении вдоль ребра.

Номера следующего ребра при обходе вершины по часовой стрелке, для начальной и конечной вершины.

Расположения «*справа*» и «*слева*» находятся исходя из того, что грани наблюдаются из источника света.

Входные данные построены на основе *реберного списка с двойными связями (РСДС)* и стандартного представления трехмерных данных — *VRML*. Организация данных позволяет выполнить за время *O(1)* следующие операции:

найти следующее ребро за заданным при обходе по часовой стрелке

найти смежную грань с данной гранью по заданному ребру

Для создания структуры *РСДС* требуется число операции *O(n)* , где *n* — количество граней объекта *(рис. 2)*.

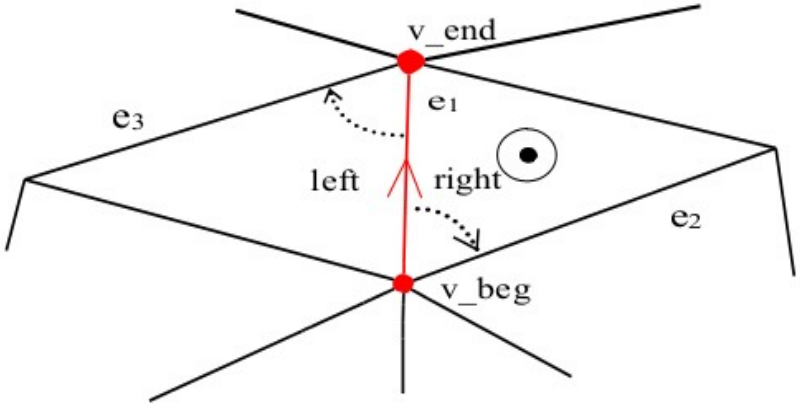


Рис. 2. Реберный список с двумя связями

Основные этапы алгоритма .

Нахождение лицевых граней, контурных циклов и их проекции на заданную плоскость . Нахождение теневых участков плоскости.

Результатом работы алгоритма является набор контуров, ограничивающих теневую область. После отыскания контуров можно провести триангуляцию теневой области и получить набор треугольников.

Нахождение лицевых граней и контурных циклов.

Для нахождения лицевых поверхностей надо выделить связанные множества лицевых граней. Граница лицевой поверхности будет состоять из граничных ребер, которые объединяются в контурные циклы.

Алгоритм:

1. Помечаем все грани как необработанные.

2. Двигаемся по массиву граней, пока не встретим необработанную лицевую грань и делаем её текущей и помечаем как обработанную. Если достигнут конец массива, *останавливаемся*.

3. Помечаем все ребра как не контурные.

4. Просматриваем все грани, которые непосредственно связаны с текущей гранью и необработаны. Если просматриваемая грань является лицевой, то записываем её номер в стек, если нет, то общее ребро текущей и просматриваемой грани помечаем как контурное.

5. Если стек не пуст, то достаем из стека номер грани, делаем эту грань текущей и переходим к *шагу 4*. Если стек пуст — переходим к *следующему*

шагу.

6. Двигаемся по массиву ребер, пока не встретим контурное ребро. Если ребро найдено — делаем его текущим и переходим к следующему шагу. Если достигнут конец массива, то считаем текущую лицевую поверхность сформированным и переходим к *шагу 2*.

7. Добавляем текущее ребро в цикл и ориентируем его таким образом чтобы лицевая грань оставалось слева. Делаем конец ребра текущей вершиной, а начало ребра — начальной вершиной. Помечаем текущее ребро как неконтурное.

8. Находим следующее ребро за текущим, определяем угол поворота против часовой стрелке от проекции текущего ребра до проекции следующего и записываем полученный результат в переменную «*alpha*». Делаем следующее ребро текущим.

9. Если текущее ребро — контурное, то переходим к следующему шагу. В противном случае находим следующее ребро, считаем угол поворота против часовой стрелке от текущего ребра к следующему и прибавляем полученный результат к переменной «*alpha*».

