Spring 3-2016

Simulating Electric Charges and Electric Force in Android

Arlind Hajredinaj

Computer Science and Engineering

Simulating Electric Charges and Electric Force in Android
Bachelor

Arlind Hajredinaj

Mars / 2016
Prishtine

Computer Science and Engineering

Thesis Paper
Academic year 2012 – 2013

Arlind Hajredinaj

Simulating Electric Charges and Electric Force in Android

Mentor: Phd. Bertan Karahoda

Mars/2016

This paper has been prepared and submitted in partial fulfillment of the requirements for the degree of Bachelor

Deklaratë

Prishtinë, 10/03/2016

Me vetëdije të plotë, unë Arlind Hajredinaj deklaroj se nënshkrimi im i vënë në fund të kësaj deklarate përmban zotimin tim që ky punim diplome është vepër imja origjinale, me përjashtim të atyre pjesëve të cilat janë të dokumentuara si duhet sipas rregullave të pranuara të citimit.

Unë po ashtu e kuptoj se ç'është plagjiatura, dhe deklaroj se i pranoj plotësisht të gjitha pasojat dhe veprimet të cilat do të ndërmerren nga profesori im dhe të tjerët përkitazi me mosfisnikërinë akademike ashtu siç e parasheh statuti i UBTse në Prishtinë po që se në punimin tim të diplomës gjendet plagjiatura.

Emri dhe Mbiemri si dhe nenshkrimi

_____

Adresa/ Address: Lagja Kalabria pn; 10000; Prishtinë; Republika e Kosovës

Telefoni/ Phone: 038 541 400; fax. 038 542 138

http://www.ubt-uni.net

ABSTRACT

Simulating physics phenomena is somewhat complicated, depending on what is to be simulated, the laws, the amount of space, the time. It all breaks down to the scale of the universe that we are going to simulate, we might want to draw objects, which means we need a screen that has pixels, a pixel size can be a meter, half a meter, millimeter or down to the smallest distance the Planck length which is $1.616199(97)\times10^{-35}$, we can simulate Newtonian laws, laws of Thermodynamics, theory of relativity, particles of light and such, the more physics we want to add means the more computer power and when we add more space and time to the simulation we will need a larger computer. In another way the time is actually not simulated by computers but it's imitated. Richard Feynman sets the following rule of simulation: the number of computer elements required to simulate a large physical system is only to be proportional to the space-time volume of the physical system. To simulate such physics we will need a Quantum computer. To simulate only electric charges and fields we can use simple computers, where the visible properties of electric charges and fields are described by variables which state are changed in time by an algorithm.

CONTENTS

TABLE OF FIGURES

GLOSSARY OF TERMS

GUI – Graphical User Interface

API – Application Programming Interface

2D – Two Dimensions

DDA –Digital differential analyzer

3D – Three Dimensions

HSB – Hue Saturation Brightness

UI – User Interface

CPU – Central Processing Unit

UML – Unified Modeling Language

1 INTRODUCTION

This thesis will describe the software simulation process of real world physics in a 2D landscape. The physics characteristics and behaviors which will be simulated are electric charges and fields. An Android application will be built to simulate all of the above. Electric charge can be transferred from one body to another; charged bodies can apply forces on other charged bodies and also on uncharged bodies. Electric charge can be a positive charge or negative charge; it is carried by elementary particles such as the electron and the proton. When electric charges are placed on electric fields, they start moving because of the forces applied to them. Electric force is almost the same as gravity where the more mass an object has the more gravitational force is applied to the object, you can think of electric charge as the mass of an object and the electric force as the gravitational force. The difference between mass and charge is that mass can only be positive will charge can be negative or positive We will not go into detail in explaining electric charges and forces, but we will go into detail on explaining and studying how we can simulate those electric charges, fields, and forces on a computer screen, using computer graphics techniques which we will be applied through the Android Graphics API.

Software simulation is used to represent and mimic a real world phenomenon. This thesis will describe the software simulation process of real world physics in a 2D landscape. The physics characteristics and behaviors which will be simulated are electric charges and fields. Simulating electric charges and fields is based on the same principles as all software simulations. An Android application will be built to deploy the simulation. Electric charge can be transferred from one body to another; charged objects can apply forces on other charged objects and also on uncharged objects. Electric charge can be a positive charge or negative charge; it is carried by elementary particles such as the electron and the proton. When electric charges are placed on electric fields, they start moving because of the forces applied to them. Electric force is almost the same as gravity where the more mass an object has the more gravitational force is applied to the object, you can think of electric charge as the mass of an object and the electric force as the gravitational force. The difference between mass and charge is that mass can only be positive will charge can be negative or

positive. We will go into detail on explaining and studying how we can simulate those electric charges, fields, and forces on a computer screen, using computer graphics techniques which will be applied through the Android Graphics API.

2 REVIEW OF LITERATURE

2.1 Physics

2.1.1 Basic Physics

"Physics is concerned with describing the interactions of energy, matter, space, and time, and it is especially interested in what fundamental mechanisms underlie every phenomenon. The concern for describing the basic phenomena in nature essentially defines the realm of physics" [1]. Physics tries to explain all the phenomena of the universe, most sciences such as chemistry, biology are built on the foundation of physics. Physics tries to answer questions about the universe, how it works, how the human body moves, and how things just are, great minds studied physics like Isaac Newton, Max Planck, Albert Einstein, Marie Curie and Stephen Hawking. Even today, in the modern age there is a lot to learn about the universe while it is just the beginning for the human beings; here is a quote by Isaac Newton:

"I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me." [2]

Energy is everywhere from light, thermal, electrical, nuclear etc. Matter is made up of atoms and atoms are composed of subatomic particles such as protons, neutrons electrons. Each atom has a nucleus that has protons and neutrons; the nucleus is surrounded by electrons which spin around the nucleus. Usually the number of electrons is the same as protons this makes the atom stable. Electrons are negatively charged and protons are positively charged, while neutrons are neutral but they add mass to the atom. In this thesis we are mostly interested in electrical energy. Below is a picture of an atom

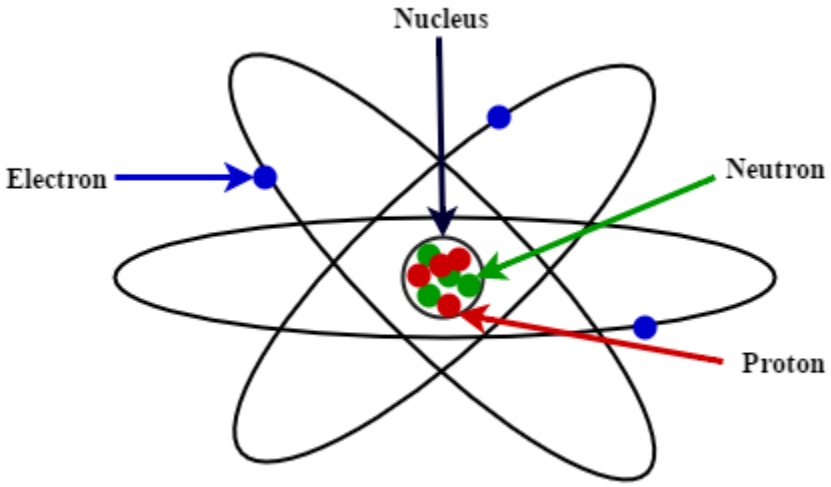

Figure 1. The structure of an atom

2.1.2 Motion

Motion is when objects move. "Motion is described in terms of displacement (x), time (t), velocity (v), and acceleration (a)" [3]. Displacement is the measure of how much did the object move from its first position to its new position, for example an object A is moved to a different position we can calculate the distance between the two positions and say that's the displacement. If we'd ask how fast did the object move from point A to point B? The answerlet's say would be 3 meters per second which is the speed and it is a scalar if we would add a direction to the speed we would be calculating the velocity of the moving object. Velocity is the rate of change of displacement, and the acceleration is the rate of change of velocity and is calculated with the formula:

