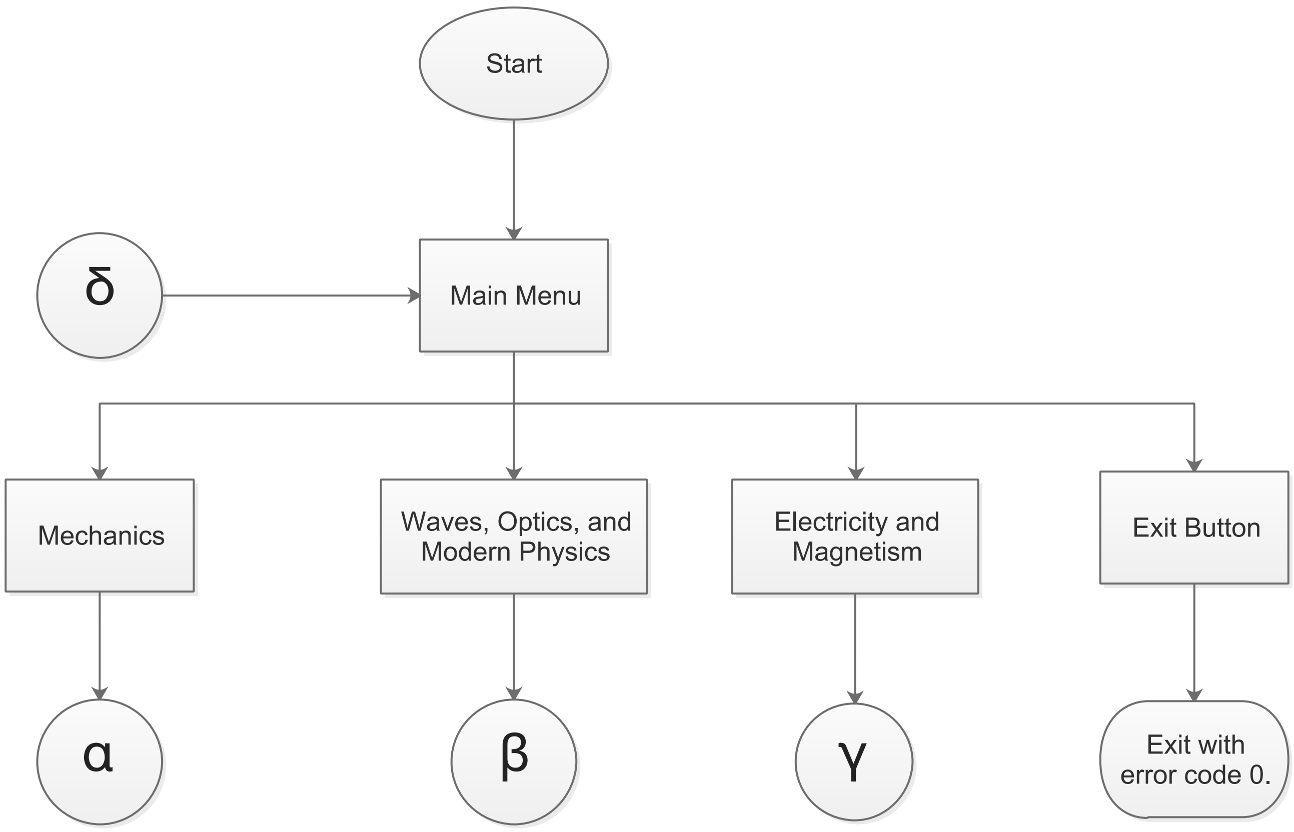
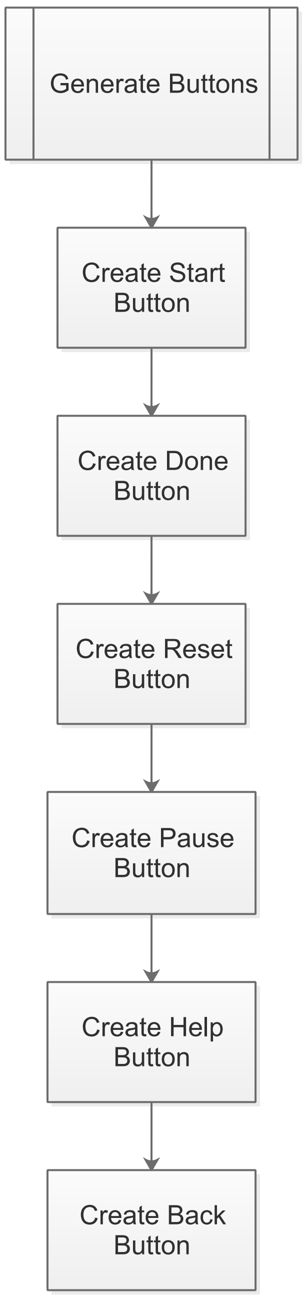




Vect2D:
A class that stores a 2-dimensional vector or point.

List:
A class capable of storing elements of a specified object type

Button:
A GUI element which triggers an event when clicked by the user

TextField:
A GUI element where the user is capable of inputing a string
Whenever a Number is required from this text field it will be parsed and if the value inputed is not a number, it will assume the input to be 0
protected int getInt() : Returns an integer input from the text field.
If no integer is found return 0
protected int getFloat() : Returns an float input from the text field.
If no integer is found return 0.0

Label:
A GUI element capable of displaying a string to the user

LineChart:
A class that represents a line graph based on two variables.

abstract PhysicalObject:
protected Vect2D position
protected Vect2D velocity
protected Vect2D acceleration
protected Image image
public abstract void applyForce(Vect2D force): Applies an accelerative force
public static double constrain(double value, double min, double max): constrains the value variable between min and max
protected void move(): Changes the velocity and position based on the velocity and acceleration.
protected void update(): Updates the object's position on screen relative to the its center.
public Vect2D getPosition()
public void setPosition(Vect2D position)
public Vect2D getVelocity()
public void setVelocity(Vect2D velocity)
public Vect2D getAcceleration()
public void setAcceleration(Vect2D acceleration)
public Image getImage()

static class ClsMain:
Primary class of the entire algorithm. Contains the main menu allowing the user to select a topic to view.

