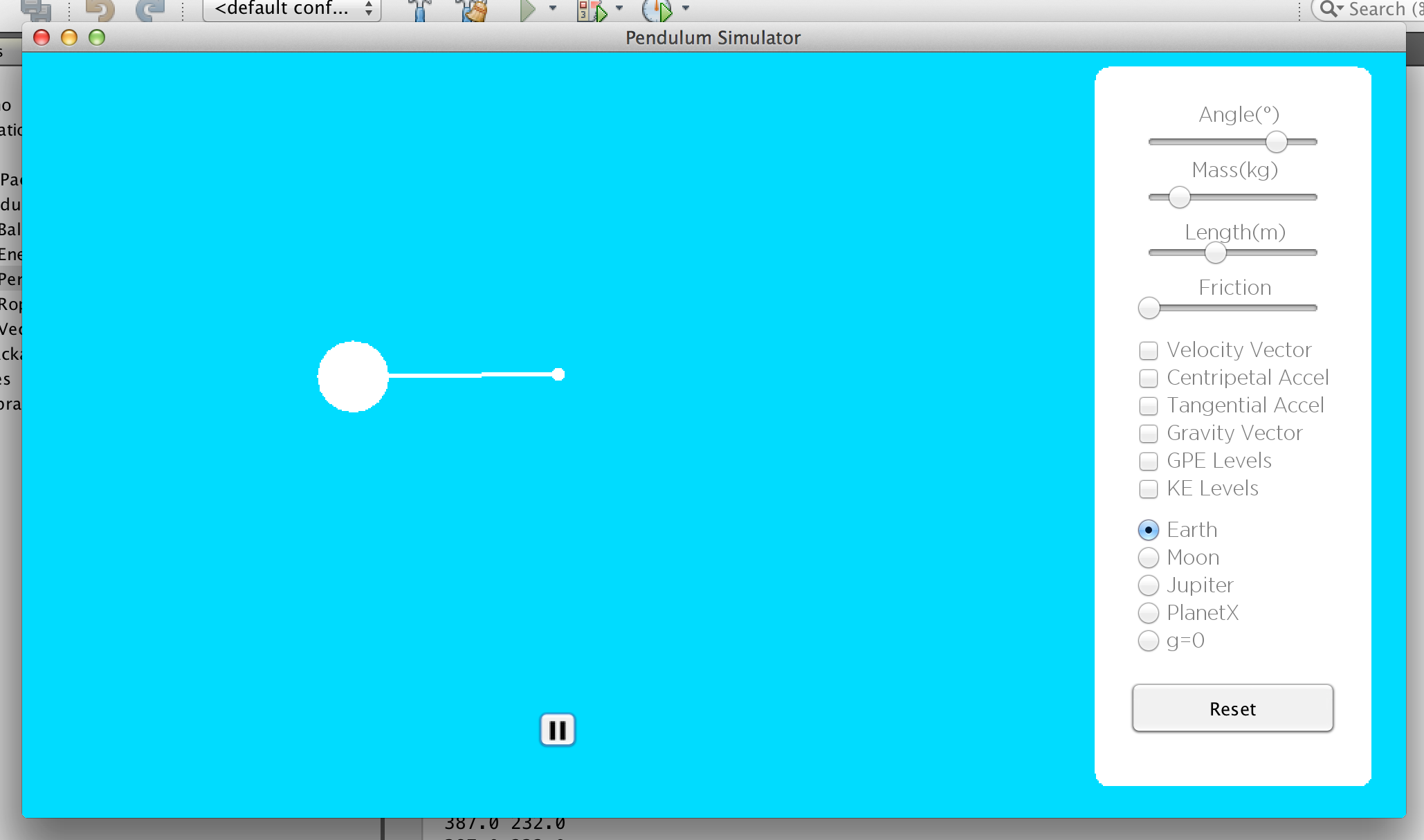
Design Document

# **Description of the program**

My program is a physics applet that will take advantage of Java graphics to simulate the motion of a simple pendulum. The simulation will follow the laws of physics, under the following assumptions and conditions:

* Energy is conserved
* Air resistance is negligible
* The surface of the rod and the pivot are frictionless
* The mass of a ball is proportional to its radius

The user will be able to interact by changing various properties that affect the motion of a pendulum. Some examples of those properties include the length, the mass, the starting angle and friction. The user will be able to make these adjustments by using integrated buttons and sliders. The program will be created with the experience of the user in mind and aesthetics will be taken into consideration. Furthermore, this program will display various vectors that are associated with the motion of a pendulum. An example would be the velocity vector of the ball at the end of the pendulum. Moreover, it will also display the potential and kinetic energy levels of the pendulum.