10. Если значение *«alpha» > 360*, то помечаем текущую вершину, как точку сборки. Ориентируем ребро так, чтобы лицевая грань оставалось слева. Делаем конец ребра текущей вершиной. Если текущая вершина совпадает с начальной, то считаем цикл сформированным, добавляем его к циклам текущей освещенной поверхности и переходим к *шагу 6*.

Если текущая не совпадает с начальной, то переходим *шагу 8* [2].

Ниже приведен пример работы программы:

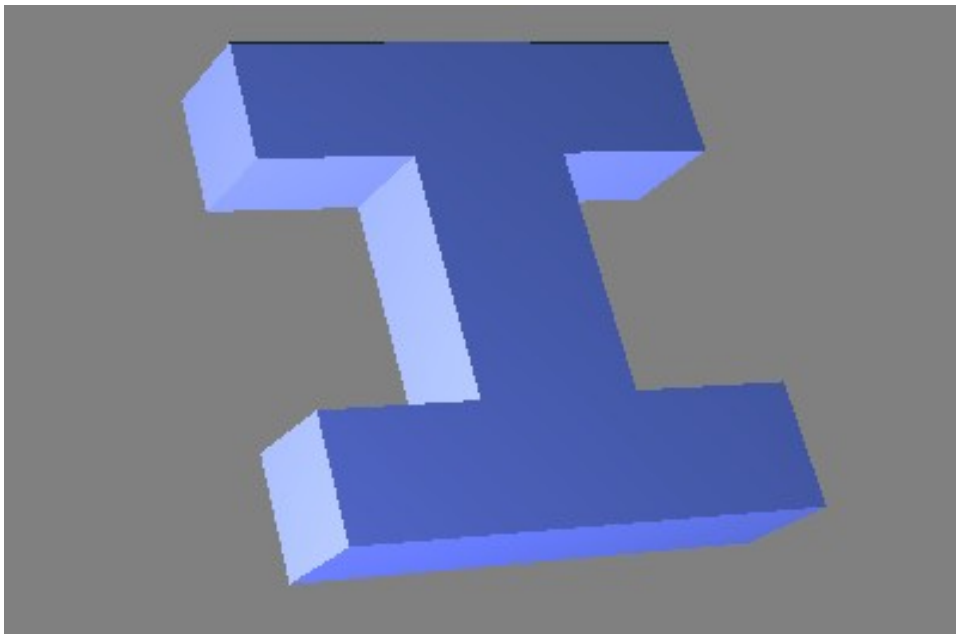


Рис. 3. Рассматриваемый объект

После первого этапа:
Построены лицевые поверхности и найдены контурные циклы, найдены вершины контурных ребер, которые являются точками сборки поверхности *(рис. 4).*

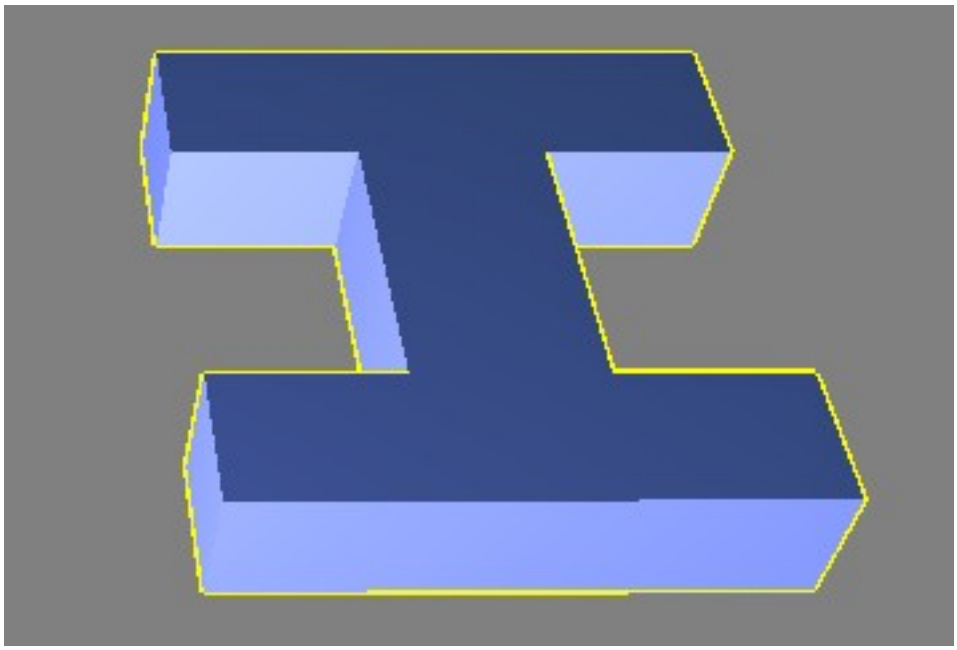


Рис. 4. Контурные циклы

Проектирование контурных циклов на плоскость.