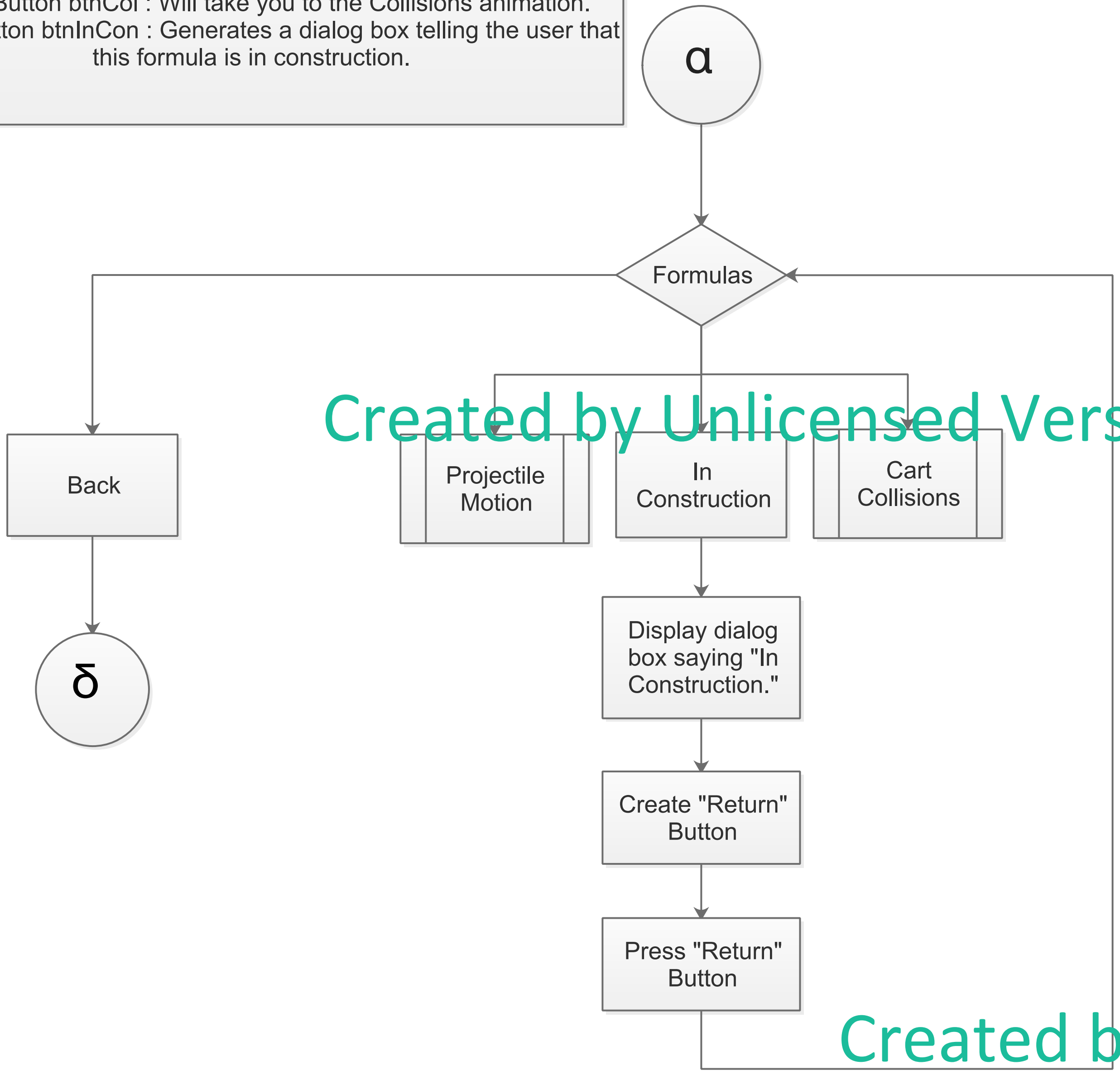
protected Button btnMech : Will take you to the Mechanics menu.
protected Button btnWav : Will take you to the Waves menu.
protected Button btnEM : Will take you to the Electricity menu.

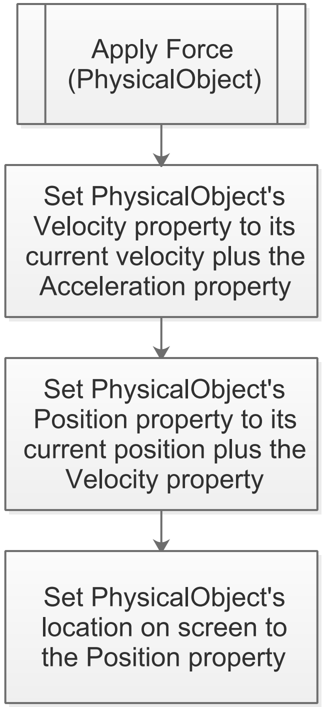
interface IConstants:
Interface containing all global constants used by every program.

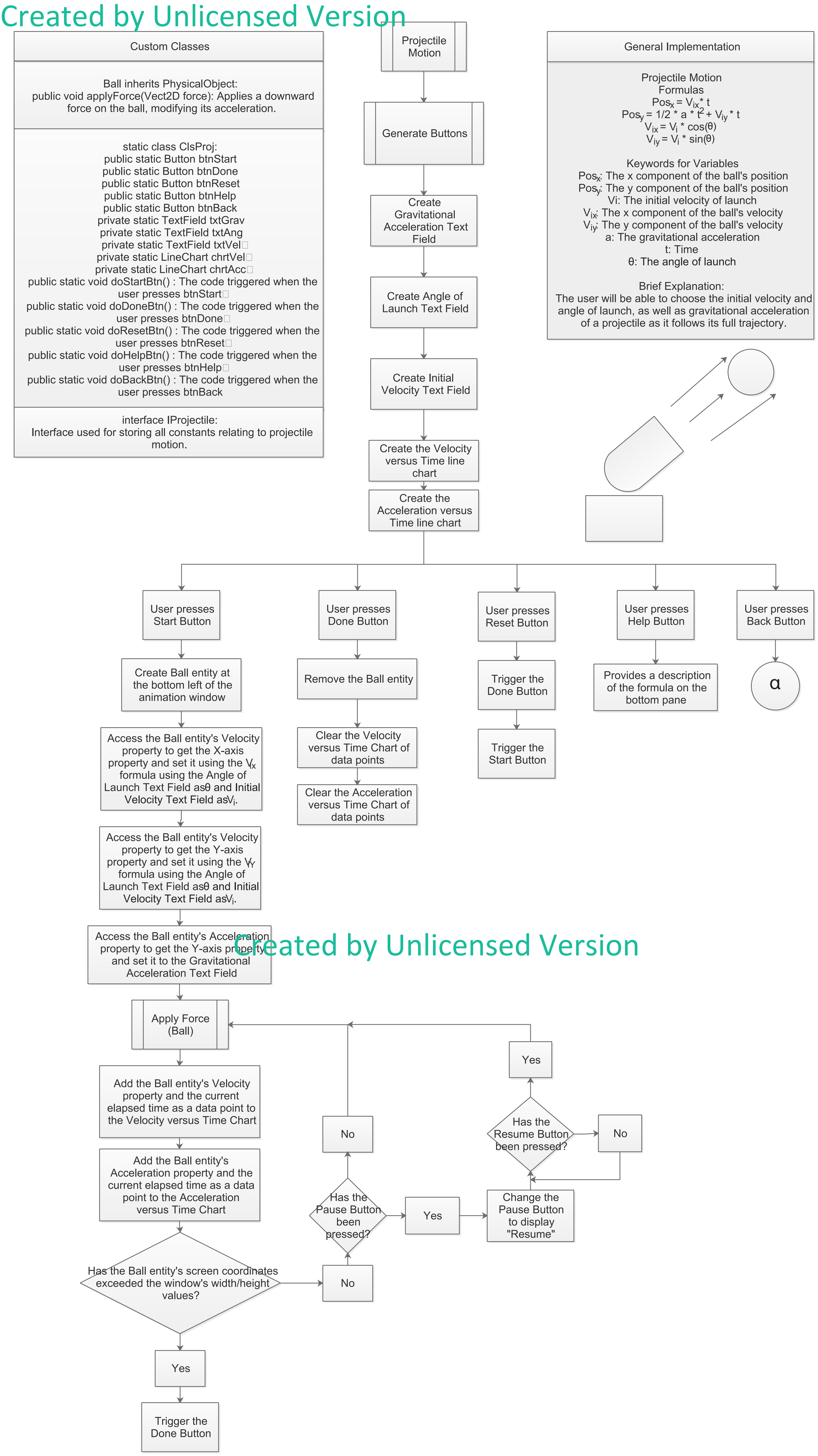
static class ClsHandlers:
Class handling the events triggered by every Button object throughout the algorithm.
All data related to what the button's actions are is stored in this class.

static class ClsMech:
Class containing the menu to take the user to one of two Mechanic formulas

public Button btnProj : Will take you to the Projectiles animation.
public Button btnCol : Will take you to the Collisions animation.
public Button btnInCon : Generates a dialog box telling the user that this formula is in construction.