$$v = \frac{s}{t}$$

Where s is the displacement in meters and t is time in seconds. Acceleration is the change in velocity over time; it is calculated with the formula:

$$a = \frac{v}{t}$$

2.1.3 Forces and the laws of motion

When we observe a moving object we can see that as it moves it will come to a stop where it does not move anymore this means that objects have a tendency to stop moving, this

assumption was believed until Isaac Newton, Galileo Galilei and Rene Descartes discovered that it works the other way around where objects have a tendency to maintain their velocity, unless acted on by unbalanced force. This means that a moving object will keep moving if no other force such as friction, gravity is applied to it. Newton's first law states that “an object will remain at rest or in uniform motion in a straight line unless acted upon by an external force” [4] what he meant by uniform motions in a straight line is that an object has a constant velocity, for example if we are in a car and are moving at a constant velocity in a way we are at rest and do not notice that we are in motion, until the moment the driver hits the breaks or speeds up (applies external force) which changes the acceleration and our velocity.

Newton’s second law “Force is equal to the change in momentum per change in time, for a constant mass, force equals mass times acceleration” [5].

$$F = m * a$$

To be clear, there is the acceleration of the object, F is the net force on the object, and m is the mass of the object.

2.1.4 Electricity and ElectricCharges

Electricity can be found and seen everywhere in nature, in lightning, solar storms, polar auroras there are even electric fish. Electricity is known as the flow of electric charge through certain objects. Charge is just another property of matter like mass. The electrons in the atoms of materials such as metals can easily escape the atom these materials are called conductors, basically all materials in the universe can to broken down to insulators or conductors for their electrical property. Imagine a copper wire which is connected to a light bulb, if electrons are flowing through the wire from one end of the wire to the light bulb and to the other end that is known as electric current which is used in our everyday lives to power our homes, offices, street lights and computers. The electrons in materials such as ceramic cannot escape the atom, these materials are known as insulators. When the a body holds more electrons than protons we say that that body is negatively charged and vice versa for positively charged bodies, Charged atoms are called ions. If we place small

charged bodies far from each where they don't interact with each other we call them point charges. No process is known which creates or destroys even the minutest amount of charge, electric charge is fundamental about the universe. "This is formally stated as the law of charge conservation: electric charge cannot be created or destroyed" [6] [7]. Electric charges are written with a *q*. A unit of charge is called a coulomb, the value of a basic unit charge is:

e = 1.602192 × 10–19 C

The system that we will be creating will contain lots of electric charges and we'll simulate the force to move the electric charges around in our "vacuum" 2D landscape.

2.1.5 Coulomb's Law

"Coulomb's law describes the magnitude of force between charges in a given distance between them" [7]. Let's say we have two point charges q1 and q2 in space with a distance r12 between them see figure 2

Figure 2. Electric charge attraction and repulsion

The charges will attract or repel each other, the force at which they are moved is called the electrical force and is the force that Coulomb's law describes. Electrical force is the same as all forces where it has magnitude (strength) and direction it is a vector. Coulomb's force law between two point charges q1 and q2 located at r1 and r2 is calculated with the formula:

$$F = k_e \frac{q_1 q_2}{r^2}$$

Where$k_e = 9.0x10^9 \frac{Nm^2}{C^2}$

$k_e$is the electrostatic constant which is used just to make the formula work. In our simulation we don't have to use it, and do not have to square the distance between them, we could just divide by distance, the only difference it will make is that the force will be larger.

The formula is almost the same with Newton's formula for gravitational pull, electric force unlike gravity works at really small distances, and gravity in longer distances for stars, moons and planets, but the electric force is much stronger and can overcome the force of gravity in short ranges atomic level.

2.1.6 Electric field

The region around a charged body within which it can exert its electrostatic influence may be called an electric field.

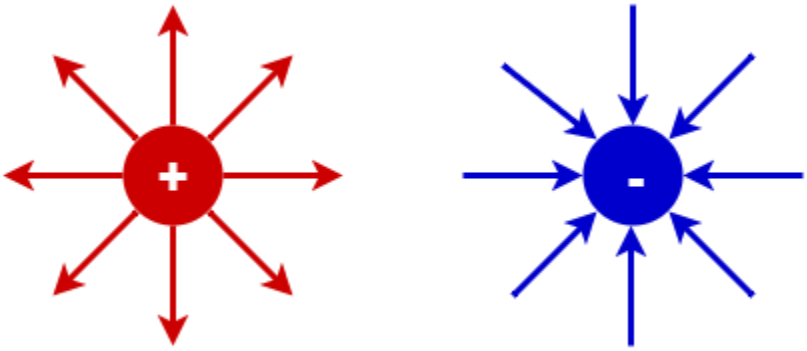

Figure 3. Electric field lines

The electric field generated by the charge at a given point r is calculated with the formula:

If we add another charge near the first charges electric field, then a force will be applied to the added charge, also the added charge applies an opposite force to the first charge.

2.2 Computer Graphics

2.2.1 Entry to Computer Graphics

Computer displays are found all around us, which makes computer graphics so widespread and extensively used. Computer graphics are a core part of Computer Science, when studying computer graphics the main focus will be in mathematics, physics and computer science. Math is needed to form geometric shapes and to move them around on the screen, when moving objects, we need to understand velocity and acceleration which has to do with physics, we can add color to the objects so we have to add light and shades and know how light is reflected and how colors are made.

Most simulations use computer graphics to express the overall shape and internal attributes of objects on a screen. Movement of objects can be calculated to show smooth real time motion, which allows us to observe the behavior of objects while moving. Motion can be simulated by moving object positions slightly from their positions in the previous frame. “All screens are made up of screen points which are called pixels or pel” [8], these pixels are refreshed redrawn on the screen a number of times a second which is called the refresh rate and there is also a frame rate which is the number of frames drawn on a screen per second. A screen can have a frame rate of 24 frames but the refresh rate can be a 60 Hz. Graphics Software comprise of programming packages/libraries which provide graphics functions that can be used in programming languages, these functions may be of drawing lines, circles, ellipses, curves, transforming objects, filling areas, drawing colors and different shapes which are called drawing primitives, other functions include transforming objects, moving objects, coloring etc. Graphic libraries are designed to work with Cartesian coordinate systems. Generally the functions can be used to draw objects on the screen, as all functions they also have attributes, these attributes describe how the primitives will be drawn by height, width, x and y positions, colors etc. Complex drawings and images can be made from x:y matrices which have different colors on the points which form the matrix. Below is an example, let’s say we want to draw a line which is accomplished by drawing all the points between the lines starting and ending point, an algorithm for line drawing is used which is called the DDA Algorithm or Bresenham's line drawing algorithm. The

library functions use a low level native library to perform the actual graphics drawing which are faster.

Usually in simulations, games and animations, incremental changes are made to the animations model at regular intervals, and then the screen is updated to represent the simulation models, this is known as a drawing loop or game loop.