Пусть задана плоскость проектирования с помощью нормали и точкой принадлежащей ей и задан направление светового потока. Последовательно проектируя все вершины контурного цикла, получаем его проекцию. Найдем проекцию произвольной точки на плоскость. Этого достаточно уметь для нахождения проекции контурного цикла.

Используемые формулы

1. $Ax+By+Cz+D=0$

2. $\vec{r}=\vec{r_0}+\vec{a}*t$

где $\vec{n}=(A, B, C)$ - нормаль плоскости проектирования

r_0 - точка, проекцию которого ищем

a - направление светового потока.

Известны координаты одной точки на плоскости, поставив её в формулу $Ax+By+Cz+D=0$ найдем *D*. Поставив $\vec{r}=\vec{r_0}+\vec{a}*t$ в $Ax+By+Cz+D=0$ найдем число t_0 .

$\vec{r}=\vec{r_0}+\vec{a}*t_0$ это есть координаты искомой точки *(рис. 5).*

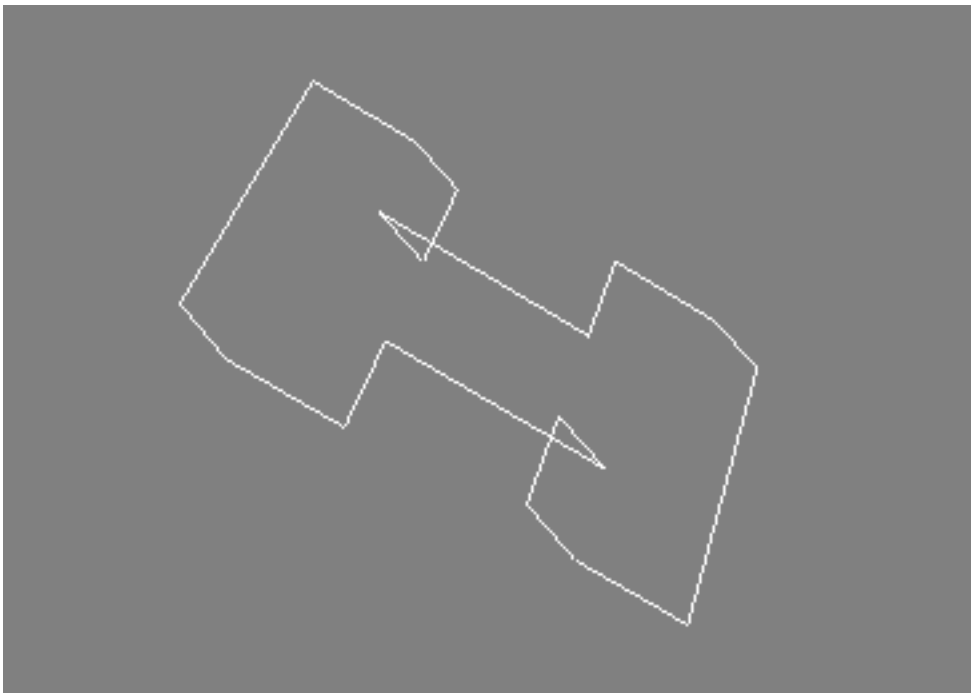


Рис. 5. Проекция контурных циклов на плоскость

Матрицу проецирования в эту плоскость можно представить в таком виде:

$$\boldsymbol{P}=\frac{1}{(n,a)}\begin{bmatrix}(n,a)-a_1A & -a_1B & -a_1C & -a_1D\\ -a_2A & (n,a)-a_2B & -a_2C & -a_2D\\ -a_3A & -a_3B & (n,a)-a_3C & -a_3D\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Обработка проекций контурных циклов. Построение тени.