Apply Force (PhysicalObject)
Set PhysicalObject's Velocity property to its current velocity plus the Acceleration property
Set PhysicalObject's Position property to its current position plus the Velocity property
Set PhysicalObject's location on screen to the Position property



Custom Classes

Ball inherits PhysicalObject:
public void applyForce(Vect2D force): Applies a downward force on the ball, modifying its acceleration.

static class ClsProj:
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
private static TextField txtGrav
private static TextField txtAng
private static TextField txtVel□
private static LineChart chrtVel□
private static LineChart chrtAcc□
public static void doStartBtn() : The code triggered when the user presses btnStart□
public static void doDoneBtn() : The code triggered when the user presses btnDone□
public static void doResetBtn() : The code triggered when the user presses btnReset□
public static void doHelpBtn() : The code triggered when the user presses btnHelp□
public static void doBackBtn() : The code triggered when the user presses btnBack

interface IProjectile:
Interface used for storing all constants relating to projectile motion.

General Implementation

Projectile Motion
Formulas
$Pos_x = V_{ix} * t$
$Pos_y = 1/2 * a * t^2 + V_{iy} * t$
$V_{ix} = V_i * \cos(\theta)$
$V_{iy} = V_i * \sin(\theta)$

Keywords for Variables
Pos_x: The x component of the ball's position
Pos_y: The y component of the ball's position
Vi: The initial velocity of launch
V_{ix}: The x component of the ball's velocity
V_{iy}: The y component of the ball's velocity
a: The gravitational acceleration
t: Time
θ: The angle of launch

Brief Explanation:
The user will be able to choose the initial velocity and angle of launch, as well as gravitational acceleration of a projectile as it follows its full trajectory.

Projectile Motion
Generate Buttons
Create Gravitational Acceleration Text Field
Create Angle of Launch Text Field
Create Initial Velocity Text Field
Create the Velocity versus Time line chart
Create the Acceleration versus Time line chart
User presses Start Button
User presses Done Button
User presses Reset Button
User presses Help Button
User presses Back Button
Create Ball entity at the bottom left of the animation window
Remove the Ball entity
Trigger the Done Button
Provides a description of the formula on the bottom pane
α
Access the Ball entity's Velocity property to get the X-axis property and set it using the Vx formula using the Angle of Launch Text Field as θ and Initial Velocity Text Field as Vi.
Clear the Velocity versus Time Chart of data points
Trigger the Start Button
Clear the Acceleration versus Time Chart of data points
Access the Ball entity's Velocity property to get the Y-axis property and set it using the Vy formula using the Angle of Launch Text Field as θ and Initial Velocity Text Field as Vi.
Access the Ball entity's Acceleration property to get the Y-axis property and set it to the Gravitational Acceleration Text Field
Apply Force (Ball)
Yes
Add the Ball entity's Velocity property and the current elapsed time as a data point to the Velocity versus Time Chart
No
Has the Resume Button been pressed?
No
Add the Ball entity's Acceleration property and the current elapsed time as a data point to the Acceleration versus Time Chart
Has the Pause Button been pressed?
Yes
Change the Pause Button to display "Resume"
Has the Ball entity's screen coordinates exceeded the window's width/height values?
No
Yes
Trigger the Done Button

Custom Classes

Cart inherits PhysicalObject:
private float mass
public void applyForce(Vect2D force): Unused for this object and thus unimplemented.
public float getMass()

static class ClsColl:
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
private static TextField txtM1
private static TextField txtM2
private static TextField txtVel
private static RadioButton rdbtnEla
private static RadioButton rdbtnInEla
public static void doStartBtn() : The code triggered when the user presses btnStart
public static void doDoneBtn() : The code triggered when the user presses btnDone
public static void doResetBtn() : The code triggered when the user presses btnReset
public static void doHelpBtn() : The code triggered when the user presses btnHelp
public static void doBackBtn() : The code triggered when the user presses btnBack

interface ICollisions:
Interface used for storing all constants relating to elastic and inelastic collisions.

General Implementation

Collisions
Formulas
Conservation of Kinetic Energy
(Elastic): $m_1 * V_1 = m_2 * V_2$
Conservation of Kinetic Energy
(Inelastic): $m_1 * V_1 = (m_1 + m_2) * V_2$

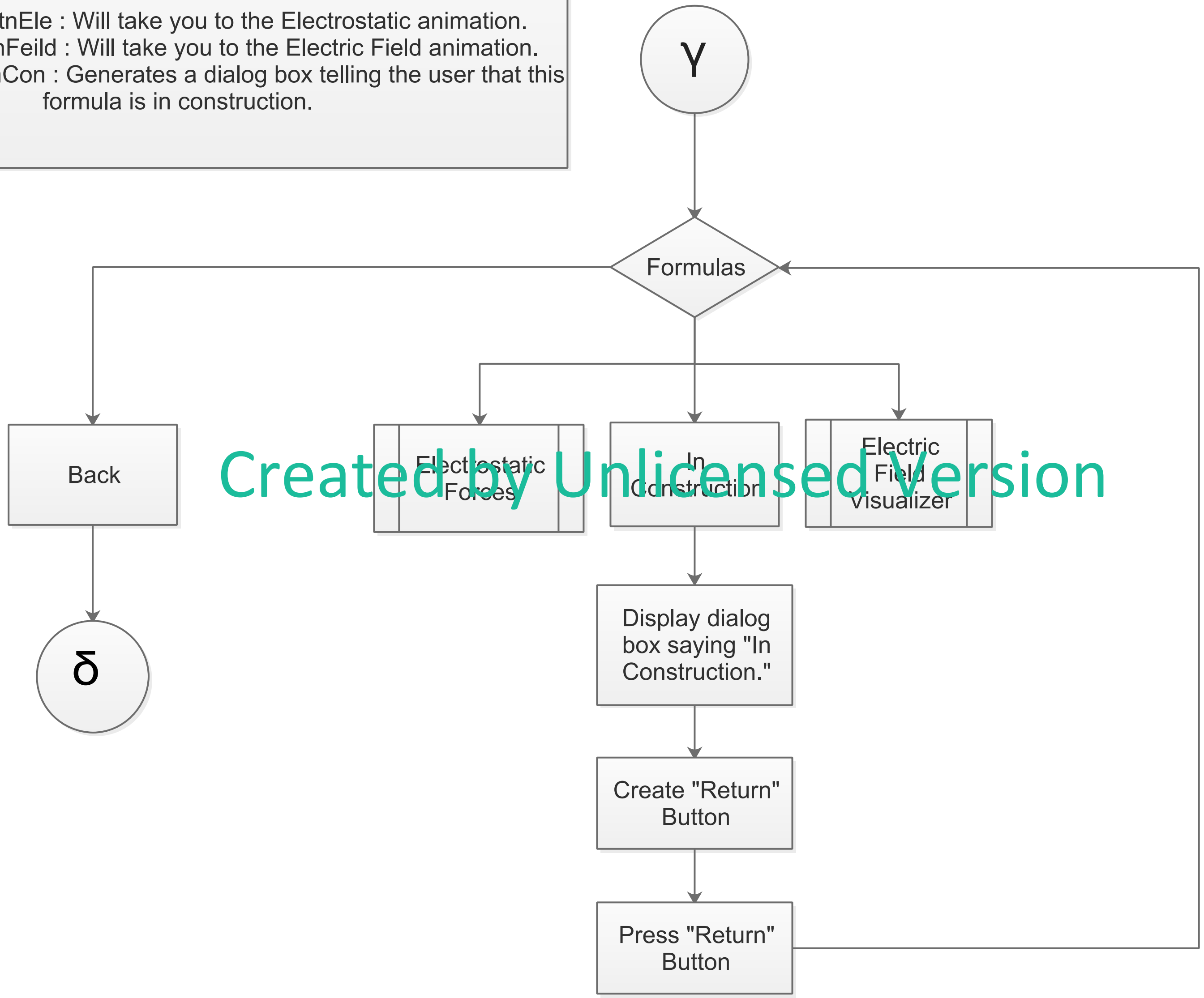
Keywords for Variables
Cart 1: An instance of Cart entity, to be the souce of impact
Cart 2: An instance of Cart entity, to be the target of impact
m_x: mass of x
V_x: velocity of x

Brief Explanation
The user will be able to choose the velocity of the source of impact, and decide the type of ensuing collision between it and the target of impact.