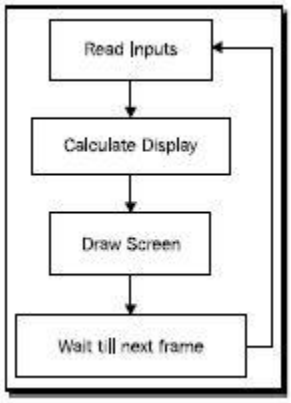

Figure 4. Gaming loop

The Android framework has a 2D graphics API (Application Programming Interface), which provides canvases, color filters, points, rectangles, that we can draw on the screen directly.

2.2.2 Android Graphics API

The Android Graphics API is very similar to all other graphics APIs; it supports two different kinds of 2D drawing:

- Drawing graphics or animations into a View object. In this manner, the drawing of graphics is handled by the system's normal View hierarchy drawing process. This option is used for drawing simple graphics,
- Drawing graphics directly to a Canvas. We can reserve an area of the screen for drawing freely, where we can look after the redrawing by our self, and optimize it for whatever the application does. This way we can call different drawing functions and perform more complicated drawing. A Canvas works as an interface; to the actual surface upon which graphics will be drawn it holds all of the draw calls. All

the drawing calls draw on a bitmap which in turn is placed into the window shown on screen.

Android has a special subclass of the class View called SurfaceView which includes a dedicated raw area of the screen where drawing is handled. The SurfaceView provides a canvas object where drawings can be performed from a thread different from the main android UI thread; this enables that drawing to be performed faster and more efficiently.

3 PROBLEM DEFINITION

Consider a professor lecturing physics inclass. The class would be incomplete without the classrooms laboratory which holds different equipment which is more than necessary to perform laboratory examples. Our problem is that there is a lack of physics simulation software applications which would aid in the student's learning process in universities, colleges or high schools that teach physics mainly subjects for electricity, electric charges and electric fields.

Here is a brief introduction to the goals of the simulation:

- Determine the variables that affect how the electric charges interact and move.
- Describe how charged bodies will interact with each other.
- Describe the strength of the electric field around an electric charge.

To simulate real world phenomenon we need to represent time as our simulation clock, each time the simulation clock ticks new values for the properties are calculated for our world objects, such as new positions, electric charge, directions, force magnitude, acceleration.

Given the time which is infinite, the simulation can never run out of time, but the simulation has a starting point and has an ending point, both of which we can declare and chose when to start, pause or finish the simulation clock. In the real world, objects are constantly moving and have different forces applied to them at all times which means all properties are recalculated all the time. In our simulation we have to simulate the time and simulate the recalculations which cannot happen all the time due to processing power limitations, but let’s say the calculations are done every 1 second or 500 milliseconds a value that we can chose this is called the refresh rate. We’ll use a refresh rate of 40hz which will give a frame rate of 25 that makes the simulation graphics run smoothly, and the recalculations may not be too processor intensive.

After each tick of the world clock all properties of objects will be recalculated, so we first need to know what properties do the objects have, in physics we can think of objects as matter and matter has different properties and states.

Let’s start with a study of real world objects and what can we do with these objects, find an object near you and pick it up, now ask yourself these questions:

1. How do we know that the object is there and that we can grab it?
2. When we pick up the object why is it going up?
3. Why is it harder to pick up bigger objects?
4. When we drop the object why does it fall?

Firstly, we will have to actually see the object to pick it up and to grab it has to be there this means that the object has mass which is measured in kilograms and never changes for that object, what I mean by "never changes" is that the object can be anywhere in the universe and it always has the same mass.

Secondly and thirdly to pick up an object we have to apply force to the object, the more mass the object has the more force we have to apply to move the object. The more force we apply the harder it is for us. Force is a vector which means it has its strength (magnitude) and direction, the direction of the force indicates the direction the object will move to. Our universe has 3 dimensions 4 if we include time, so while we are moving the object we are actually changing its 3 dimensional coordinates x, y and z.

Fourthly, Isaac Newton studied the Universal Law of Gravitation which is the law that describes why the object falls in the direction towards earth. Isaac Newton was sitting under an apple tree when an apple fell from the tree and he thought as the apple was hanging on the tree it had zero velocity(which is the speed of an object in a given direction) and then the apple is accelerated(acceleration is the increase in the rate of speed) and moves towards earth. So there must be a force which is gravity that gave the apple acceleration.

We are simulating electric charges which do not have gravity forces instead they have electric forces. Electric forces can be repelled or attracted. Charges also have a property called charge.

From the study above we can conclude that objects have the following properties:

1. Mass
2. Velocity
3. Acceleration
4. Position in a 3D coordinate system
5. Force applied to them.

4 METHODOLOGY

This thesis is based on the knowledge that I have gained during my years of studying at the University of Business and Technology and elsewhere, here I will blend the knowledge from different subjects Basics of Electronics, Mathematics, Algorithms, Programming in Java and Android. I will start of explaining some basic physics principles for electric charges, electric force, Coulomb's law and Electric field, then the literature will explain how will the simulation model works through physics and mathematics, towards the end we will capture all software elements and components that are combined to create the electric charge and force field simulation.

5 RESULTS

5.1 How the Simulation Works

5.1.1 The Mathematical Algorithm

The simulation that we are building will be in a two dimensional space, so the objects will only have x and y coordinates. Here is an example of two charges q_1 is negative and $q_2$positively charged in one dimensional space let's say they are sitting on the x axis which means y is equal to 0 for both charges.

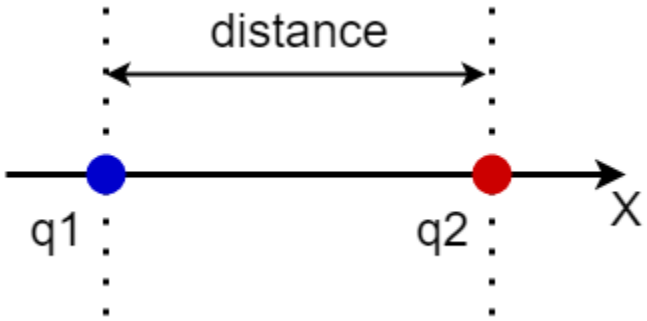

Figure 5. First example of electric force calculation

$q_1$has a charge of 1C and $q_2$has a charge of 3C, $q_1$is positioned on x at 1 and $q_2$is positioned on x at 5. The distance between them is 4. Now we can calculate the force between them.

$$q_1 = 1C, q_2 = 3C, x_1 = 1m, x_2 = 5m\text{r} = 5\text{m} - 1\text{m} = 4\text{m}$$

$$F = k_e \frac{q_1 q_2}{r^2} = 9.0 * 10^9 \frac{Nm^2}{C^2} * \frac{1C * 3C}{4^2 m^2}$$

$$F = 9.0 * 10^9 \frac{Nm^2}{C^2} * \frac{3C^2}{16m^2}$$

$$F = \frac{3}{16} * 9.0 * 10^9 N = \frac{27}{16} * 10^9 N$$

$$F = 1.6875 * 10^9$$

The resulting force is applied to both charges. When force is applied to an object we can use Isaac Newton second law of motion which describes how velocities change when force is applied, The relationship between an object's mass m, its acceleration a, and the applied force $F = m * a$, acceleration and force are vectors and have directions, in this law the direction of force is the same as the direction of acceleration. Now we can calculate the charge's acceleration and velocity acceleration from the last example:

$$a = \frac{F}{q}$$

$$a_1 = \frac{1.6875N}{1C}$$

$$a_1 = 1.6875\frac{m}{s^2}$$

$$a_1 = \frac{1.6875N}{3C}$$

$$a_2 = 0.5625\frac{m}{s^2}$$

Now, we can change the charges x position by just adding acceleration to the object which adds velocity to its current position.

$$v = v + a$$

Now, here is take an example of two charges which the y positions are not equal to 0 and the charges have different y positions from each other.