Линейно-узловой алгоритм построения оверлеев полигонов.

После того как мы спроектировали контурные циклы в плоскость, остается удалить самопересечения, объединить полученные полигоны и отсечь их заданным прямоугольником.

Задача построения оверлеев двух полигонов (объединение, пересечение и разность) является одной из базовых задач вычислительной геометрии. В том или ином виде она встает в системах автоматизированного проектирования, геоинформационных системах, машинной графике.

Существуют многочисленные алгоритмы для решения данной задачи. Однако большинство из них обладает различными ограничениями. В первую очередь это связано с типом исходных данных. Наиболее легко решается случай оверлея двух выпуклых полигонов или когда хотя бы один из полигонов является выпуклым. При этом дополнительно предполагается отсутствие различных вырожденных случаев (отсутствие совпадений последовательных точек, ненулевая площадь полигонов и т.д.). Один из первых алгоритмов, работающих для невыпуклых и многоконтурных полигонов, был представлен Вейлером и Азертоном. К сожалению, в этом алгоритме приходится отслеживать многочисленные частные случаи, приводящие к существенному усложнению реализации.

Также есть универсальный алгоритм построения оверлеев на основе алгоритма построения триангуляции с ограничениями. Алгоритм обладает очень простой логикой, легок в реализации, но, к сожалению, работает значительно дольше других известных алгоритмов.

Здесь представлен также универсальный алгоритм, позволяющий работать с произвольными полигонами, в том числе многоконтурными и самопересекающимися, но значительно более быстрый, чем основанный на триангуляции [5].

Определения.

Границией полигона *G* будем называть множество контуров $\{K_1, K_2, \ldots, K_m\},\ m \geq 1$, каждый из которых является замкнутой ломаной линией, соединяющей набор точек на плоскости.

Полигоном P будем называть геометрическое место точек, лежащих на заданной границе *G* либо находящихся “внутри” нее. Здесь мы не будем вводить различные классы полигонов, поэтому будем использовать такое общее определение.

Задача построения оверлеев.

Пусть заданы два полигона *P1* и *P2* в виде границ *G1* и *G2*. Требуется найти границы полигонов *I, U, D*, определенных как соответствующие булевы операции над множеством всех точек исходных полигонов:

$$I = \overline{P1 \cap P2},\ U = \overline{P1 \cup P2},\ D = \overline{P1 - P2}$$

где знаком $\breve{P}$ обозначена операция размыкания (вычитание из множества его границы); $\overline{P}$ – замыкание множества. Использование операций замыкания и размыкания в постановке задачи позволяет избежать вырожденных ситуаций при построении оверлеев. Иначе, например, в случае нахождения пересечения двух касающихся полигонов может образоваться множество в виде линии или даже одной точки, которое нельзя представить в виде невырожденного многоугольника.

Основная идея предлагаемых алгоритмов заключается в предварительном построении линейно-узловой модели – геометрического планарного графа специального вида, ребра которого должны соответствовать отрезкам исходных границ полигонов. Затем после построения графа производится классификация ребер графа по признаку вхождения в результирующий полигон. И в конце выполняется сборка отрезков графа в последовательность отрезков границы требуемого полигона. Происхождение термина «линейно-узловой» связано с используемыми в геоинформатике похожими топологическими моделями данных – покрытиями, называемыми также линейно-узловыми моделями. Для решения ранее поставленных задач дадим некоторые определения и поставим некоторые более простые задачи.

Определение. *Линейно-узловой моделью M* будем называть геометрический граф $\{N, R\}$, где *N* – множество вершин графа – точек (x_i, y_i) на плоскости; R – множество непересекающихся ребер – отрезков $((x_j^1, y_j^1),(x_j^2, y_j^2))$, соединяющих вершины графа.

Задача построения линейно-узловой модели.

Пусть задано произвольное множество отрезков $((x_j^1, y_j^1),(x_j^2, y_j^2))$ на плоскости. Надо построить линейно-узловую модель, в которую войдут в качестве вершин все точки $(x_j^1, y_j^1),(x_j^2, y_j^2)$ и все точки пересечения исходных отрезков, а также в качестве ребер должно быть взято минимальное множество непересекающихся отрезков (возможно, касающихся концами), совпадающее как геометрическое место точек с исходным множеством отрезков.