Cart Collisions
Generate Buttons
Create Cart 1 Mass Text Field
Create Cart 2 Mass Text Field
Create Initial Cart 1 Velocity Text Field
Create Collision Type Radio Button Group
Add Elastic Radio Button to Collision Type
Add Inelastic Radio Button to Collision Type
Select Elastic Radio Button as default
Create the Velocity versus Time line chart
User presses Start Button
User presses Done Button
User presses Reset Button
User presses Help Button
User presses Back Button
Create Cart 1 entity at the bottom left of the animation window
Create Cart 2 entity at the bottom center of the animation window
Remove Cart 1 entity
Remove Cart 2 entity
Clear the Velocity versus Time Chart of data points
Trigger the Done Button
Trigger the Start Button
Provides a description of the formula on the bottom pane
α
Apply Force (Cart 1)
Add the Cart 1 entity's Velocity property and the current elapsed time as a data point to the Velocity versus Time Chart
Has the Cart 1 entity collided with Cart 2 entity?
No
Has the Pause Button been pressed?
No
Yes
Change the Pause Button to display "Resume"
Has the Resume Button been pressed?
No
Yes
Yes
Collision type Selected?
Elastic
Set Cart 2 entity's Velocity property to a number based on the Conservation of Kinetic Energy forumula, with Cart 1 entity's Velocity and Mass properties
Set Cart 1 entity's Velocity property to 0
Apply Force (Cart 2)
Add the Cart 2 entity's Velocity property and the current elapsed time as a data point to the Velocity versus Time Chart
Has Cart 2 entity's screen coordinates exceeded the window's width/height values?
No
Has the Pause Button been pressed?
No
Yes
Change the Pause Button to display "Resume"
Has the Resume Button been pressed?
No
Yes
Yes
Trigger the Done Button
Inelastic
Set Cart 1 and Cart 2 entity's Velocity properties to a number based on the Conservation of Kinetic Energy forumula, with Cart 1 and Cart 2 entity's Velocity and Mass properties
Apply Force (Cart 1)
Apply Force (Cart 2)
Add the Cart 1/2 entity's Velocity property and the current elapsed time as a data point to the Velocity versus Time Chart
Has Cart 2 entity's screen coordinates exceeded the window's width/height values?
No
Has the Pause Button been pressed?
No
Yes
Change the Pause Button to display "Resume"
Has the Resume Button been pressed?
No
Yes
Yes
Trigger the Done Button

static class ClsEM:
Class containing the menu to take the user to one of two Electricity & Magnetism formulas

public Button btnEle : Will take you to the Electrostatic animation.
public Button btnFeild : Will take you to the Electric Field animation.
public Button btnInCon : Generates a dialog box telling the user that this formula is in construction.



Custom Classes

Particle inherits PhysicalObject:
private bool selected
private float mass
private float charge
public Vect2D attract(Particle p2): Returns an attraction force based on the Mass propety and displacement between the two Particle entities
public void applyForce(Vect2D force): Applies a force on the given Particle entity
public float getMass()
public void setCharge(float charge)
public float getCharge()
public void setSelected(bool selected)
public bool isSelected()

static class ClsEle:
public static List particles
private int mode
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
public static Button btnAdd
public static Button btnSelect
public static Button btnRemove
private static TextField txtCharge
private static TextField txtEleConst□
private static RadioButton rdbtnEla□
private static RadioButton rdbtnInEla□
public static void doStartBtn() : The code triggered when the user presses btnStart□
public static void doDoneBtn() : The code triggered when the user presses btnDone□
public static void doResetBtn() : The code triggered when the user presses btnReset□
public static void doHelpBtn() : The code triggered when the user presses btnHelp□
public static void doBackBtn() : The code triggered when the user presses btnBack

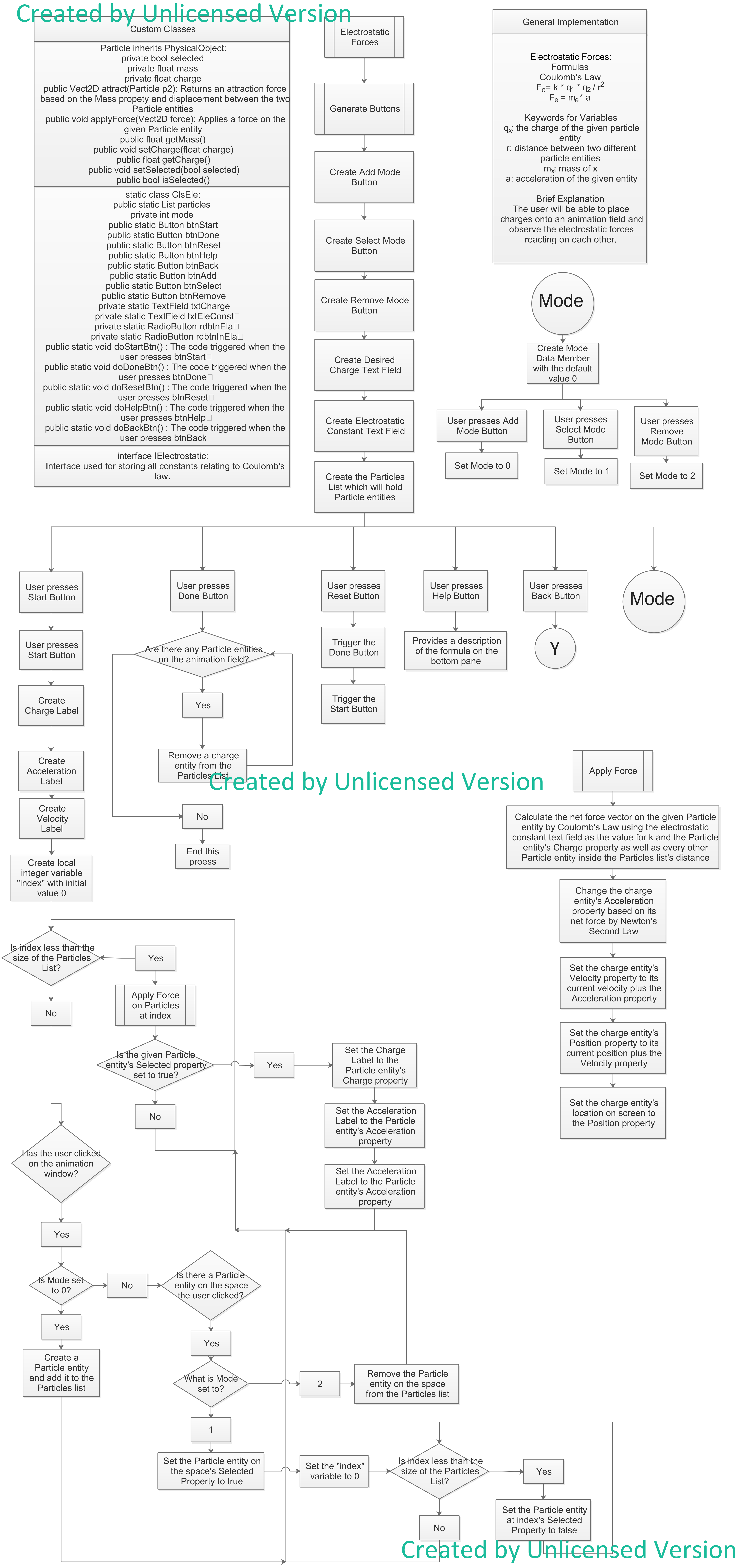
interface IElectrostatic:
Interface used for storing all constants relating to Coulomb's law.