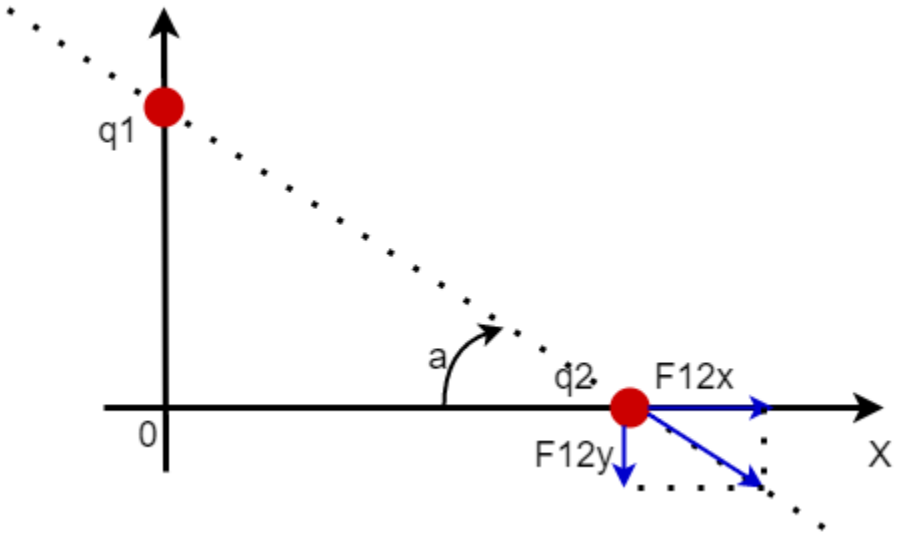

Figure 6. Second example of electric force calculation

In this example the same rules apply just that the distance between two charges is found with the Euclidian distance formula:

$$d = \sum_{i=1}^{n} (P_{1i} - P_{2i})^n$$

Therefore we have to calculate the force direction which is the angle between the two charges; we can find the angle of the force vector which is the same as the angle which the $q_1$charge is pointing to $q_2$charge, by using the atan2 which converts rectangular coordinates to polar. The atan2 method computes the phase theta (θ) by computing an arctangent of y/x in the range of - $\pi$to π. Here is the formula

$$\tan^{-1}\frac{y_2 - y_1}{x_2 - x_1}$$

If the two charges are repelled then the forces direction (angle) that we found is in the opposite direction which means we have to add 180° degrees or the πconstant for radians to the forces direction.

Finally we have everything we need to draw the force which applies to the two charges q_1 andq_2.

In our simulation there can be more than two charges at a given moment, so we have to calculate the force for each charge that applies force and sum all the forces to get the

resultant force. Since force is a vector to add to forces we have to check the direction of the forces which changes if we are adding or subtracting. Forces that have the same direction (north and north, south and south, west and west, east and east) can be added and forces with opposite directions are subtracted (east and west or north and south). There is another case where forces might be aligned as in the image below

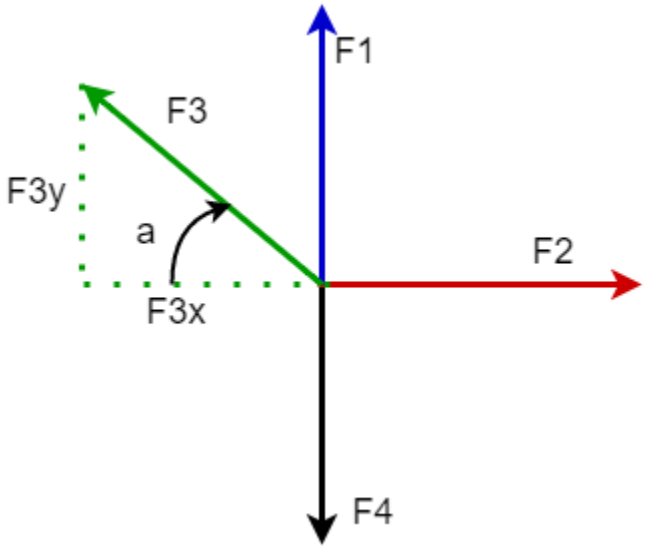

Figure 7. Example of force vector calculations

We can see thatF_3 is not in a horizontal or vertical direction, we can use trigonometry to calculate the F_3magnitude on the horizontal plane and its magnitude on the vertical plane, let's say $F_{3x}$is the horizontal force magnitude which is equal to:

$$F_{3x} = \cos\theta * F_3$$

$$F_{3y} = \sin\theta * F_3$$

And now we can add $F_{3x}$to the sum of forces on the horizontal axis and $F_{3y}$to the sum of forces vertical axis.

To calculate the electric field around a charge the same process applies only that the x and y coordinates are every point coordinate in the simulation and the points do not have charge, so we use the same formula only that $q_2 = 1$ is 1.

5.1.2 The Algorithm

The algorithm that I have implemented is explained below with pseudo code on how everything fits together and the workflow, steps, iterations of the algorithm.

The simulation has a Timer which is set to tick every 40 milliseconds, each time the timer ticks the values of the properties of objects in the simulation are recalculated and the view is redrawn, the charges may have different positions, charge, and force applied to them. We defined the properties of charges that are to be used:

- Position, x and y coordinate
- Velocity, speed for x and y coordinate
- Acceleration, acceleration for x and y coordinate
- Charge, electric charge in Coulombs
- Type, type of charge -1(negative) or 1(positive)

An array is used to store all the electric charges that are in the simulation. Below the pseudo code is presented for calculating the force and moving the charges.

```
FOR all electric charges in the electric charge list
   force_rect is initially 0
   last_force_rect is initially 0

   FOR all electric charges in the electric charge list except the one before // so we don't
calculate the force for the same charges

      force_magnitude = calculate force magnitude between two charges
      theta_angle = calculate the angle between the charges

      IF the type of charge is the same for the two charges
         add PI to theta_angle

      force_rect = convert polar force coordinates to rectangular coordinates from
(force_magnitude and theta_angle)
      addlast_force_rect to force_rect // to get the resultant force

      last_force_rect equal to force_rect // save the last force for next iteration calculation
   END FOR
   calculate the acceleration for x and y coordinate for the first charge by force_rect // first
charge charge( Coulombs)
END FOR
```