**Inputs to the program**

* Starting angle, length and mass of the pendulum
* The gravitational and damping constant
* The radius of the ball
* The color of the rod,ball,pivot,vectors and energy levels
* The speed of the animation

|  |  |
| --- | --- |
| **Fields** | |
| double GPE | The value of the gravitational potential energy |
| double KE | The value of the kinetic energy |
| double height | The value of the height of the pendulum |
| double velocity | The velocity of the pendulum |
| double mass | The mass of the pendulum |
| double gravity | The value of gravity |
| double scale | The scale for the GPE and KE values |
| Color color | The color of the energy levels |

|  |  |
| --- | --- |
| **Methods** | |
| public Energy(Ball b, Color c, double g) | Constructor. Sets the color, mass, and gravity using the arguments. |
| public void calculateGPE(Rope r, Ball b) | Calculates the GPE of the pendulum using various fields from the Ball and Rope class |
| public void calculateKE(Rope r, Ball b) | Calculates the KE of the pendulum |
| public void updateMass(Ball b) | Updates the mass, as it changes throughout the program |
| public void drawGPELevel(Graphics 2Dg, int w, int h) | Draws the GPE levels based on the window height and width |
| public void drawKELevel(Graphics 2Dg, int w, int h) | Draws the KE levels based on the window height and width |

# **Classes**

## **Ball class**

|  |  |
| --- | --- |
| **Fields** | |
| double xPos, yPos | The current position of the centre of the ball |
| double radius, diameter | The radius and diameter of the ball |
| double mass | The mass of the ball, which I will treat as the diameter of the ball |
| Color color | The current colour of the ball |

|  |  |
| --- | --- |
| **Methods** | |
| public **Ball**( double x, double y, double r, double, Color c ) | Constructor. Sets all fields using arguments except for diameter, which is calculated from radius, and mass, which is based on the diameter. |
| public void **updatePosition**(Rope r) | Sets (xPos, yPos) equal to the (xEnd,yEnd) of the rope |
| public void **updateDiameterAndMass**() | Updates the diameter using the radius, as it changes throughout the program. Updates mass using the updated diameter |
| public void **drawBall**(Graphics g) | Draws the ball on the screen based on its radius,xPos,yPos and color |

## **Rope class**

|  |  |
| --- | --- |
| **Fields** | |
| double xStart,yStart | The starting x and y positions of the rope |
| double xEnd,yEnd | The ending x and y positions of the rope |
| double initialAngle | The starting angle of the rope |
| double angle | The current angle of the rope |
| double angleVel | The angular velocity of the pendulum |
| double angleAccel | The angular acceleration of the pendulum |
| double length | The length of the pendulum |
| double dampingConstant | Controls the affect of friction on the pendulum |
| Color color | The color of the rope |

|  |  |
| --- | --- |
| **Methods** | |
| public **Rope**( Ball b, double a, double l, Color c, double d) | Constructor. Sets all fields using arguments. The xPos and yPos fields of the Ball class are used to initialize the xStart and yStart of the rope. |
| public void **setEndPosition**( ) | Sets (xEnd,yEnd) based on the (xStart,yStart), length and the angle. |
| public void **calculateAccelAngle**(double g, Ball b ) | Calculates the angular acceleration based on gravity and friction. The Ball class is used because the equation for determining friction requires the mass of the ball |
| public void **calculateVelAngle**(double g, double t, Ball b ) | Calculates the angular velocity based on the angular acceleration and delta time. Calls the calculateAccelAngle method. |
| public void **updateAngle**(double g, double t, Ball b ) | Updates the angle based on the angular velocity. Calls the calculateVelAngle method. |
| public void **drawRope**(Graphics2D g) | Draws the rope based on starting and ending positions. |

