Варианты заданий

Важно: начиная с этой задачи и дальше по курсу обязательным условием для выполнения будет использование Git, причём нужно уметь работать с ним через консольную оболочку. Помочь в этом сможет “Приложение 1. Работа с Git через консоль”.

Репозиторий рекомендуется делать приватным. В отдельных случаях преподаватель может принимать задачи через него или попросить при сдаче продемонстрировать работу с Git.

Первый блок: алгоритмы растеризации.

В этом разделе представлены варианты, связанные с отрисовкой примитивов. В отличие от первой задачи курса, вы не можете использовать здесь готовые библиотечные методы. Единственная доступная опция — заливка цветом отдельного пикселя по его координатам.

Для ускорения работы рекомендуется воспользоваться вспомогательными проектами из репозитория.

Задачи отсортированы по возрастанию сложности. Первые три вообще следует брать только тем, кто отстаёт: тем, кто вынужден подтягивать базовые навыки программирования в данный момент.

Последняя задача с отрисовкой треугольника сложнее, чем кажется на первый взгляд. Её очень важно взять кому-то в группе и желательно взять сильному студенту. Код, который он напишет, будет использоваться остальными в следующих заданиях для растеризации полигонов.

При тестировании обязательно постройте все возможные варианты своей фигуры. Переставляйте координаты местами, сжимайте и растягивайте. Поворачивайте её, если это возможно. Очень частая ситуация, когда программа работает верно при одних входных параметрах, но ломается при других.

Для этих задач не нужен пользовательский интерфейс. Достаточно того, чтобы при запуске программы на экране по прописанным в коде точкам строились фигуры.

1. Реализовать метод (класс) для рисования отрезка алгоритмом DDA. Добавить возможность интерполяции цвета вдоль прямой: от одного конца отрезка до другого. На вход методу поступают координаты концов, цвет в этих точках и т.д.

2. Тоже самое, но с алгоритмом Брезенхэма.

3. Тоже самое, но с алгоритмом Ву.

4. Реализовать метод (класс) для рисования границ эллипса. На вход могут поступать координаты левого верхнего угла вместе с шириной и высотой фигуры. Или центр эллипса и размеры полуосей a и b. Возможны и другие варианты.

5. Реализовать метод (класс) для заполнения эллипса. Добавить возможность интерполяции цвета, например, при удалении от центра фигуры. На вход могут поступать координаты левого верхнего угла вместе с шириной и высотой фигуры. Или центр эллипса и размеры полуосей a и b. Возможны и другие варианты.

6. Реализовать метод (класс) для рисования дуги окружности. Добавить возможность интерполяции цвета от одного конца кривой до другого.

7. Реализовать метод (класс) для заполнения сектора окружности. Добавить возможность интерполяции цвета, например, при удалении от центра круга.

8*. Реализовать метод (класс) для заполнения треугольника. Добавить возможность интерполяции цвета с использованием барицентрических координат. На вход методу поступают координаты вершин треугольника, цвет в этих точках и т.д.

Второй блок: кривые и сплайны.

В выполнении этих задач поможет “Приложение 3. Введение в теорию сплайн-функций” и описанные внутри вспомогательные проекты. Программа должна быть холстом, на котором пользователь может с помощью кликов мыши создавать точки. По ним в реальном времени строится кривая. В коде она является ломаной линией, но отступ между точками берётся таким маленьким, что человек не замечает разбиения. Масштабирование, отрисовка координатных осей не требуются.

Важно, что строящаяся кривая не обязана быть функцией, то есть одному x в ней могут соответствовать несколько y. Для того, чтобы это было возможно, в некоторых вариантах придётся применить трюк, описанный во вспомогательном проекте, где заводятся две кривые: для осей x и y.

При желании в конце можно добавить возможность перетаскивать точки так, чтобы положение кривой обновлялось а реальном времени.

9. Реализовать интерполяционный полином Лагранжа.

10. Реализовать кубический сплайн.

11. Реализовать кривые Безье.

Третий блок: произвольные задачи.

В этом блоке несвязанные напрямую с лекциями задания на двумерную графику. За желание взять что-то масштабное придётся расплачиваться повышенной сложностью и большим количеством правок.

12. Реализовать программу для построения графиков произвольных функций $f(x)$. Пользователь вбивает функцию в текстовое поле. Ваш собственный интерпретатор разбирает строку, которая может содержать все основные операции, разные форматы чисел, элементарные функции, скобки. По полученному в коде выражению строится график. Этот график можно ещё и масштабировать, перемещать с помощью мышки.

13. Программа позволяет строить цветные фракталы. Для того, чтобы архитектура была универсальной, нужно реализовать минимум два множества: например, Мандельброта и треугольник Серпинского. Пользователь может переключаться между ними, перетаскивать картинку с помощью мышки и до бесконечности масштабировать на колёсико.

14. Написать конвертер произвольной цветной картинки в таблицу ASCII символов. Приложение может быть консольным. На вход поступает картинка, параметры. На выходе то же изображение, но составленное из символов. В качестве параметров следует передавать уровень сжатия: сколько пикселей превращается в один символ, а также цветное или монохромное изображение ожидается на выходе. Необходимо добиться лучшей передачи формы, самостоятельно разобраться, к каким трюкам пришли люди. Для этой цели следует в первую очередь отладить до совершенства монохромный режим. В конце же работы надо добавить режим батча: возможность работы с несколькими изображениями сразу. Обработать небольшую секвенцию кадров (видео).

15. Любое достаточно сложное задание, предложенное преподавателем.