По сути, требуется взять в качестве ребер все исходные отрезки, причем если эти отрезки пересекаются между собой, то они должны быть разбиты на части точками пересечений.

Данная задача легко решается в терминах классической геометрии. К сожалению, как правило, реальные компьютерные вычисления выполняются только с ограниченной точностью, что приводит в рамках данной задачи к интересным эффектам. Например, найденная точка пересечения отрезков в силу ограниченности точности вычислений оказывается немного в стороне от

реального вещественного значения. В итоге два исходных пересекающихся отрезка после разбиения на две части точкой пересечения оказываются лежащими не на прямых, проходящих через исходные отрезки. В итоге это может привести к пересечению этих новых отрезков с какими-то другими ранее непересекаемыми отрезками.

Таким образом, при разработке алгоритма построения линейно-узловой модели необходимо учесть возможность появления новых пересечений и при необходимости разбивать в том числе и вновь образуемые отрезки.

Алгоритм построения линейно-узловой модели.

Предполагается, что дано множество исходных отрезков $((x_j^1, y_j^1),(x_j^2, y_j^2))$.

Шаг 1. Формируем множество вершин $N=\{n_i\}$, по очереди добавляя в него вершины исходных отрезков; совпадающие вершины отбрасываем. Множество ребер делаем пустым: $R=\varnothing$.

Шаг 2. Заносим в список *L*, еще не обработанных отрезков все исходные отрезки в виде пар (n_j^1, n_j^2) .

Шаг 3. Пока список *L* не пуст, извлекаем из него самый длинный отрезок $(n^{1,} n^2)$ и пытаемся вставить его в текущую линейно-узловую модель R. Если такой отрезок уже вставлен в качестве ребра, то ничего не делаем. Если вставляемый отрезок не пересекает ни одно ребро из *R* и не проходит через вершины из *N*, кроме n_1 и n_2 , то вставляем его. В противном случае, если имеет место последовательное прохождение через вершины $\tilde{n_1}, \tilde{n_2}, \ldots, \tilde{n_m} \in N$, заносим в список *L* ребра проходящие через эти вершины $(n_1, \tilde{n_1}), (\tilde{n_1}, \tilde{n_2}), \ldots, (\tilde{n_m}, n_2)$. Иначе, если обнаружено последовательное пересечение с некоторыми ребрами $r_1, r_2 \ldots, r_m,\ r_i=(n_i^1, n_i^2)$ в точках $p^i=(x^i, y^i)$, удаляем эти ребра из *R* и заносим в список *L*, новые разбитые отрезки $(n^1, p^1),(p^1, p^2),\ \ldots (p^m, n^2) i=\overline{1,m}$ и $(n_i^1, p^i),(p^i, n_i^2),\ где\ i=\overline{1,m}$.

Конец алгоритма.

Построение топологии.

Теперь перейдем к вопросу применения линейно-узловой модели для построения оверлеев и упрощения полигонов.

Основная идея предлагаемых ниже алгоритмов построения оверлеев и упрощения полигонов заключается в передаче на вход алгоритма построения линейно-узловой модели всех отрезков исходных полигонов, последующего построения топологии, затем классификации ребер по некоторому критерию и, в

заключение, объединения классифицированных отрезков в границу результирующего полигонов.

Классификация ребер модели и конструирование полигонов .

Этап классификации ребер линейно-узловой модели при построении оверлеев предназначен для выделения тех ребер, которые должны будут войти в результирующий полигон. Рассмотрим его для разных типов оверлеев.

Вначале нужно для каждого ребра определить, по какую сторону от него находится первый и второй полигон. Далее же используем следующие логические условия в зависимости от типа операции.