General Implementation

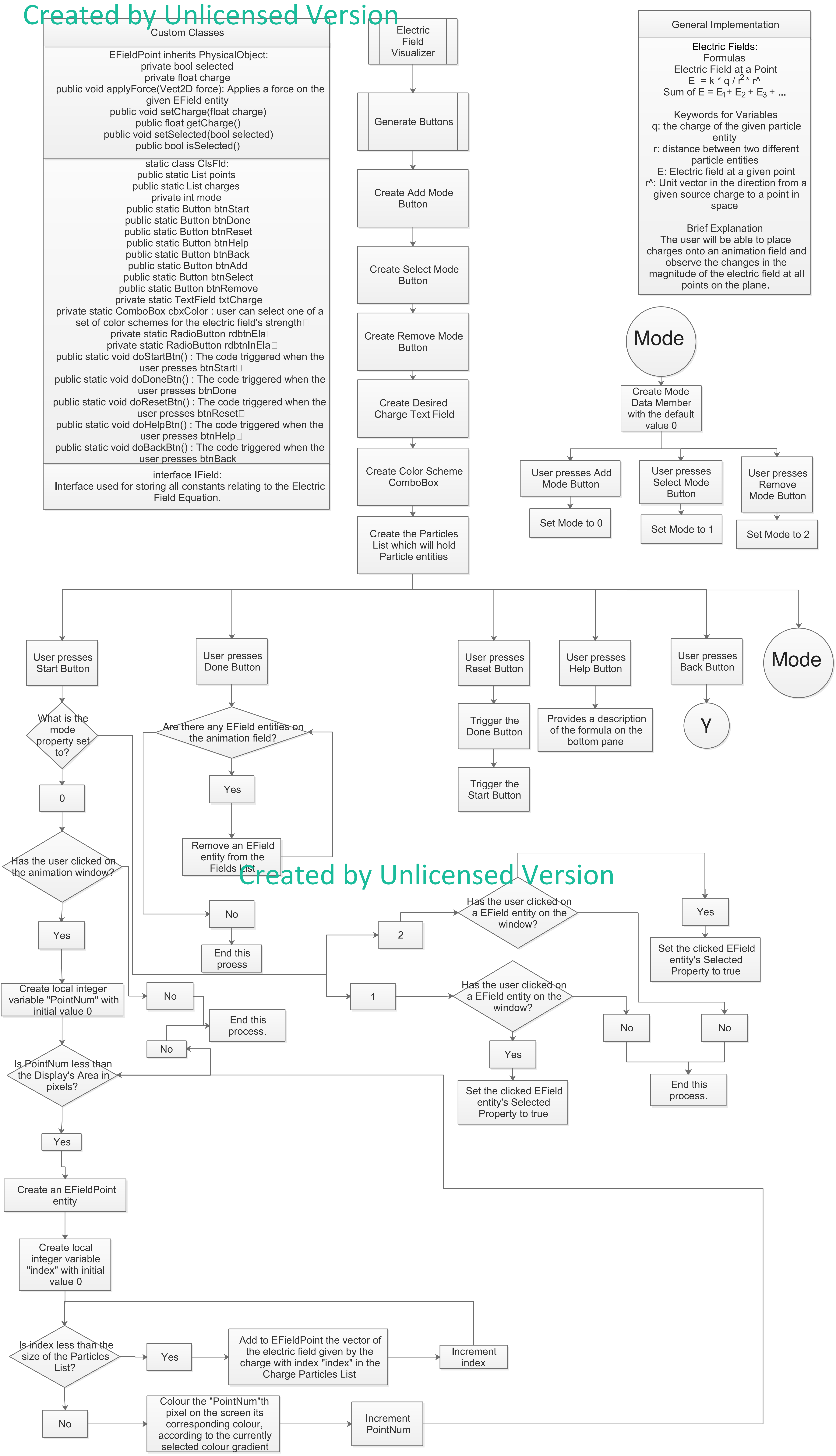
Electrostatic Forces:
Formulas
Coulomb's Law
$F_e= k * q_1 * q_2 / r^2$
$F_e = m_e * a$

Keywords for Variables
q_x: the charge of the given particle entity
r: distance between two different particle entities
m_x: mass of x
a: acceleration of the given entity

Brief Explanation
The user will be able to place charges onto an animation field and observe the electrostatic forces reacting on each other.



Electrostatic Forces
Generate Buttons
Create Add Mode Button
Create Select Mode Button
Create Remove Mode Button
Create Desired Charge Text Field
Create Electrostatic Constant Text Field
Create the Particles List which will hold Particle entities
Mode
Create Mode Data Member with the default value 0
User presses Add Mode Button
Set Mode to 0
User presses Select Mode Button
Set Mode to 1
User presses Remove Mode Button
Set Mode to 2
User presses Start Button
User presses Start Button
Create Charge Label
Create Acceleration Label
Create Velocity Label
Create local integer variable "index" with initial value 0
User presses Done Button
Are there any Particle entities on the animation field?
Yes
Remove a charge entity from the Particles List
No
End this proess
User presses Reset Button
Trigger the Done Button
Trigger the Start Button
User presses Help Button
Provides a description of the formula on the bottom pane
User presses Back Button
Y
Mode
Apply Force
Calculate the net force vector on the given Particle entity by Coulomb's Law using the electrostatic constant text field as the value for k and the Particle entity's Charge property as well as every other Particle entity inside the Particles list's distance
Change the charge entity's Acceleration property based on its net force by Newton's Second Law
Set the charge entity's Velocity property to its current velocity plus the Acceleration property
Set the charge entity's Position property to its current position plus the Velocity property
Set the charge entity's location on screen to the Position property
Is index less than the size of the Particles List?
Yes
Apply Force on Particles at index
Is the given Particle entity's Selected property set to true?
Yes
Set the Charge Label to the Particle entity's Charge property
Set the Acceleration Label to the Particle entity's Acceleration property
Set the Acceleration Label to the Particle entity's Acceleration property
No
No
Has the user clicked on the animation window?
Yes
Is Mode set to 0?
Yes
Create a Particle entity and add it to the Particles list
No
Is there a Particle entity on the space the user clicked?
Yes
What is Mode set to?
2
Remove the Particle entity on the space from the Particles list
1
Set the Particle entity on the space's Selected Property to true
Set the "index" variable to 0
Is index less than the size of the Particles List?
Yes
Set the Particle entity at index's Selected Property to false
No



Custom Classes

EFieldPoint inherits PhysicalObject:
private bool selected
private float charge
public void applyForce(Vect2D force): Applies a force on the given EField entity
public void setCharge(float charge)
public float getCharge()
public void setSelected(bool selected)
public bool isSelected()

static class ClsFld:
public static List points
public static List charges
private int mode
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
public static Button btnAdd
public static Button btnSelect
public static Button btnRemove
private static TextField txtCharge
private static ComboBox cbxColor : user can select one of a set of color schemes for the electric field's strength
private static RadioButton rdbtnEla
private static RadioButton rdbtnInEla
public static void doStartBtn() : The code triggered when the user presses btnStart
public static void doDoneBtn() : The code triggered when the user presses btnDone
public static void doResetBtn() : The code triggered when the user presses btnReset
public static void doHelpBtn() : The code triggered when the user presses btnHelp
public static void doBackBtn() : The code triggered when the user presses btnBack

interface IField:
Interface used for storing all constants relating to the Electric Field Equation.