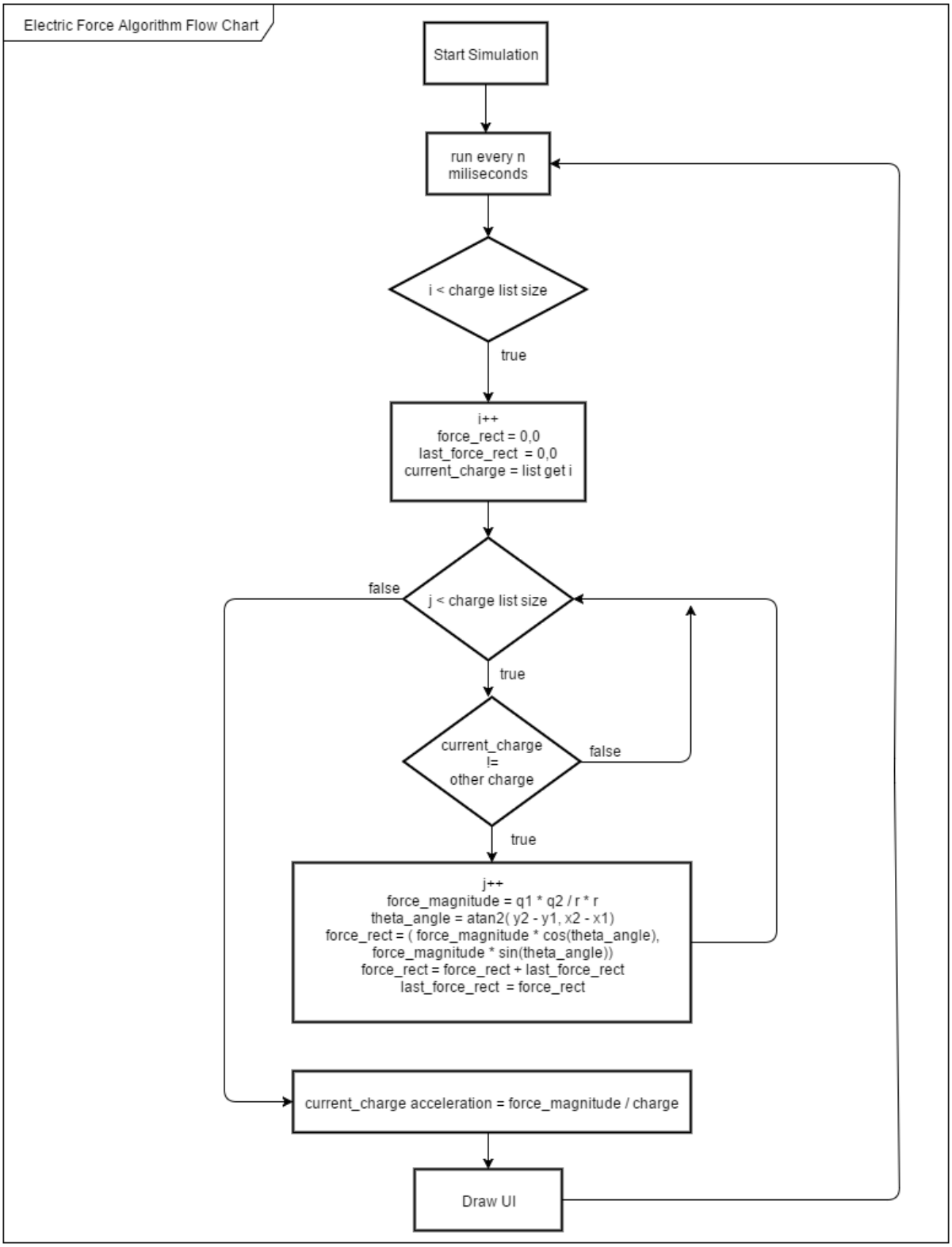

Figure 8. Algorithm flowchart

We iterate through the array to calculate the force that is applied from other charges, thus we use a second iteration through the array to get the other charges we exclude the charge that force is applied to, because an electric charge does not apply force to itself. We add up all forces that are applied from each electric charge and store them. The mathematics that is used here is explained in the previous section.

The same algorithm is used to generate the electric field, the resultant force is not applied to any charge, it is used as an amount of brightness in an HSB(Hue Saturation Brightness) color. The amount of brightness can be from 0 - 1 a decimal number, 0 is black and 1 is bright. When the force is larger than 1 it is set equal to 1.

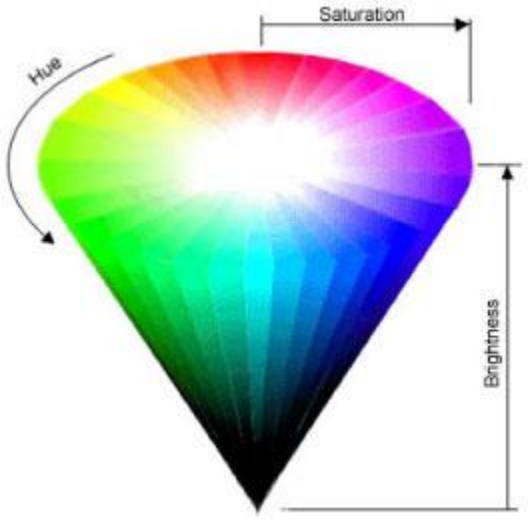

Figure 9. HSB color spectrum

The electric field is calculated for each pixel on screen and drawn as a rectangle. There is resolution variable which by default is one, as the name states it's used to set the resolution of pixels(rectangles) being drawn a resolution of 1x1 is ideal in our case but to speed up the calculations a resolution of 5x5 can be used. The resolution is the increment of pixels on the width x and height y axis of the whole screen, also it states the width and height of the rectangle being drawn.