1. Объединение полигонов. В результирующий полигон должны войти только два типа рёбер (рис. 6, а, б):

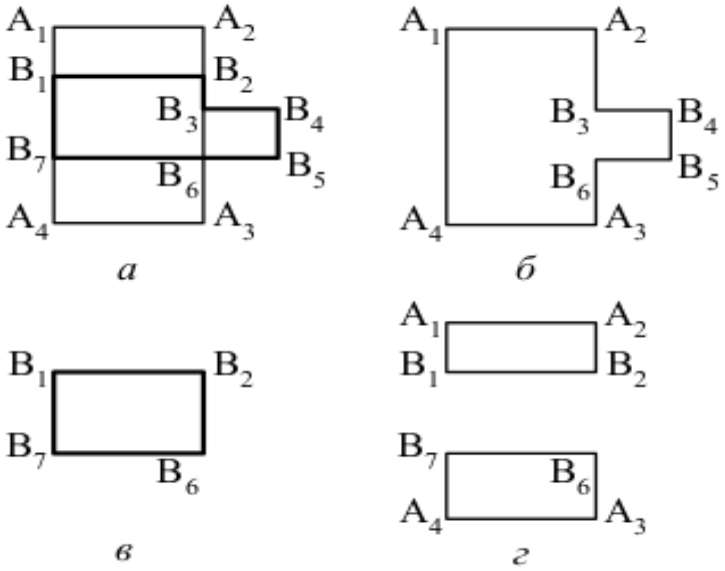
– если ребро используется в обоих полигонах и оба они лежат по одну сторону от ребра;

– если ребро используется только одним из полигонов и оно лежит вне другого.[5]

2. Пересечение полигонов. В результирующий полигон должны войти только два типа рёбер (рис. 6, а, в):

– если ребро используется в обоих полигонах и оба они лежат по одну сторону от ребра;

– если ребро используется только одним из полигонов и оно лежит внутри другого.



Таким образом, для классификации всех ребер требуется умение выполнять две операции:

1) определять расположение полигона относительно ребра линейно-узловой модели;

2) определять, попадает ли некоторое ребро на границу, внутрь или вне полигона.

Применение.

Предполагается что этот алгоритм будет использоваться для решения задачи определения выработки электроэнергии с батарей служебного модуля Российского сегмента *Международной Космической Станции (МКС)* во время полета

(рис11, 12)