**Vector class**

|  |  |
| --- | --- |
| **Fields** | |
| double velDirection | The direction of the velocity vector. |
| double centripetalAccelDirection | The direction of the centripetal acceleration vector |
| double tangentialAccelDirection | The direction of the tangential acceleration vector |
| double gravityDirection | The direction of the gravity vector |
| double velMagnitude | The magnitude of the velocity vector |
| double centripetalAccelMagnitude | The magnitude of the centripetal acceleration vector |
| double tangentialAccellMagnitude | The magnitude of the tangential acceleration vector |
| double gravityMagnitude | The magnitude of the gravity vector |
| double scale | The scale of all of the vector magnitudes |
| double phi | The angle of the arrow heads. Default set to 45 degrees |
| int barb | The length of the arrow heads |
| Color color | The color of the vector |

|  |  |
| --- | --- |
| **Methods** | |
| publicVector(double g, Color c) | Constructor. Sets the color of the vector and sets the value of gravity |
| public void **calculateVelMagnitude**(Rope r) | Calculates the magnitude of the velocity vector using some of the fields of the Rope class |
| public void **calculateVelDirection**(Rope r) | Calculates the direction of the velocity vector |
| public void **calculateCentripetalAccelMagnitude**(Rope r) | Calculates the magnitude of the centripetal acceleration vector |
| public void **calculateCentripetalAccelDirection**(Rope r) | Calculates the direction of the centripetal acceleration vector |
| public void **calculateTangentialAccelMagnitude**(Rope r) | Calculates the magnitude of the tangential acceleration vector |
| public void **calculateTangentialAccelDirection**(Rope r) | Calculates the direction of the tangential acceleration vector |
| public void **gravityMagnitude**(Rope r) | Calculates the magnitude of the gravity vector |
| public void **gravityDirection**(Rope r) | Calculates the direction of the gravity vector |
| public void **drawVelVector**(Rope r, Graphics2D g) | Draws the velocity vector with arrow heads. Calls the drawArrowHead method. |
| public void **drawCentripetalAccelVector**(Rope r, Graphics2D g) | Draws the centripetal acceleration vector with arrow heads. Calls the drawArrowHead method. |
| public void **drawTangentialAccelVector**(Rope r, Graphics2D g) | Draws the tangential acceleration vector with arrow heads. Calls the drawArrowHead method. |
| public void **drawGravityVector**(Rope r, Graphics2D g) | Draws the gravity vector with arrow heads. Calls the drawArrowHead method. |
| public void **drawArrowHead**(Graphics2D g, Point tip, Point tail) | Draws the arrow head of any vector based on the coordinates of its tip and tail. |

**Energy class**

**Main class**

|  |  |
| --- | --- |
| **Fields** | |
| double startingAngle | The Starting Angle of the pendulum |
| double ropeLength | The length of the rope of the pendulum |
| double ball radius | The radius of the ball |
| double gravitaionalConstant | The value of gravity |
| double deltaTime | Controls the speed of the animation |
| double dampingConstant | The value of the constant that affects Friction |

|  |  |
| --- | --- |
| **Methods** | |
| public void **paint(**Graphics g) | Draws the ball,rope,pivot,vectors and energy levels to the screen |
| public void **swing**() | The main algorithm. It does all the calculations necessary to simulate the motion of a pendulum. For example: updates the angle of the pendulum, calculates the magnitudes for the vectors and calculates the magnitude of the energy levels. This method has an infinite while loop, so the pendulum keeps swinging forever. |
| public void **setVariable**() | Sets the variables based on what the user inputs and redefines the objects based on those values |
| public void **resetVariable**s() | Sets the variables to their initial conditions |
| public static void main(String[]args) | Initializes the window. Sets up the JFrame. Calls the swing method. |
| public void sleep(int duration) | Pauses the thread for whatever time is passed to it |

# **Run-time analysis of the program**

The main algorithm does these steps

**(1) Initialize the window**.

This takes constant time **O(1)** because it does not depend on anything other than the CPU.

**(2) Set the initial conditions for the pendulum**

This takes **O(1)** time because it does not depend on anything other than the CPU.

**(3) Run the animation**

This takes **O(1)** time because the animation runs in an infinite while loop. The infinite while loop runs in constant time because it is not deponent upon anything.

While(true){

updateAngle

repaint

}

**(4) Reset and set variables**

This takes **O(1)** time because setting the variables to their initial conditions or to the conditions that the user specifies does not deponent upon anything.

**Total for the three parts**

**O(1) + O(1) + O(1) + O(1) = O(1)** because **O(1)** is the dominant term.