Allegro Animations

1.0

Generated by Doxygen 1.8.15

Chapter 1

Bug List

File Updateable.h
: no known bugs

File Vector.h
: No known bugs

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 _line Struct Reference

represents a straight line from the Point object _start to the Point object _end with a certain slope

```
#include <Line.h>
```

Public Member Functions

- **_line** (Point a, Point b)
- double length ()

  finds the scalar distance between the two points
- double get_angle_ccw (double change_in_angle)

  calculates the new angle (in radians)
- double get_angle_cw (double change_in_angle)

  calculates the new angle (in radians)
- Point get_endpoint_ccw (double change_in_angle, double new_length)

  calculates the position of the endpoint for the new line with some change in angle
- Point get_endpoint_cw (double change_in_angle, double new_length)

  calculates the position of the endpoint for the new line with some change in angle

Public Attributes

- Point _start
- Point _end

5.1.1 Detailed Description

represents a straight line from the Point object _start to the Point object _end with a certain slope

Definition at line 26 of file Line.h.

5.1.2 Member Function Documentation

5.1.2.1 get_angle_ccw()

```
double _line::get_angle_ccw (
            double change_in_angle )  [inline]
```

calculates the new angle (in radians)

Parameters

change_in_angle	the change in angle (added; counter-clockwise change)

Returns

double value representing radians in the range [-PI, PI]

Definition at line 55 of file Line.h.

```
                                                                 {
        return -(atan2(_start.y - _end.y, _end.x - _start.x) + change_in_angle);
    }
```

5.1.2.2 get_angle_cw()

```
double _line::get_angle_cw (
              double change_in_angle )  [inline]
```

calculates the new angle (in radians)

Parameters

change_in_angle	the change in angle (subtracted; clockwise change)

Returns

double value representing radians in the range [-PI, PI]

Definition at line 65 of file Line.h.

```
                                                                {
        return -(atan2(_start.y - _end.y, _end.x - _start.x) - change_in_angle);
    }
```

5.1.2.3 get_endpoint_ccw()

```
Point _line::get_endpoint_ccw (
              double change_in_angle,
              double new_length )  [inline]
```

calculates the position of the endpoint for the new line with some change in angle

Parameters

change_in_angle	the change in angle (radians)
new_length	the straight line distance from start to end for the new line

Returns

Point object

Definition at line 78 of file Line.h.

```
                                                                              {
    return Point(_end.x - new_length * cos(get_angle_ccw(change_in_angle)),
                 _end.y - new_length * sin(get_angle_ccw(change_in_angle)));
  }
```

5.1.2.4 get_endpoint_cw()

```
Point _line::get_endpoint_cw (
             double change_in_angle,
             double new_length )  [inline]
```

calculates the position of the endpoint for the new line with some change in angle

Parameters

change_in_angle	the change in angle (radians)
new_length	the straight line distance from start to end for the new line

Returns

Point object

Definition at line 92 of file Line.h.

```
                                                                             {
    return Point(_end.x - new_length * cos(get_angle_cw(change_in_angle)),
                 _end.y - new_length * sin(get_angle_cw(change_in_angle)));
  }
```

5.1.2.5 length()

```
double _line::length ( )  [inline]
```

finds the scalar distance between the two points

Parameters

none	

Returns

value of type double

Definition at line 44 of file Line.h.

```
                  {
    return sqrt(pow(_end.y - _start.y, 2) + pow(_end.x - _start.x, 2));
  }
```

5.1.3 Member Data Documentation

5.1.3.1 _end

`Point _line::_end`

represents the end point

Definition at line 33 of file Line.h.

5.1.3.2 _start

`Point _line::_start`

represents the start point

Definition at line 33 of file Line.h.

The documentation for this struct was generated from the following file:

- src/Line.h

5.2 Display Class Reference

Class to initialize allegro and open the main window.

```
#include <Display.h>
```

Public Member Functions

- Display (int w=800, int h=600)

  Initializes allegro and constructs a window of given size.
- ∼Display ()

  Frees allegro resources.
- int getW () const

  Returns the width of the window.
- int getH () const

  Returns the height of the window.
- ALLEGRO_DISPLAY ∗ getAllegroDisplay () const

  Returns the Allegro display.

5.2.1 Detailed Description

Class to initialize allegro and open the main window.

Class Display initializes allegro and the primitives add-on in the constructor. A single object must be instantiated prior to attempting any kind of drawing. The instantiated object can be used to retrieve the dimensions of the window.

