ПРИМЕНЕНИЕ РАСЧЕТНЫХ МЕТОДОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ МАСШТАБИРОВАНИЯ ИЗОБРАЖЕНИЯ ПРИ РАЗРАБОТКЕ ГРАФИЧЕСКИХ ПРИЛОЖЕНИЙ

Андреев К.В., студент гр. ИТб-171, 2 курс
Научный руководитель: Асанов С.А., старший преподаватель
Кузбасский государственный технический университет
имени Т.Ф. Горбачёва
г. Кемерово

Аннотация

В статье рассматривается применение знаний математики для решения графических задач в программировании при разработке приложений. Как данные знания могут помочь для решения проблем с масштабированием фигур с учетом поворота и расположения на координатной плоскости относительно других объектов.

Содержание

В современном мире все больше специализаций охватывает IT сфера. Все автоматизируется, учеты ведутся в электронном виде, применяются учебные электронные ресурсы, Интернет-форумы, обучающие статьи, онлайн курсы, обучающее приложение-тьютор многое друго. В связи с некоторыми трудностями обеспечения взаимодействия с техническими средствами в реальных условиях, в обучающих целях используют имитирующие модели реальных систем. Данные модели имеют ряд преимуществ, такие как:

1) Мобильность – можно использовать в любых условиях и в любое время: дома на компьютере, в транспорте на мобильном устройстве и т.д.
2) Наглядность – возможно отображение таких параметров, которые невозможно отобразить в реальности (прохождение электронов по проводнику), или в таких местах, доступ к которым существенно затруднен.
3) Экономичность – меньше затраты на приобретение специализированного оборудования, площади для его размещения.

В 2018 году была поставлена задача разработать кроссплатформенное приложение виртуальной лабораторной работы по дисциплине «Электрооборудование автомобилей», которое обеспечивало бы имитацию работы системы зажигания (генераторной установки) по принципиальной электрической схеме (рис. 1).

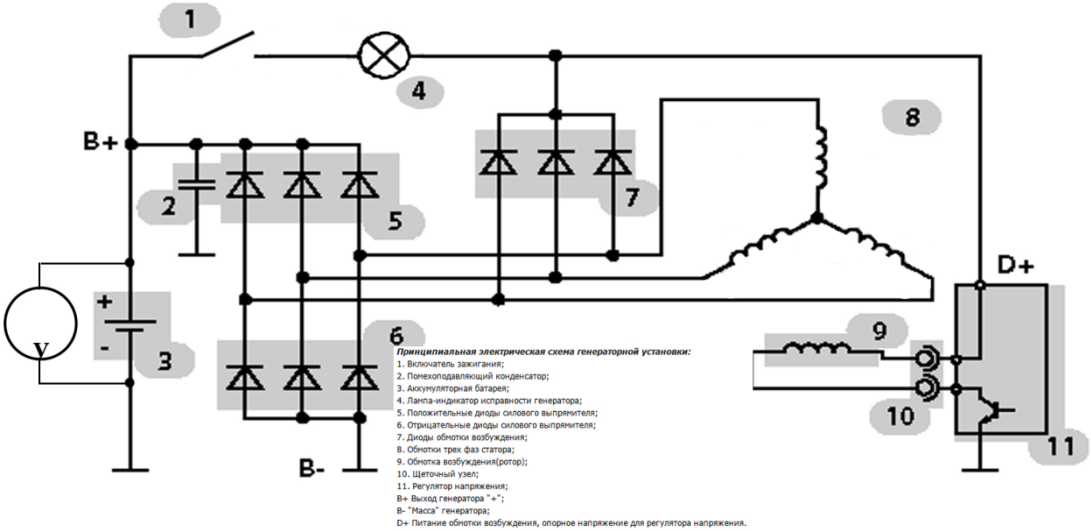

Рис. 1 – Принципиальная электрическая схема генераторной установки

Данная программа позволяет студентам изучить схему установки, собрать ее, подключив проводники между соответствующими приборами, запустить и сравнить показания на вольтметре с теоретическими.