5.2 Software Components

5.2.1 UML diagrams

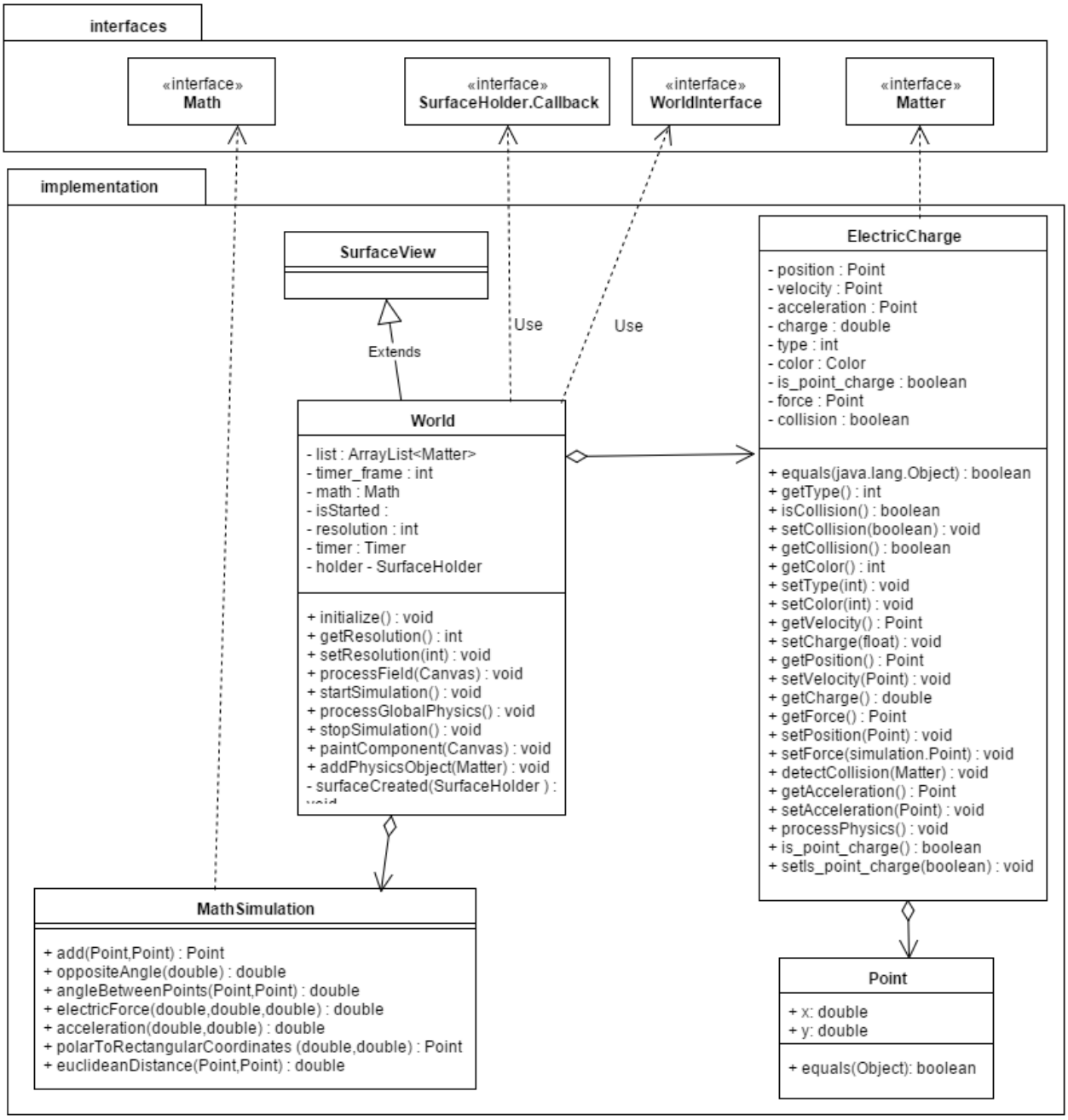

Figure 10. UML Diagram of classes

5.2.2 Class and Method Descriptions

The following packages with classes are used in the simulation:

Interfaces

- Math
- PhysicsObject,
- WorldInterface

Exceptions

- SimulationException is a specific exception class which extends the Extends class and is used to denote an anomaly in the simulation such as: matter cannot have negative mass, or a positive charge and negative are repelled.

Classes:

- MainActivity, is an Android component which is used to interact with the user, it takes care of creating a windows and placing views in their specific positions. The onCreate() method is used to layout our view, initialize our simulation, add a couple of electric charges to the simulation. In the activity layout we have the World surfaceview and the field views on the bottom, the field views are:
    - Checkbox for hiding/showing the electric field, when it is clicked the world object's setShowElectricField(boolean) method is called.
    - Checkbox adding point charges which do not move, when force is applied to them. when this field is clicked the electric charge's setIsPointCharge(boolean) method is called.
    - Checkbox adding negative/positive charges to the simulation. When it is clicked the electric charge's setType(int) method is called.
    - Seekbar adds the number of coulomb's to the charge. When the progress of the seekbar is changed the electric charges setCharge(double) method is called.

- Seekbar is used to change the resolution of the electric field. When the progess of the seekbar is changed the worldssetResolution(int) method is called.
- Button pauses/starts the simulation. When this button is clicked the world startSimulation() or stopSimulation() method is called depending on which state the simulation is in.

- Electric Charge is used to represent a real electric charge in the simulation. This class implements Matter interface. Below is a list of methods and their descriptions:
  - equals(Object) : boolean checks to see if two Electric charges are the same, they are compared by positions and type(positive or negative).
  - + getType() : int returns 1(positive) or -1 (negative)
  - + isCollision() : boolean returns true when the charge collides with another charge
  - + setCollision(boolean) : void when two charges collide this method is called
  - + getColor() : int returns the charges color
  - + setType(int) : void sets the charges type 1 (positive) or -1 (negative)
  - + setColor(Color) : void sets the charges color by default charges with positive type are blue and negative type are red.
  - + getVelocity() : Point. Returns the velocity vector in rectangular format
  - + setCharge(float) : void sets the electric charges charge in magnitude
  - + getPosition() : Point. returns the position on screen of the charge in x and y coordinates
  - + setVelocity(Point) : void. sets the velocity vector in rectangular format.
  - + getCharge() : double. returns the electric charges charge.
  - + getForce() : Point. sets the velocity vector in rectangular format.
  - + setPosition(Point) : void. sets the position of the charge on screen in x and y coordinates.
  - + setForce(simulation.Point) : void. Returns the force vector in rectangular format.

- + detectCollision(Matter) : void. In our case a collision is when two electric charges have a distance of 0, the distance is found with the euclidean distance formula and subtracting the charges radius which by default is 10. If the distance is 0 the x coordinate position of this charge is adding by 20, this animates the charge as if it has collided. Aside from changing the charges position, the charges charge is recalculated by subtracting the charges and dividing the charge in half and setting resultant charge amongst the two colliding charges.
- + getAcceleration() : Point. Returns the acceleration vector in rectangular format.
- + setAcceleration(Point) : void. sets the force vector in rectangular format.
- + processPhysics() : void. Adds velocity to the charges position and acceleration to velocity only if the charge is not a point charge, in which case the charge is not to be moved.
- + is_point_charge() : boolean. return true if this charge is not supposed to move.
- + setIs_point_charge(boolean) : void sets the is_point_chargeboolean value.

- MathSimulation is used for mathematical calculations and contains the formulas for phyics calculations. Below is a list of methods and the descriptions:
  - + add(Point,Point) : Point. Adds two point objects together by adding their x coordinates, and y coordinates.
  - + oppositeAngle(double) : double. The parameter should be an angle value in radians, the method adds PI two the angle two change its direction into the opposite direction
  - + angleBetweenPoints(Point,Point) : double. Calculates the angle which points from the first point two the second point.
  - + electricForce(double q1,double q2,double distance) : double. Calculates the force which is applied to the charges. The electric force is calculated with Coloumbs law.

    - + acceleration(double force,double mass) : double. Calculates the acceleration given force and mass, with Newtons second law of motion.
    - + polarToRectangularCoordinates (double distance,double theta) : Point. Converts the polar vector given distance and theta to rectangular vector.
    - + euclideanDistance(Point,Point) : double. Calculates the distance between two points with Euclidean distance formula.
- Point has only an x and y attributes, it's used the represent Caucasian coordinates and vectors in rectangular formats.
- World is an android view which extends SurfaceView class, it handles all drawings to the screen that is, electric field, charges, vectors, and charges. This class also holds the algorithm for force field generation and calculating the electric force vectors for the electric charges. Below is a list of methods and the descriptions:
- + initialize() : void. This method is called in all available constructors. This method specifies the initial state of world object by adding surfaceholder callbacks, initializing touch listeners, and creating the paint objects.
- + getResolution() : int. returns the electric field drawing resolution.
- + setResolution(int) : void. sets the electric field drawing resolution
- + processField(Canvas) : void. This method uses the electric field algorithm to draw the field on the canvas parameter.
- + startSimulation() : void. Starts the timer and runs it at a fix rate which is specified in the timer_frames attribute. When the timer runs the new positions for charges are calculated, force field is generated, and everything is redrawn on canvas. The timer can be paused/stopped by calling stopSimulation() method.
- + processGlobalPhysics() : void This method uses the electric field algorithm to generate the new positions for the electric charges in the simulation
- + stopSimulation() : void. Stops/pauses the simulation, by stopping the timer, the timer can be restarted by calling the startSimulation() method
- + paintComponent(Canvas) : void. Handles the drawings to the canvas which is available through the Surface Holder lockCanvas method and it is released with

unlockCanvasAndPost(Canvas). This method draws the electric charges in their positions, charge text, and the electric force vector.

- + addPhysicsObject(Matter) : void. This method adds electric charges the objects list attribute. This method is thread safe, it can be called to add matter objects whenever the world object is available.