General Implementation

Electric Fields:
Formulas
Electric Field at a Point
$E = k * q / r^2 * r^\wedge$
Sum of E = $E_1 + E_2 + E_3 + \ldots$

Keywords for Variables
q: the charge of the given particle entity
r: distance between two different particle entities
E: Electric field at a given point
r^: Unit vector in the direction from a given source charge to a point in space

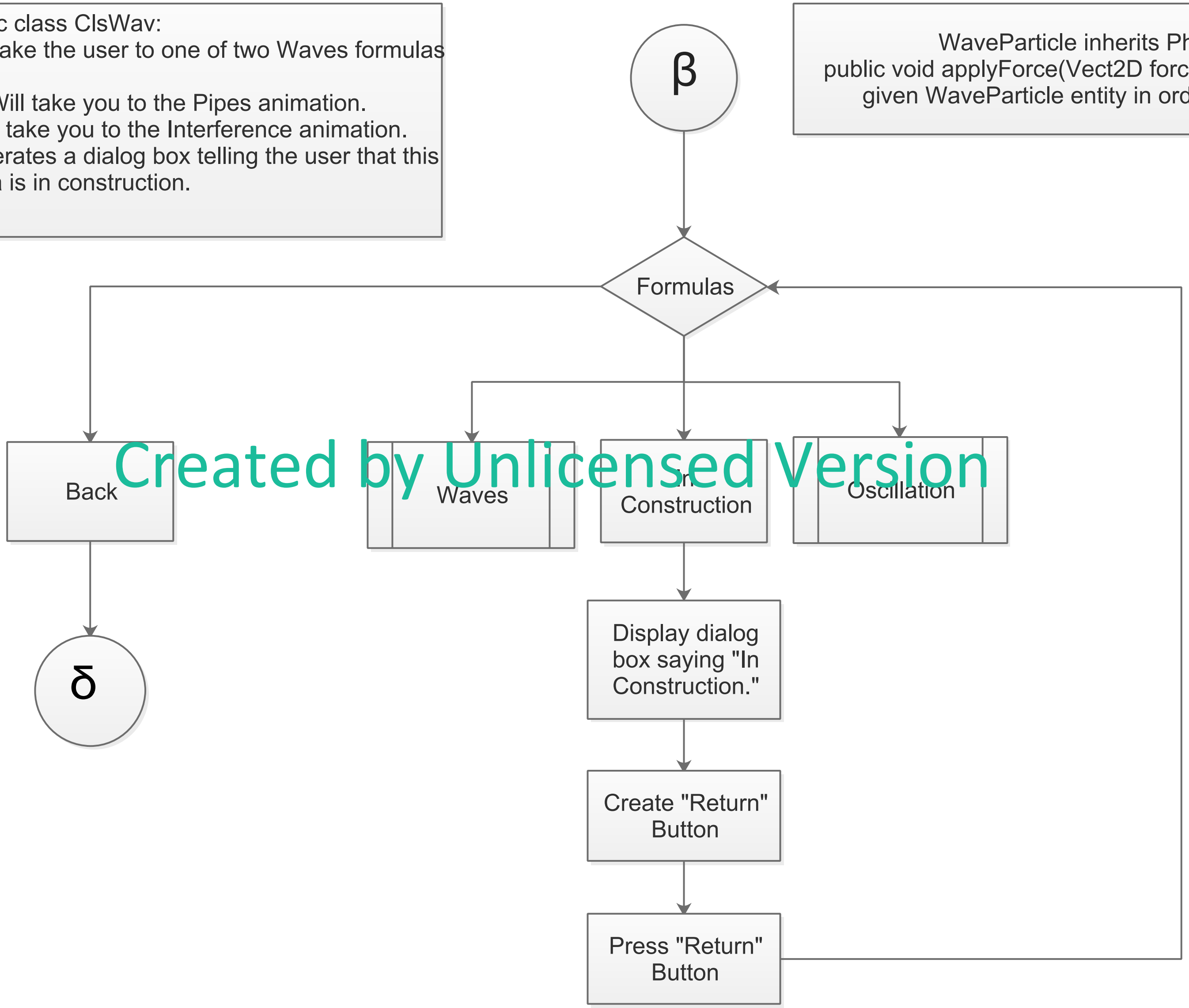
Brief Explanation
The user will be able to place charges onto an animation field and observe the changes in the magnitude of the electric field at all points on the plane.

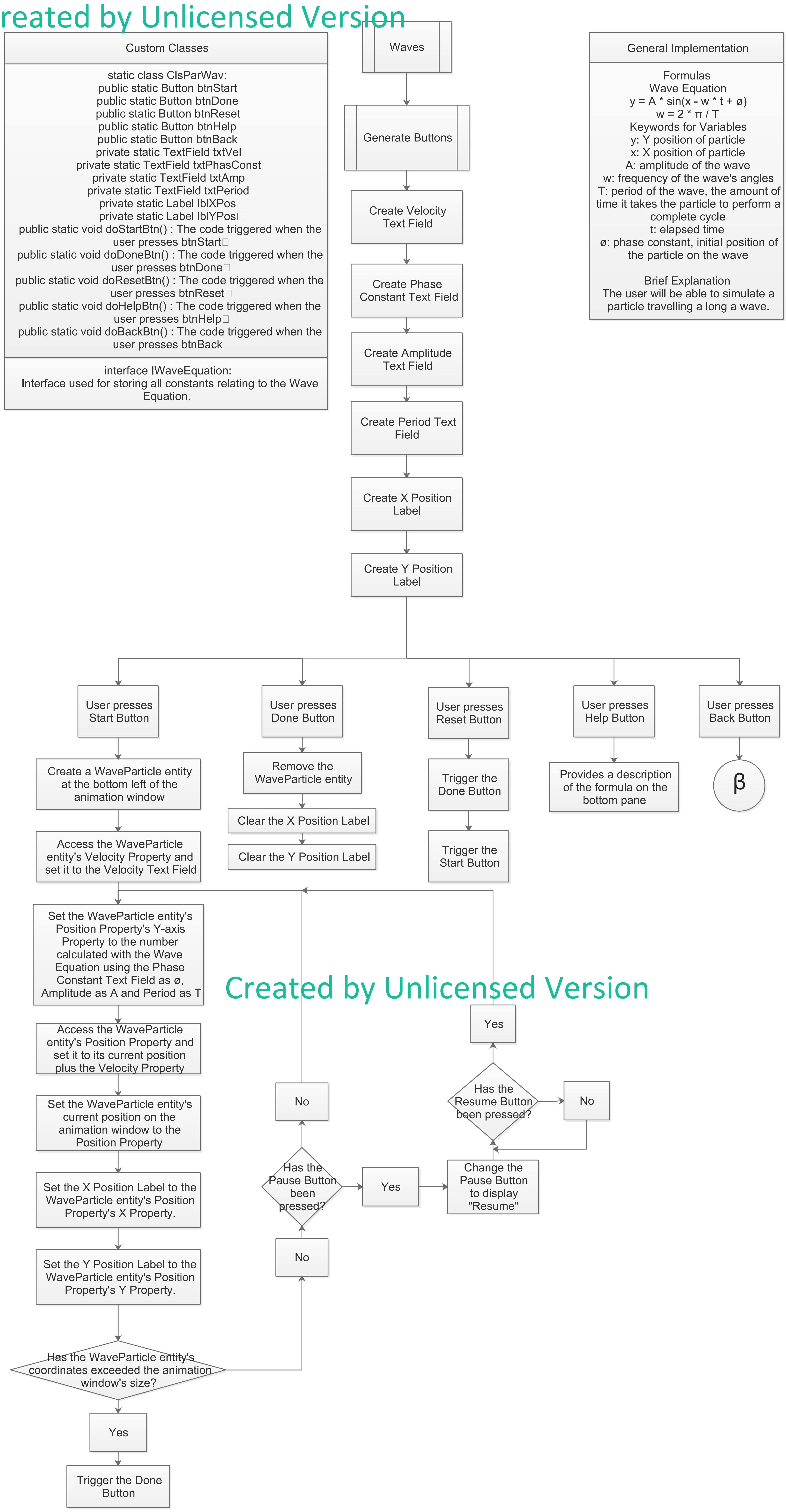
Electric Field Visualizer
Generate Buttons
Create Add Mode Button
Create Select Mode Button
Create Remove Mode Button
Create Desired Charge Text Field
Create Color Scheme ComboBox
Create the Particles List which will hold Particle entities
Mode
Create Mode Data Member with the default value 0
User presses Add Mode Button
Set Mode to 0
User presses Select Mode Button
Set Mode to 1
User presses Remove Mode Button
Set Mode to 2
User presses Start Button
What is the mode property set to?
0
Has the user clicked on the animation window?
Yes
No
Create local integer variable "PointNum" with initial value 0
No
End this process.
Is PointNum less than the Display's Area in pixels?
Yes
Create an EFieldPoint entity
Create local integer variable "index" with initial value 0
Is index less than the size of the Particles List?
Yes
Add to EFieldPoint the vector of the electric field given by the charge with index "index" in the Charge Particles List
Increment index
No
Colour the "PointNum"th pixel on the screen its corresponding colour, according to the currently selected colour gradient
Increment PointNum
User presses Done Button
Are there any EField entities on the animation field?
Yes
Remove an EField entity from the Fields List
No
End this prcess
User presses Reset Button
Trigger the Done Button
Trigger the Start Button
User presses Help Button
Provides a description of the formula on the bottom pane
User presses Back Button
Y
Mode
2
Has the user clicked on a EField entity on the window?
Yes
Set the clicked EField entity's Selected Property to true
1
Has the user clicked on a EField entity on the window?
Yes
Set the clicked EField entity's Selected Property to true
No
No
End this process.

