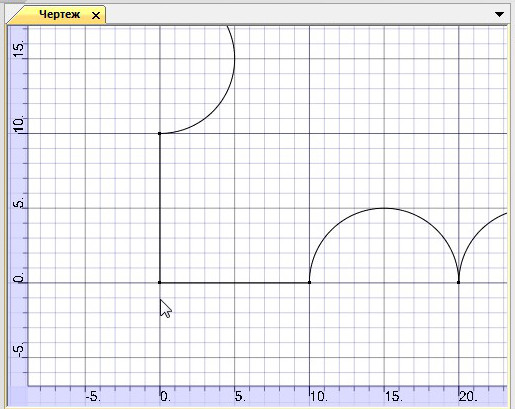
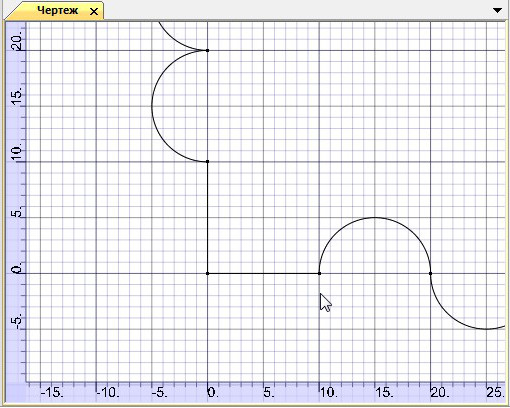
Описание модуля для создания дуговой фаски.

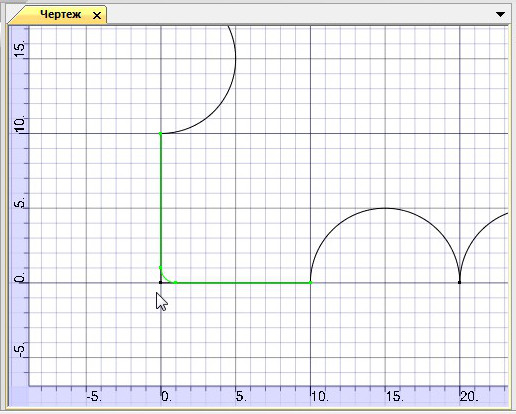
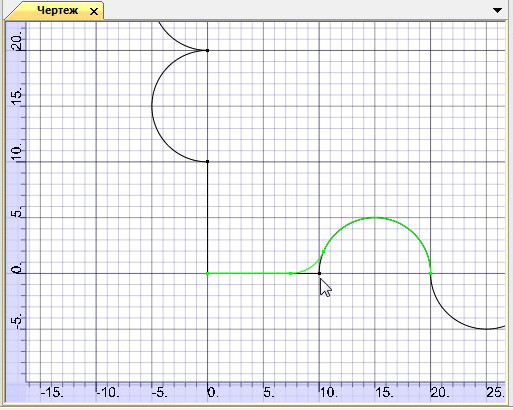
Целью написания данного модуля является возможность быстрого и эффективного создания дуговой фаски. Сам процесс для пользователя занимает 3 этапа:

1. Выбор инструмента «фаска» и точки (т.е. угла, который будет спилен):

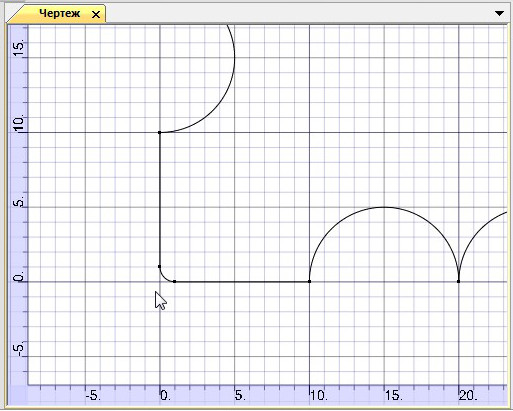
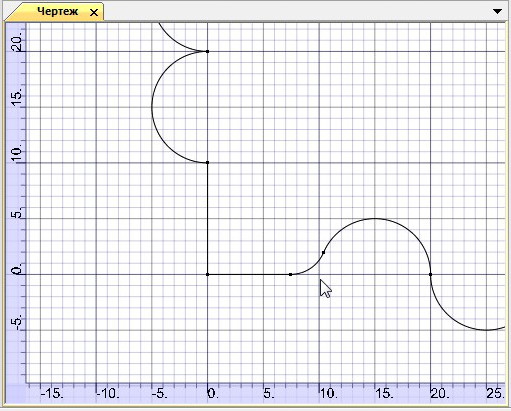
|  |  |
| --- | --- |
| Рис. 1а | Рис. 1б |