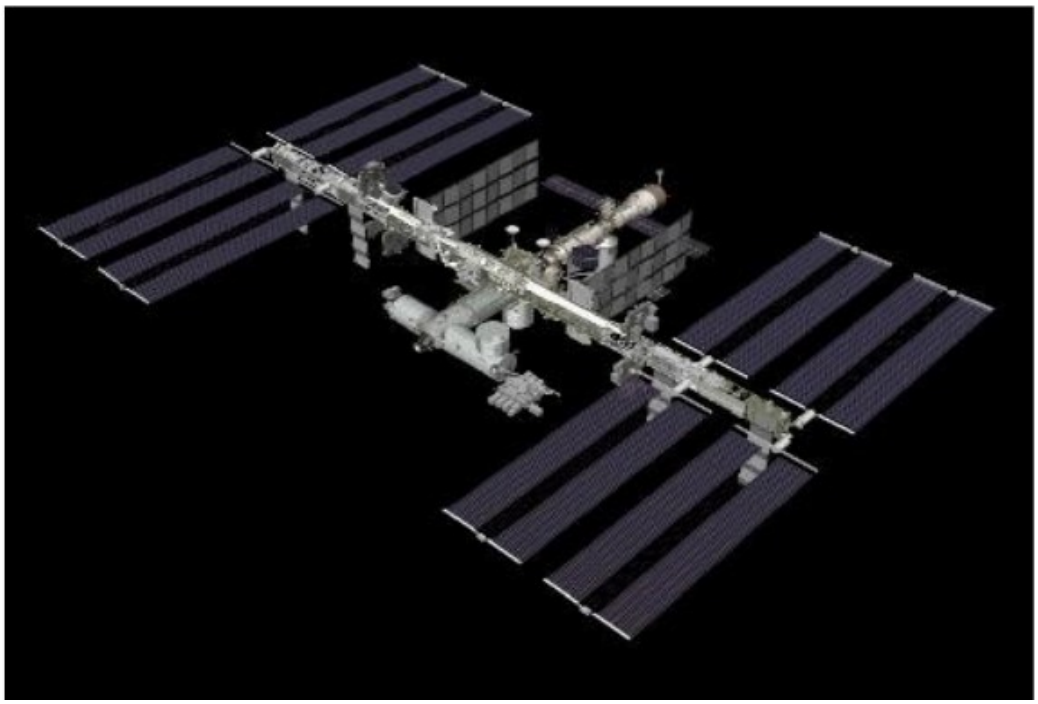


Рис. 11. Международная космическая станция

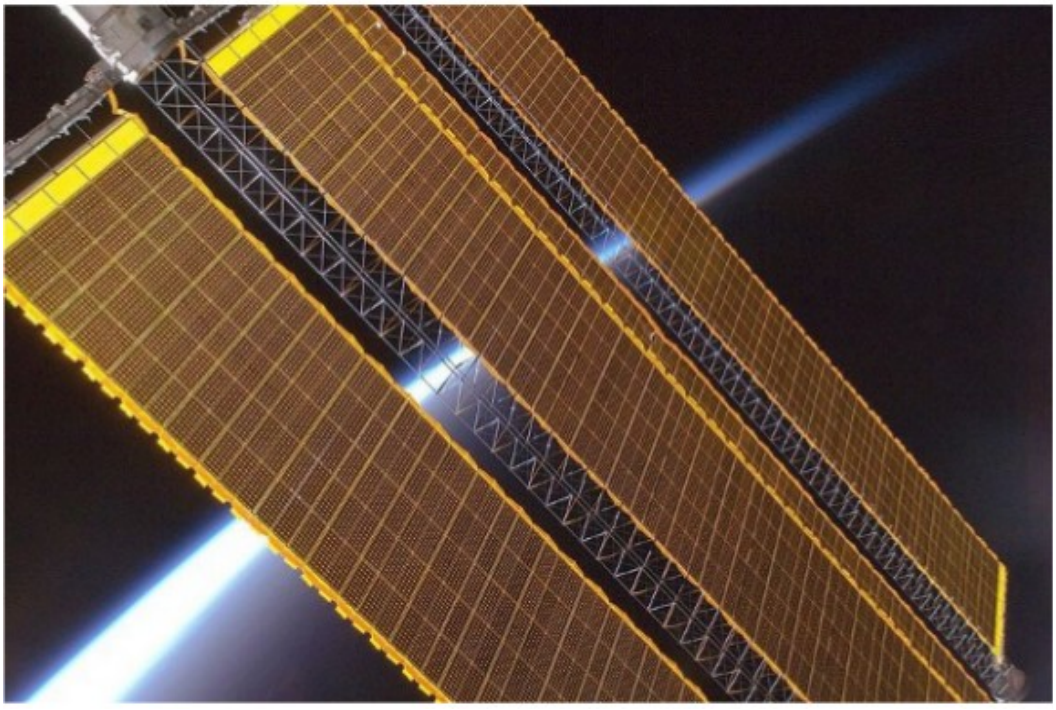


Рис. 12. Солнечная батарея на МК

Литература

1. *Appel A.* The Notion of Quantitative Invisibility and Machine Rendering of Solids, Proc. of ACM National
Conference, Thompson Book, 1967, p.387.

2. *В.В. Сазонов.* Алгоритм отыскания освещенных участков многогранных поверхностей в плоскопараллельном световом потоке // Математическое моделирование, 2007, июнь, т. 19, No6, с. 16-30.

3. *Е.В. Шикин, А.В. Боресков.* Компьютерная графика. Полигональные модели. – М.: ДИАЛОГ-МИФИ, 2000, 464 с.

4. *М.В. Леонов, А.Г. Никитин.* Эффективный алгоритм, реализующий замкнутый
набор булевых операций над множествами многоугольников на плоскости . Новосибирск, 1997, 25 с.

5. *А.В. Скорцов.* Линейно-узловой алгоритм построения оверлеев двух полигонов. Статья представлена кафедрой теоретических основ информатики факультета информатики Томского государственного университета, поступила в научную редакцию номера 3 декабря 2001 г.

6. *Г. Раушенбах.* Справочник по проектированию солнечных батарей. М. Энергоатомиздат. 1983г. 360 с.