Definition at line 22 of file Display.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Display()

```
Display::Display (
              int w = 800,
              int h = 600 )
```

Initializes allegro and constructs a window of given size.

Construct a new Display:: Display object.

Exactly one object must be created before any allegro functions can be used. Both allegro and the primitives add-on (for drawing) are initialized, and a window is displayed.

Parameters

w	the width of the window displayed in pixels
h	the height of the window displayed in pixels
w	width of the display window
h	height of the display window

Definition at line 23 of file Display.cc.

```
                                {
  width = w;
  height = h;

  al_init();

  // if the display cannot be initialized, we should throw an
  // exception. We will deal with exceptons later in the course, so
  // for now, we simply exit
  if ((display = al_create_display(width, height)) == NULL) {
    std::cerr « "Cannot initialize the display" « std::endl;
    exit(1);  // non-zero argument means "trouble"
  }

  al_init_primitives_addon();
}
```

5.2.2.2 ∼Display()

```
Display::∼Display ( )
```

Frees allegro resources.

Destroy the Display:: Display object.

The allegro window is closed and the allegro resources are freed. Drawing is not possible afterwards.

Definition at line 44 of file Display.cc.

```
                  {
  al_destroy_display(display);
}
```

5.2.3 Member Function Documentation

5.2.3.1 getAllegroDisplay()

`ALLEGRO_DISPLAY* Display::getAllegroDisplay ( ) const [inline]`

Returns the Allegro display.

\ret a pointer to the Allegro display structure that can be passed to allegro functions requiring an Allegro display argument

Definition at line 61 of file Display.h.

```
{ return display; }
```

5.2.3.2 getH()

`int Display::getH ( ) const [inline]`

Returns the height of the window.

\ret the height of the window in pixels

Definition at line 54 of file Display.h.

```
{ return height; };
```

5.2.3.3 getW()

`int Display::getW ( ) const [inline]`

Returns the width of the window.

\ret the width of the window in pixels

Definition at line 48 of file Display.h.

```
{ return width; };
```

The documentation for this class was generated from the following files:

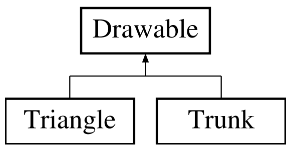
- src/Display.h
- src/Display.cc

5.3 Drawable Class Reference

interface for drawable objects

```
#include <Drawable.h>
```

Inheritance diagram for Drawable:



Public Member Functions

- virtual void draw ()=0

  virtual function to draw derived objects

5.3.1 Detailed Description

interface for drawable objects

provides the declaration of the draw function

Definition at line 10 of file Drawable.h.

The documentation for this class was generated from the following file:

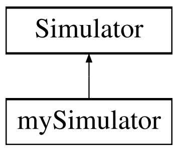
- src/Drawable.h

5.4 mySimulator Class Reference

contains lists of Drawable objects and Updateable objects and calls draw and update for their derived objects respectively

```
#include <mySimulator.h>
```

Inheritance diagram for mySimulator:



Public Member Functions

- mySimulator (const Display &d, int fps)

  Constructor.
- void addDrawable (std::shared_ptr< Drawable > p)

  takes a smart pointer to Drawable object and pushes it onto the toDraw list
- void addUpdateable (std::shared_ptr< Updateable > p)

  takes a pointer to Updateable object and pushes it onto the to Update list
- void updateModel (double dt)

  takes some amount of time dt and calls each Updateable object's update function
- void drawModel ()

  iteratively calls each Drawable object's draw function

5.4.1 Detailed Description

contains lists of Drawable objects and Updateable objects and calls draw and update for their derived objects respectively

Definition at line 28 of file mySimulator.h.

5.4.2 Member Function Documentation

5.4.2.1 addDrawable()

```
void mySimulator::addDrawable (
            std::shared_ptr< Drawable > p )  [inline]
```

takes a smart pointer to Drawable object and pushes it onto the toDraw list

Parameters

p	the pointer to the Drawable object

Definition at line 62 of file mySimulator.h.

```
62 { toDraw.push_back(p); }
```

5.4.2.2 addUpdateable()

```
void mySimulator::addUpdateable (
            std::shared_ptr< Updateable > p )  [inline]
```

takes a pointer to Updateable object and pushes it onto the to Update list

Parameters

p	the pointer to the Updateable object

Definition at line 69 of file mySimulator.h.

```
{ toUpdate.push_back(p); }
```

5.4.2.3 updateModel()

```
void mySimulator::updateModel (
            double dt )  [inline], [virtual]
```

takes some amount of time dt and calls each Updateable object's update function

Parameters

dt	the amount of time passed since the last update occurred

Implements Simulator.