static class ClsWav:
Class containing the menu to take the user to one of two Waves formulas

public Button btnPip : Will take you to the Pipes animation.
public Button btnInter : Will take you to the Interference animation.
public Button btnInCon : Generates a dialog box telling the user that this formula is in construction.

WaveParticle inherits PhysicalObject:
public void applyForce(Vect2D force): Applies a force on the given WaveParticle entity in order to shift its position





Custom Classes

static class ClsParWav:
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
private static TextField txtVel
private static TextField txtPhasConst
private static TextField txtAmp
private static TextField txtPeriod
private static Label lblXPos
private static Label lblYPos□
public static void doStartBtn() : The code triggered when the user presses btnStart□
public static void doDoneBtn() : The code triggered when the user presses btnDone□
public static void doResetBtn() : The code triggered when the user presses btnReset□
public static void doHelpBtn() : The code triggered when the user presses btnHelp□
public static void doBackBtn() : The code triggered when the user presses btnBack

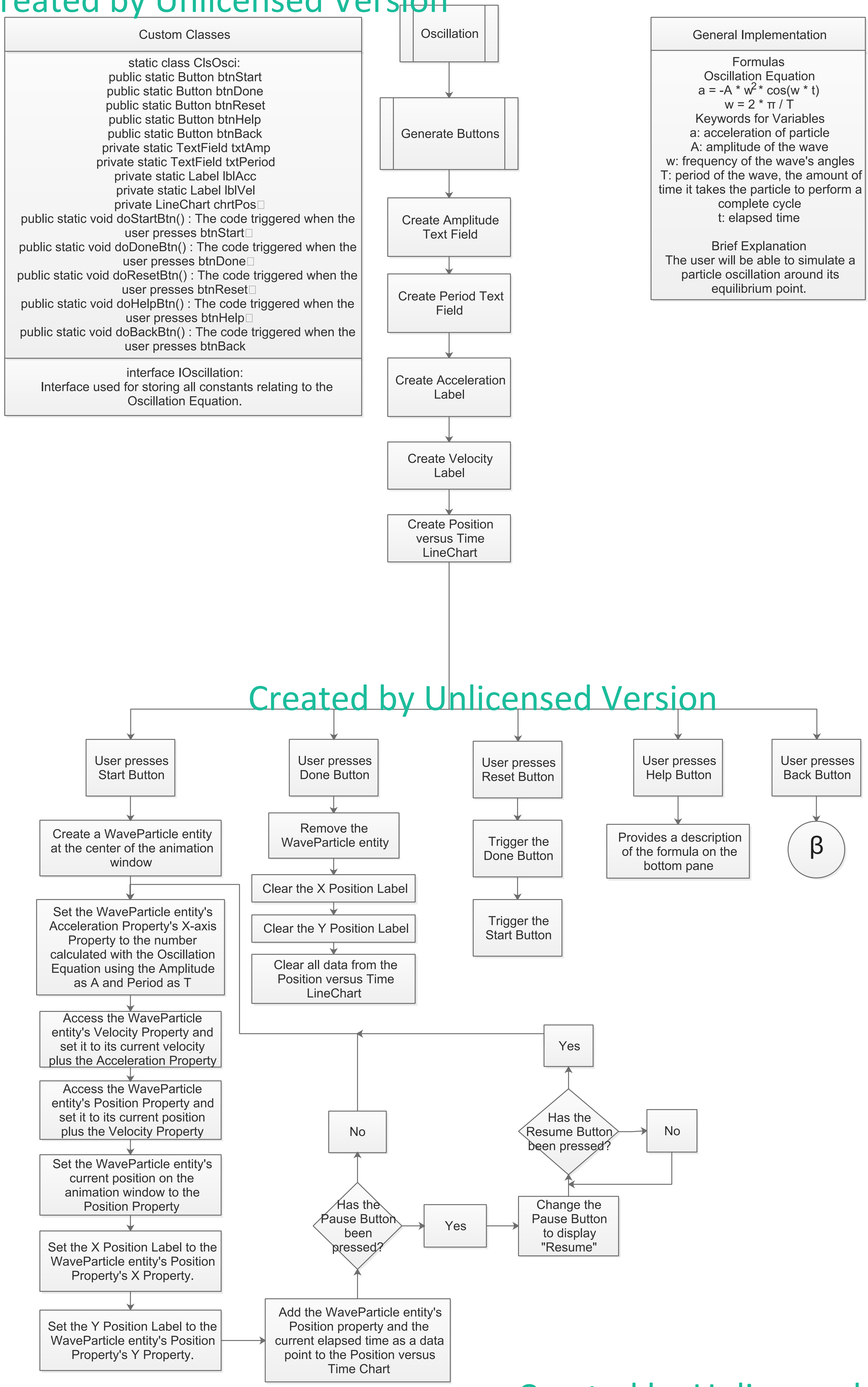
interface IWaveEquation:
Interface used for storing all constants relating to the Wave Equation.

General Implementation

Formulas
Wave Equation
y = A * sin(x - w * t + ø)
w = 2 * π / T
Keywords for Variables
y: Y position of particle
x: X position of particle
A: amplitude of the wave
w: frequency of the wave's angles
T: period of the wave, the amount of time it takes the particle to perform a complete cycle
t: elapsed time
ø: phase constant, initial position of the particle on the wave

Brief Explanation
The user will be able to simulate a particle travelling a long a wave.