1. Наведение курсора и просмотр возможного результата (вспомогательного контура). На этом этапе пользователь ещё может изменить выбор точки и задать свой радиус фаски:

|  |  |
| --- | --- |
| Рис. 2а | Рис 2б |

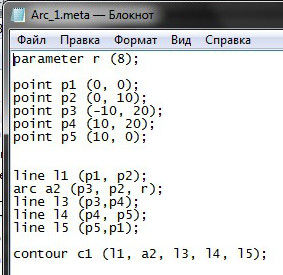
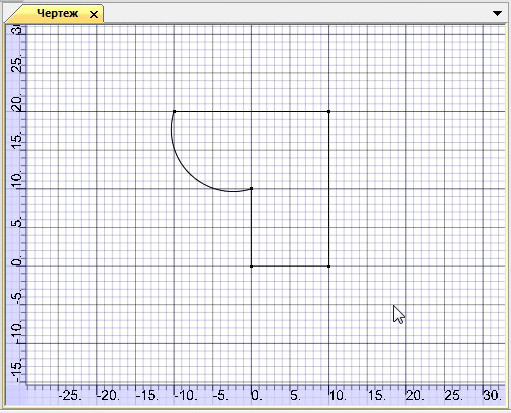
1. Фиксация результатов простым нажатием на левую кнопку мыши:

|  |  |
| --- | --- |
| Рис. 3а | Рис 3б |

Данный модуль работает с комбинациями двух типов кривых: отрезок и дуга.

Все параметрические чертежи хранятся в обычном текстовом файле с расширением .meta:

 превращается в 

Т.е. работа возможно только с точками, типами дуг и контурами.

Ниже приводится список файлов и функций с кратким описанием логики их работы.

Для первого этапа работы необходимо осуществить выбор инструмента «фаска» и фиксация выбранной пользователем точки. Это обеспечивают две функции:

**ChildView.cpp**

void CChildView::OnMouseMove(UINT nFlags, CPoint point).

При движении курсора над рабочей областью около узловой точки, если тип курсора – кисть, то записываются координаты курсора и вызывается функция SetTempFacet(&cursor) для создания временного вспомогательного контура (переход к этапу 2).

void CChildView::OnLButtonUp(UINT nFlags, CPoint point).

При клике курсора над рабочей областью около узловой точки, если тип курсора – кисть, то записываются координаты курсора и вызывается функция итогового построения дуговой фаски SetFacet(&cursor) (переход к этапу 3).

Для второго этапа работы необходимо создать вспомогательный контур. На Рис. 2а и 2б он отмечен зелёным цветом.

**I2DScetch.cpp**

void I2DSketch::SetTempFacet(Math::C2DPoint \*ppoint).

Вначале происходит проверка получаемой точки. Происходит поиск двух кривых из общего списка, имеющих данную точку как общую. Если такие есть, то фиксируются начальные и конечные точки этих кривых. Эти точки располагаются в правильном порядке (функцией SetTempFacetPointsOrder(…)) для создания вспомогательного контура. Вспомогательный контур состоит из двух кривых, являющихся копиями исходных: Outline.CreateCurve(*тип*, *узел1*, *узел2*);

Если среди кривых встретились дуги, то они уточняются указанием радиуса, ориентации и, если необходимо, заменой мест начальной и конечной точек. Также предусмотрены случаи, когда обе кривые образуют замкнутую область, т.е. начальные и конечные точки совпадают. Это возможно для двух случаев: «отрезок-дуга» и «дуга-дуга». Фиксируется идентификатор углового узла, строится временная фаска (t\_Outline.AddFacet(…)) и исходная угловая точка удаляется из вспомогательного контура.

void I2DSketch::SetFacet(Math::C2DPoint \*ppoint).

Аналогично SetTempFacet за исключением того, что вспомогательный контур не создаётся, а происходит уже итоговое построение фаски (AddFacet(…)).

Для третьего этапа работы необходимо определить тип получаемых кривых, возможность построения фаски с заданным радиусом и, если нет ошибок, то построить её.

**C2DOutline.cpp**

bool C2DOutline::AddFacet(…).

Функция получает два идентификатора кривых, идентификатор углового узла и желаемый радиус дуговой фаски. Фиксируются начальные и конечные точки у кривых. В зависимости от типа полученных кривых вызываются функции:

bool C2DOutline::AddFacet\_2lines(…), случай «отрезок-отрезок»

bool C2DOutline::AddFacet\_line\_arc(…), случай «отрезок-дуга»

bool C2DOutline::AddFacet\_2arcs(…). случай «дуга-дуга»

Если тип кривых не является ни отрезком, ни дугами, то возвращается false и фаску построить нельзя.

bool C2DOutline::AddFacet\_2lines(…) («отрезок-отрезок»).