С помощью графического интерфейса студент может:

1) собрать схему автоматически;
2) начать сборку заново, удалив все элементы;
3) открыть принципиальную электрическую схему и на ней наблюдать работу системы;
4) создать проводник, поменять цвет, перетащить, подключить к прибору или к другому проводнику;
5) открыть меню «помощь» для ознакомления с программой;
6) запустить программу в тестовом режиме без автоматической сборки;
7) запускать рабочую модель, экспериментировать с ней (допускаются ошибочные сборки модели).

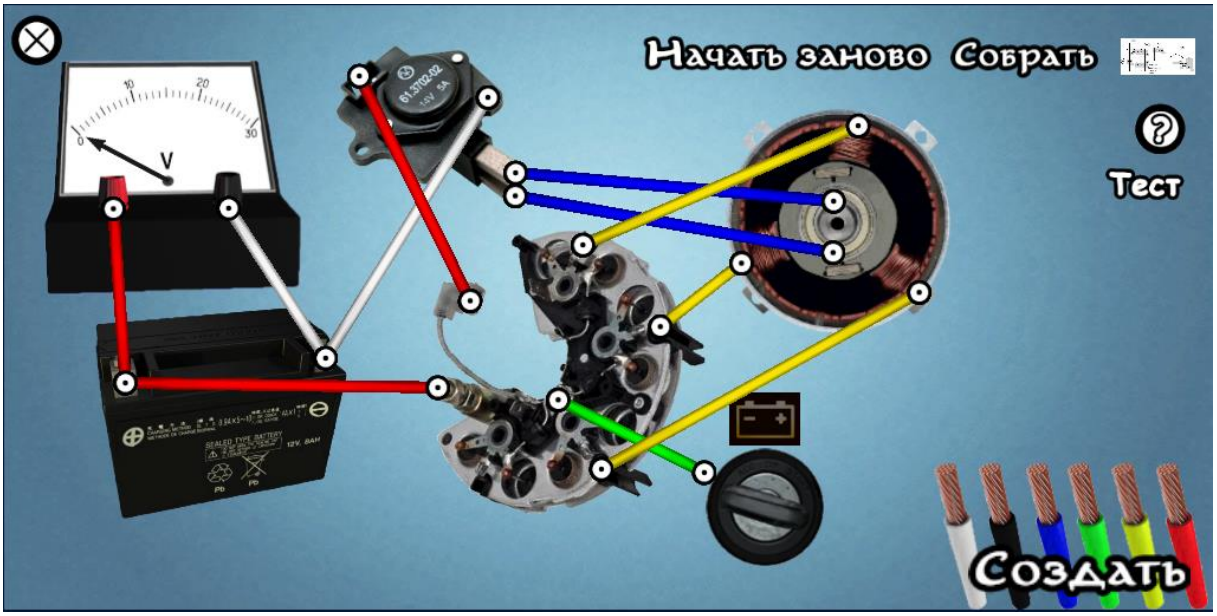

Рис. 2 – Собранная модель генераторной установки

С целью решения поставленной задачи о кроссплатформенности разработки, для реализации приложения была выбрана среда Unity, допускающая работу в самых различных комбинациях аппаратных и программных средств.

При разработке программы одной из главных проблем была проблема масштабирования объектов относительно друг друга. Опишем ситуацию на примере проводников.

Каждый проводник состоит из 3 частей, две соединительные точки и тело провода (рис 3.). Углы прямоугольного тела провода должны находиться за точками, создавая тем самым иллюзию проводника как единого объекта. Точки пользователь может передвигать на модели для подключения проводника к прибору или к другому проводку. Из этого следует, что тело должно всегда перемещать свой центр между двумя точками и масштабировать длину (увеличивая или уменьшая) с сохранением ширины, в любой момент времени соединяя данные точки.

Рис. 3 – Составные части проводника

Среда Unity для решения данной задачи встроенных инструментов не предоставляет, поэтому необходимо реализовать алгоритм самостоятельно.

В первую очередь надо разобраться со средствами отображения фигур. Unity предоставляет объект transform для отображения произвольного объекта в пространстве. Компонент в свою очередь состоит из следующих данных:

- position - положение объекта в координатах X, Y, и Z.
- rotation – вращение объекта вокруг осей X, Y, и Z, измеряется в градусах.
- localeScale – масштаб вдоль осей X, Y, и Z относительно наследуемого объекта. Значение “1” означает оригинальный размер.