Waves
Generate Buttons
Create Velocity Text Field
Create Phase Constant Text Field
Create Amplitude Text Field
Create Period Text Field
Create X Position Label
Create Y Position Label
User presses Start Button
User presses Done Button
User presses Reset Button
User presses Help Button
User presses Back Button
Create a WaveParticle entity at the bottom left of the animation window
Remove the WaveParticle entity
Trigger the Done Button
Provides a description of the formula on the bottom pane
β
Access the WaveParticle entity's Velocity Property and set it to the Velocity Text Field
Clear the X Position Label
Clear the Y Position Label
Trigger the Start Button
Set the WaveParticle entity's Position Property's Y-axis Property to the number calculated with the Wave Equation using the Phase Constant Text Field as ø, Amplitude as A and Period as T
Yes
Access the WaveParticle entity's Position Property and set it to its current position plus the Velocity Property
No
Has the Resume Button been pressed?
No
Set the WaveParticle entity's current position on the animation window to the Position Property
Has the Pause Button been pressed?
Yes
Change the Pause Button to display "Resume"
Set the X Position Label to the WaveParticle entity's Position Property's X Property.
No
Set the Y Position Label to the WaveParticle entity's Position Property's Y Property.
Has the WaveParticle entity's coordinates exceeded the animation window's size?
Yes
Trigger the Done Button



Custom Classes

static class ClsOsci:
public static Button btnStart
public static Button btnDone
public static Button btnReset
public static Button btnHelp
public static Button btnBack
private static TextField txtAmp
private static TextField txtPeriod
private static Label lblAcc
private static Label lblVel
private LineChart chrtPos□
public static void doStartBtn() : The code triggered when the user presses btnStart□
public static void doDoneBtn() : The code triggered when the user presses btnDone□
public static void doResetBtn() : The code triggered when the user presses btnReset□
public static void doHelpBtn() : The code triggered when the user presses btnHelp□
public static void doBackBtn() : The code triggered when the user presses btnBack

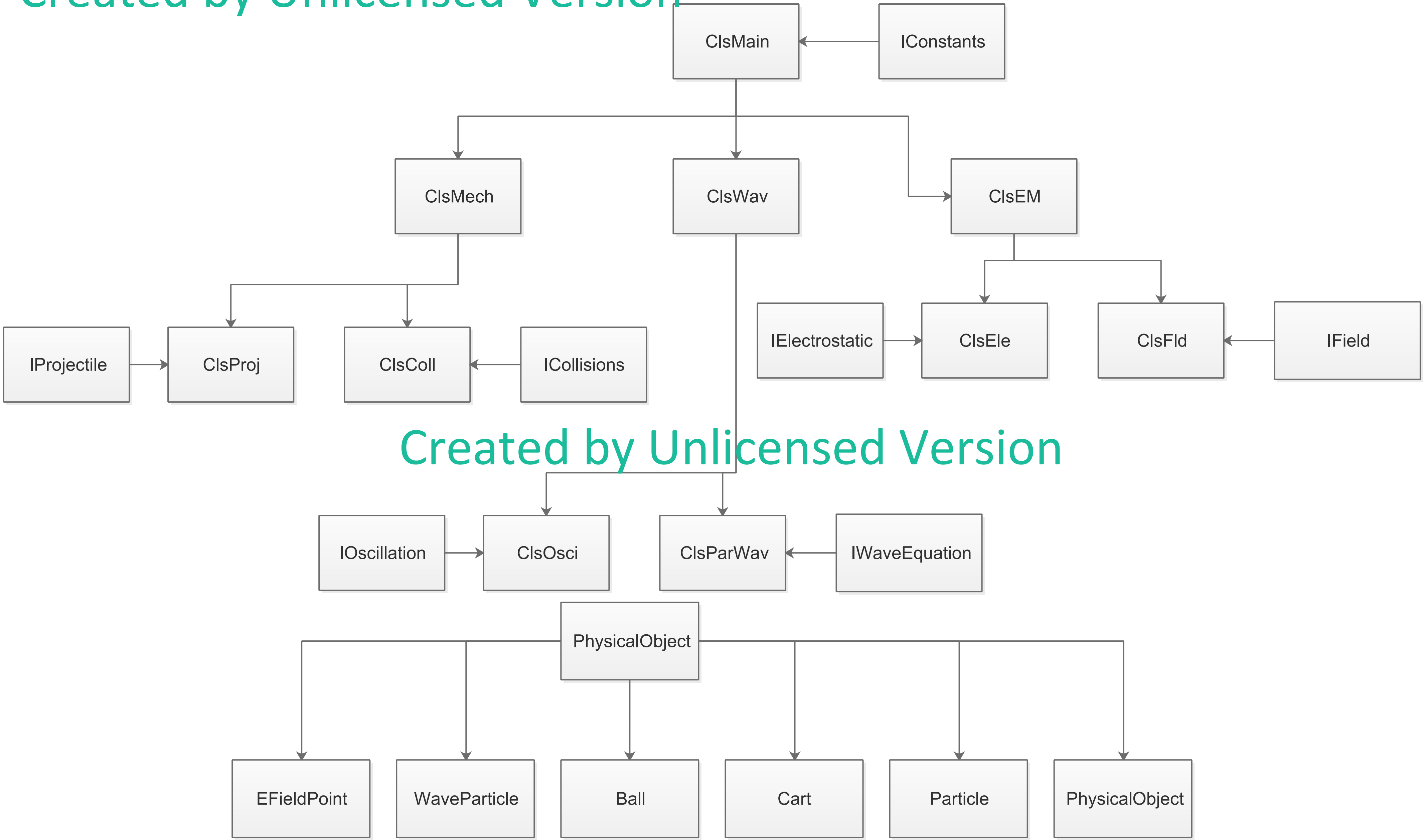
interface IOscillation:
Interface used for storing all constants relating to the Oscillation Equation.

General Implementation

Formulas
Oscillation Equation
$a = -A * w^2 * \cos(w * t)$
$w = 2 * \pi / T$
Keywords for Variables
a: acceleration of particle
A: amplitude of the wave
w: frequency of the wave's angles
T: period of the wave, the amount of time it takes the particle to perform a complete cycle
t: elapsed time

Brief Explanation
The user will be able to simulate a particle oscillation around its equilibrium point.

Oscillation
Generate Buttons
Create Amplitude Text Field
Create Period Text Field
Create Acceleration Label
Create Velocity Label
Create Position versus Time LineChart
User presses Start Button
User presses Done Button
User presses Reset Button
User presses Help Button
User presses Back Button
Create a WaveParticle entity at the center of the animation window
Remove the WaveParticle entity
Trigger the Done Button
Provides a description of the formula on the bottom pane
β
Clear the X Position Label
Trigger the Start Button
Clear the Y Position Label
Set the WaveParticle entity's Acceleration Property's X-axis Property to the number calculated with the Oscillation Equation using the Amplitude as A and Period as T
Clear all data from the Position versus Time LineChart
Access the WaveParticle entity's Velocity Property and set it to its current velocity plus the Acceleration Property
Yes
Access the WaveParticle entity's Position Property and set it to its current position plus the Velocity Property
No
Has the Resume Button been pressed?
No
Set the WaveParticle entity's current position on the animation window to the Position Property
Has the Pause Button been pressed?
Yes
Change the Pause Button to display "Resume"
Set the X Position Label to the WaveParticle entity's Position Property's X Property.
Set the Y Position Label to the WaveParticle entity's Position Property's Y Property.
Add the WaveParticle entity's Position property and the current elapsed time as a data point to the Position versus Time Chart


ClsMain
IConstants
ClsMech
ClsWav
ClsEM
IProjectile
ClsProj
ClsColl
ICollisions
IElectrostatic
ClsEle
ClsFld
IField
IOscillation
ClsOsci
ClsParWav
IWaveEquation
PhysicalObject
EFieldPoint
WaveParticle
Ball
Cart
Particle
PhysicalObject