5.2.3 GUI (Graphical User Interface)

The simulation's GUI is rather simple to use, it comprises of only one screen/page where all interactions with the user are handled. The complex drawings of the simulation are handled by world object, and the controls/views that we use to control and interact with the simulation are handled in the activity class. Below is a screenshot of the simulation which contains a couple of charges.

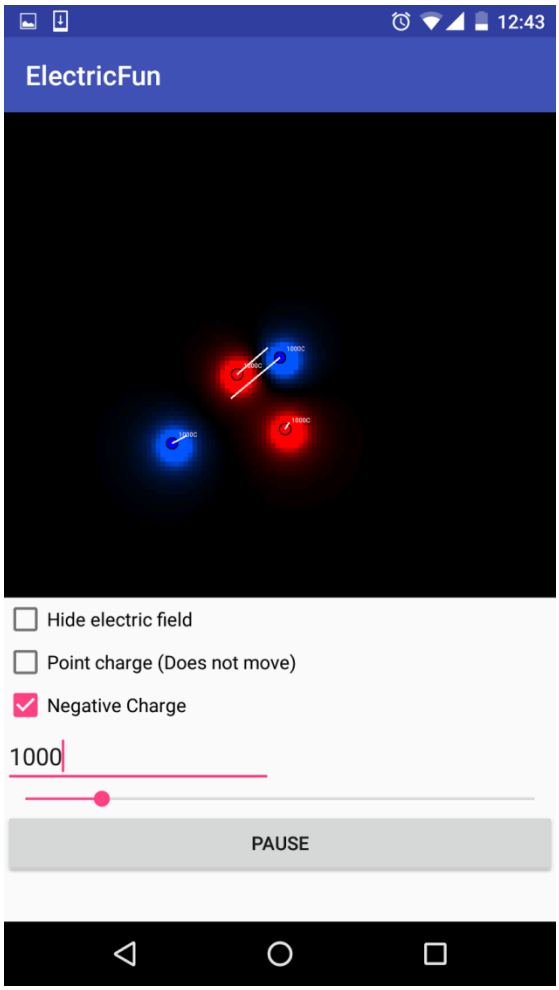

Figure 11. Screenshot of the main UI

The screen is divided into two parts the simulation visual graphics, and the simulations UI controls. The UI controls contain the following views and event listeners:

- TouchListener world view: When the simulation is touched/tapped, a charge is added into that position. The charges properties are gathered from the UI controls properties.
- Checkbox, Electric field: Control is used to show or hide the electric field, when we click this control is checked the electric field will be hidden and when it is not checked the electric field will be shown.

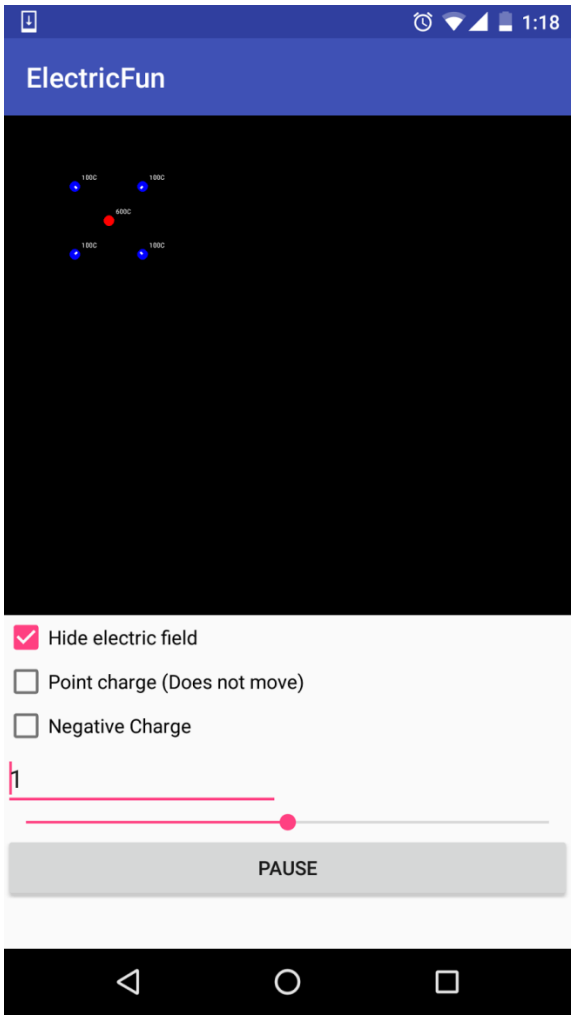

Figure 12. Screenshot, hidden electric field

- Checkbox, Point charge: Control is used to set the electric charges property for movement, when it is checked it means that the electric charge that will be added to the simulation cannot move (velocity will always equal to 0).
- Checkbox, Negative charge: Control is used to set the electric charges property for type, when it is checked it means that the electric charge that will be added to the simulation will be negative (type equals to -1) with a blue color and when it is not clicked the charge will be positive (type equals to 1) with a red color.
- SeekBar, Coulombs: Control is used to set the electric charges property for charge, when the progress is changed it means that the electric charge that will be added to the simulation will have that amount of charge.

- SeekBar Resolution: Control is used to set the resolution of the electric field which is a property of the world object. Below are a two images one with high resolution and the other with pixelated low resolution.