Случай, когда исходные кривые являются отрезками. Создаются два вектора (vect1,vect2) с помощью функции AddFacet\_ugl(…), которые направлены из угловой (общей) точки к противоположным концам отрезков. Если сбоя не произошло, то на основе полученных векторов (vect1,vect2) создаются вектора (a,b) для параллельного перемещения вспомогательных отрезков функцией AddFacet\_scale\_rotate(…). Дальше по вектору «b» происходит создание узлов отрезка и самого отрезка, параллельного одному исходному. Этот случай проще остальных, т.к. сразу известно, где создавать вспомогательный отрезок. Функцией GetIntersection(…) происходит проверка, пересекаются ли только что созданный отрезок с тем, в основе которого лежит вектор «a». Если да, то точка их пересечения будет являться центром дуговой фаски. Если нет (возможно был задан не тот радиус фаски), то фаску построить нельзя. Следующим шагом находятся точки на исходных отрезках, в которых дуговая фаска будет крепиться. Теперь заново создаются отрезки, но с уже включённой между ними фаской с помощью функции AddFacet\_2lines\_create(…). И в конце удаляются исходные отрезки.

bool C2DOutline::AddFacet\_line\_arc(…) («отрезок-дуга»).

Случай, когда одна исходная кривая – отрезок, а вторая – дуга. Сначала происходит опознание отрезка и дуги. В переменную v1 помещается отрезок, в v2 – дуга. Если кривые расположены не так, то они меняются местами и заново переопределяются начальные и конечные точки.

Создаются переменные «r» (радиус исходной дуги), «o1» (центр дуги), «o2» (точка на середине дуги). Как и в случае «отрезок-отрезок», создаются два вектора (vect1,vect2) и происходит создание вспомогательного вектора «a» функцией AddFacet\_scale\_rotate(…) для параллельного сдвига отрезка. То есть создаются два вспомогательных отрезка, параллельных исходному. Также создаются два вспомогательных вектора для вспомогательных дуг, и происходит их нормализация. Так как мы не знаем, как расположена исходная дуга по отношению к отрезку (возможно 8 случаев), нам приходится создавать по две вспомогательных дуги (большего и меньшего радиуса). Это делается функцией AddFacet\_create\_arc(…). Она работает так: по расположению точки середины дуги (o2) начальные и конечные узлы новых дуг становятся в нужном порядке, задаётся направление обхода и радиус. Новые дуги получают идентификаторы carc\_up и carc\_down.

Для нахождения центра фаски вызывается функция AddFacet\_line\_arc\_point(…). В ней происходит выбор нужной точки (центра будущей дуговой фаски) из 4х возможных точек пересечения вспомогательных дуг со вспомогательными отрезками. Пересечения находятся функцией GetIntersection(…). Потенциальные точки хранятся в массиве. Вспомогательные дуги удаляются, и возвращается единственная точка – центр дуговой фаски. Если такой точки нет, то фаску построить нельзя, и возвращается false.

Далее находятся точки на исходных кривых, в которых дуговая фаска будет крепиться. Если хотя бы одна из них совпадает с угловой точкой, значит, фаску построить нельзя (касательная в угловой точке для исходной дуги и есть исходный отрезок). Функция AddFacet\_line\_arc\_create(…) создаёт итоговые кривые (отрезок, фаска и дуга). В конце происходит удаление двух исходных кривых.

bool C2DOutline::AddFacet\_2arcs(…) («дуга-дуга»).

Случай, когда обе кривые являются дугами. Определяются середины дуг (l1,l2), центры дуг (o1,o2) и радиусы (r1,r2). Находятся вектора (vect1,vect2), как и в предыдущих случаях. Создаётся 4 вспомогательных вектора для дуг (каждой по 2) и происходит их нормализация. На основе этих векторов получается 4 вспомогательных дуги (каждая большего или меньшего радиуса, чем исходная) функцией AddFacet\_create\_arc(…). Итого 4 идентификатора: carc1\_up, carc1\_down, carc2\_up, carc2\_down.

Две дуги могут образовать замкнутую область. Если это так, то вызывается функция AddFacet\_2arcs\_2points(…). Если нет, то AddFacet\_2arcs\_point(…). Различие в том, что в первой ищутся 2 точки пересечения и сравниваются между собой, а во второй – одна. Эти функции пересекает друг с другом вспомогательные дуги (GetIntersectionArcs2Points(…) и GetIntersectionArcs(…)) для нахождения точек пересечений – потенциальных центров фаски. Вспомогательные дуги удаляются, и выбирается единственная точка для фаски. Если такой точки нет, то фаску построить нельзя, и возвращается false.

Далее находятся точки на исходных кривых, в которых дуговая фаска будет крепиться. Если хотя бы одна из них совпадает с угловой точкой, значит, фаску построить нельзя. Функция AddFacet\_line\_arc\_create(…) создаёт итоговые кривые (дуга, фаска (тоже дуга) и дуга). В конце происходит удаление двух исходных кривых.