Как видим, данные характеристики описываются векторами из 3 координат. Однако так, как у нас объекты плоские, то будем работать только с X и Y для размещения и манипуляции объектом. Координату Z будем использовать для отображения перекрытия объектов друг другом (для тела провода используем уровень 0.02f) и для вращения изображения.

Далее расмотрим смещение соединительной точки и изменение размеров проводника подробнее (рис. 4).

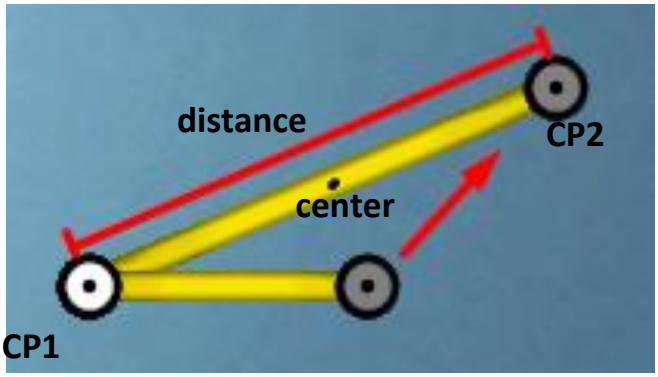

Рис. 4 – Наглядное смещение точки и изменение провода

После смещения серой точки в позицию CP2 поменялись и ее координаты. Данный факт поможет нам найти расстояние (distance) между CP1 и CP2, середину этого отрезка (center) и вращение (rotation).

Обозначим позиции точек векторами $\overline{P_1}(x_1, y_1)$ для CP1 и $\overline{P_2}(x_2, y_2)$ для CP2 соответственно. Используя знания математики, находим:

$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$\text{center} = \frac{\overline{P_1} + \overline{P_2}}{2}$$

$$\text{rotation} = \arccos\left(\frac{\overline{P_1} \cdot \overline{P_2}}{|\overline{P_1}| \cdot |\overline{P_2}|}\right)$$

Используя функцию Atan2(x, y), находящую угол вектора (x, y), можно формулу переписать в следующем виде:

$$\text{rotation} = \text{Atan2}(y_1 - y_2, x_1 - x_2) \cdot \frac{180}{\pi}$$

В итоге подставив в полученные формулы наши переменные, напишем метод по изменению тела провода при перемещении.

Код метода трансформации тела провода на языке C#:

```
public void Transform()
    {
        float distance = Mathf.Sqrt(Mathf.Pow((CP1.transform.localPosition -
CP2.transform.localPosition).x, 2) + Mathf.Pow((CP1.transform.localPosition -
CP2.transform.localPosition).y, 2));
        float rotation = Mathf.Atan2(CP1.transform.position.y -
CP2.transform.position.y, CP1.transform.position.x - CP2.transform.position.x)
* 180 / Mathf.PI;
       rotation = (rotation < 0) ? rotation + 360 : rotation;
       Vector3 center = (CP1.transform.position + CP2.transform.position) / 2;

         transform.localScale = new Vector3(ScalingFactor*distance, 0.02f, 0f);
         transform.rotation = Quaternion.Euler(0, 0, rotation);
         transform.position = new Vector3(center.x, center.y,
transform.position.z);
    }
```

где:

- CP1 – экземпляр объекта первой точки;
- CP2 – экземпляр объекта второй точки;
- transform – компонент расположения текущего тела;
- distance – расстояние между объектами точек CP1 и CP2;
- rotation – угол поворота итогового тела между объектами CP1 и CP2;
- ScalingFactor – коэфициент масштаба тела для данной модели;
- center – новая позиция тела в пространтсве;

Вывод

Таким образом, хорошее знание программистом математики и развитое ориентирование в пространстве и на плоскости, позволяют значительно уменьшить затраты времени на поиск решения, а также решать задачи, готовой реализации которых на момент разработки программы не существует.

Список литературы:

1. Руководство Unity. [Электронный ресурс] https://docs.unity3d.com/Manual/index.html
2. Изучение математики онлайн. [Электронный ресурс] https://ru.onlinemschool.com/
3. Г.Г. Литова, Д.Ю. Ханукаева, Основы векторной алгебры, учебно-методическое пособие для самостоятельной работы студентов. [Электронный ресурс] http://kvm.gubkin.ru/vector.pdf