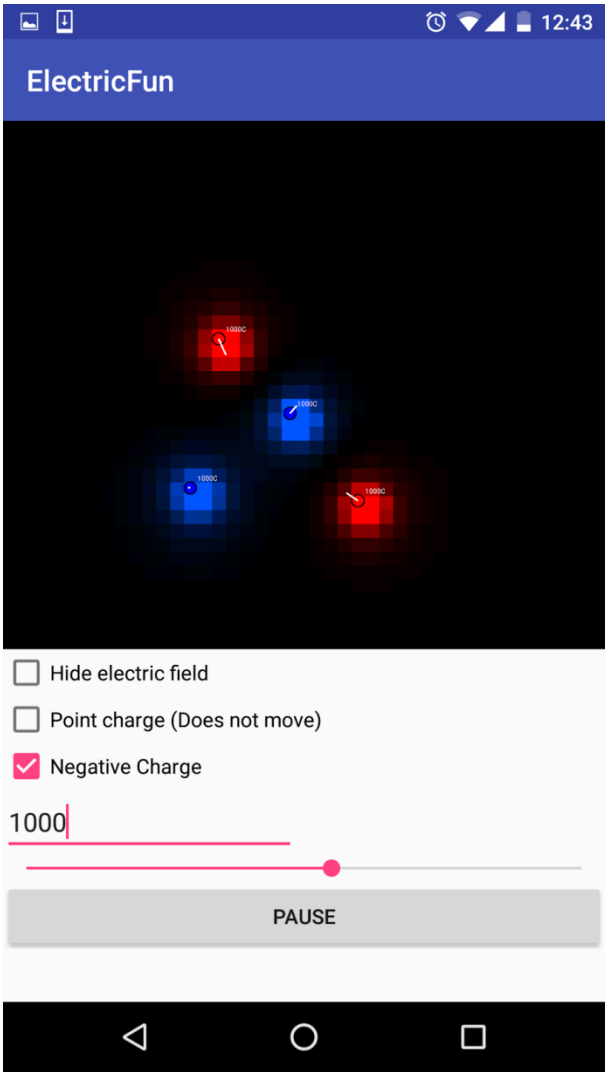

Figure 13. Screenshot, low resolution pixilated electric field

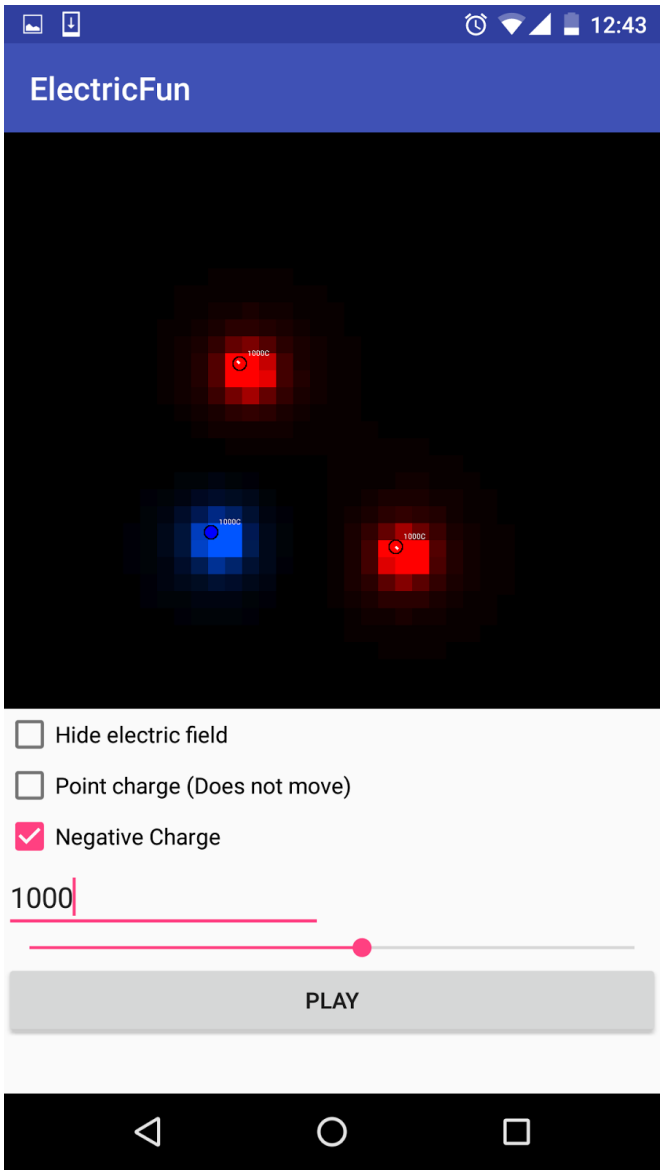

Figure 14. Screenshot medium resolution electric field

- Button Play/Pause: Control is used to start/stop or resume/pause the simulation. When it is in pause state all drawings and calculations will be paused. The drawings and calculations will be resumed when the button is a start state.

6 DISCUSSIONS AND CONCLUSIONS

6.1 Recommandations

A download link:

https://raw.githubusercontent.com/arlindiDev/ElectricChargeSimulation/master/app/out/artifacts/app_jar/app.jar

can be found for the Android application or the Java Application. I suggest using the Java application since the simulation requires more processing power and laptops or pc's have more CPU power than android smart phones. The source code is on the following Github repository https://github.com/arlindiDev/ElectricChargeSimulation

As I finish my bachelor studies, I know have a clearer vision from my starting point, I now know what I want to specialize and that is in Software Engineering and Computer Graphics two topics which are very broad and contain a lot of interesting topics such as:

- 2D computer graphics
- 3D computer graphics
- Alpha composition
- Anti-aliasing
- Curves
- Bitmap Graphics
- Motion
- Vector Graphics
- Shaders

While most programmers have little knowledge of mathematics and physics, I personally reccomend studing both topics, since programming evolved from phyiscs, electric engineering and mathematics. Most mathematicition can start programming fairly easy, even implement their mathematical models to provide algorithms for solving different problems. For most programmers studying mathematics seems as daunting task, but while studying computer graphics mathematics are a must, and I highly reccomend at least

learning algebra, trigonometry and linear algebra and consider your mathematics to be an ongoing process throughout your career as a programmer.

In the references section I reccomend reading the books that I've citated from and read to write the thesis.

6.2 Conclusion

The goal of this thesis was to explain how to simulate electric charges in software mainly in android, but the same principles apply to any other platform Windows (C#, Java, Python), IOS (Objective C, Swift), or web based (JavaScript). During my BA studies I have gained knowledge in the fields of electricity and magnetism, programming, computer graphics, and software engineering. I wanted this thesis to be a combination of all these subjects in theory and to a create practical example which is the Android electric charge and force field simulation application, where I had the chance to combine and enhance my perception around these fields of study.

The simulation can be used by students to learn and make sense of the unfamiliarity's and behaviors of electric charges and electric fields around these charges. It is easier to learn by visual graphics and animations that are conducted by the modeling simulations. Professors can use the simulation to enhance learning during lectures, and to engage the students even more with the learning process.

7 REFERENCES

[1] P. P. R. H. K. D. a. M. S. Urone, College Physics, Houston, TX: OpenStax College, Rice U, 2013, p. 6.

[2] I. Newton, " Newton Exhibition - Isaac Newton at work," Cambridge University Library , 01 12 2000. [Online]. Available: http://www.lib.cam.ac.uk/exhibitions/Footprints_of_the_Lion/introduction.html. [Accessed 20 01 2016].

[3] [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/mot.html. [Accessed 17 01 2016].

[4] P. P. R. H. K. D. a. M. S. Urone, in *College Physics*, Houston, TX, Rice University, OpenStax College, 2013, p. 164.

[5] P. P. R. H. K. D. a. M. S. Urone, "College Physics," in *College Physics*, Houston, TX, OpenStax College, 2013, p. 285.

[6] R. G. Brown, "Introductory Physics I," in *Elementary Mechanics*, Durham, NC, Duke University, 1997, p. 194.

[7] J. B. Tatum, "Electricity and Magnetism," in *Physics topics*, 2000, p. 29.

[8] D. a. M. P. B. Hearn, "Computer Graphics, C Version," in *Computer Graphics, C Version*, Upper Saddle River, NJ, Prentice Hall, 1997, p. 40.

Google,"Canvas andDrawables. [http://developer.android.com/guide/topics/graphics/2d-graphics.html], date accessed: 15 Jan. 2016

Google,"Graphics." *Graphics*.Web. [https://source.android.com/devices/graphics/] date accessed: 15 Jan. 2016.

Hearn, Donald, and M. Pauline.Baker. *Computer Graphics, C Version*. Upper Saddle River, NJ: Prentice Hall, 1997. Print.

Pickover, Clifford A. *The Physics Book: From the Big Bang to Quantum Resurrection, 250 Milestones in the History of Physics*. New York: Sterling Pub., 2011. Print.

Shaw, Alex. *Android 3.0 Animations: Beginners Guide: Bring Your Android Applications to Life with Stunning Animations*. Birmingham, UK: Packt Pub., 2011. Print.

Shirley, Peter, and Michael Ashikhmin. *Fundamentals of Computer Graphics*. Wellesley, MA: AK Peters, 2005. Print.

Tatum, J. B. "Physics - Electricity and Magnetism." *Physics - Electricity and Magnetism*.Web. 21 Jan. 2016.

Georgia State University.[*Http://hyperphysics.phy-astr.gsu.edu/hbase/newt.html*.].date accessed: 20 Jan. 2016.

Urone, Paul Peter., Roger Hinrichs, Kim Dirks, and Manjula Sharma. "College Physics." College Physics. Houston, TX: OpenStax College, 2013. 285.