Definition at line 76 of file mySimulator.h.

```
                              {
    for (std::list<std::shared_ptr<Updateable»::iterator it = toUpdate.begin();
         it != toUpdate.end(); ++it)
      (*it)->update(dt);
  }
```

The documentation for this class was generated from the following file:

- src/mySimulator.h

5.5 Point Struct Reference

represents a position on the display (grid)

`#include <Point.h>`

Public Member Functions

- Point (double a=0.0, double b=0.0)

  Constructor.
- Point operator+ (Vector v)

  adds the value of member x of v to this objects member x, and adds the value of member y of v to this objects member y

Public Attributes

- double **x**
- double **y**

5.5.1 Detailed Description

represents a position on the display (grid)

Definition at line 20 of file Point.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Point()

```
Point::Point (
              double a = 0.0,
              double b = 0.0 )  [inline]
```

Constructor.

Parameters

Definition at line 27 of file Point.h.

```
27 : x(a), y(b){};
```

5.5.3 Member Function Documentation

5.5.3.1 operator+()

```
Point Point::operator+ (
              Vector v )  [inline]
```

adds the value of member x of v to this objects member x, and adds the value of member y of v to this objects member y

Parameters

v	represents the change in position for this Point object

Returns

Point

Definition at line 35 of file Point.h.

```
35 { return Point(x + v.x, y + v.y); }
```

The documentation for this struct was generated from the following file:

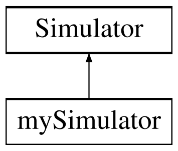
- src/Point.h

5.6 Simulator Class Reference

Simulator object. Sets up Allegro library, and runs the main simulation loop.

```
#include <Simulator.h>
```

Inheritance diagram for Simulator:



Public Member Functions

- Simulator (const Display &d, int fps)

  event storage
- ∼Simulator ()

  Destroy the Simulator object, free all Allegro resources allocated by constructor.
- void run ()

  Invoke to begin the simulation. Main rendering loop.
- virtual void updateModel (double dt)=0

  Updates the state of the objects in the model.
- virtual void drawModel ()=0

  Draws the model to the display.

5.6.1 Detailed Description

Simulator object. Sets up Allegro library, and runs the main simulation loop.

Definition at line 21 of file Simulator.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Simulator()

```
Simulator::Simulator (
              const Display & d,
              int fps )
```

event storage

Construct a new Simulator object, and initialize the Allegro library.

Parameters

d	Display object.
fps	Frames per second.

Definition at line 18 of file Simulator.cc.

```
    : framesPerSec(fps), timer(NULL), eventQueue(NULL) {
  if ((timer = al_create_timer(1.0 / fps)) == NULL)
    throw std::runtime_error("Cannot create allegro timer");

  if ((eventQueue = al_create_event_queue()) == NULL)
    throw std::runtime_error("Cannot create event queue");

  al_register_event_source(eventQueue,
                           al_get_display_event_source(d.getAllegroDisplay()));

  al_register_event_source(eventQueue, al_get_timer_event_source(timer));

  al_start_timer(timer);
}
```

5.6.2.2 ∼Simulator()

```
Simulator::∼Simulator ( )
```

Destroy the Simulator object, free all Allegro resources allocated by constructor.

Destroy the Simulator:: Simulator object, and clean up resources for timer and eventQueue.

Definition at line 39 of file Simulator.cc.

```
                        {
  if (timer != NULL) al_destroy_timer(timer);
  if (eventQueue != NULL) al_destroy_event_queue(eventQueue);
}
```

5.6.3 Member Function Documentation

5.6.3.1 drawModel()

```
virtual void Simulator::drawModel ( )  [pure virtual]
```

Draws the model to the display.

Implemented in mySimulator.

5.6.3.2 run()

```
void Simulator::run ( )
```

Invoke to begin the simulation. Main rendering loop.

Run the simulator.

Definition at line 47 of file Simulator.cc.

```
                     {
  // switch to trigger model drawing
  bool redraw = true;
  // current time and previous time in seconds; needed so we can try
  // to keep track of the passing of real time.
  double currentTime, previousTime = 0;

  while (1) {
    ALLEGRO_EVENT ev;
    al_wait_for_event(eventQueue, &ev);

    if (ev.type == ALLEGRO_EVENT_TIMER) {
      currentTime = al_current_time();
      updateModel(currentTime - previousTime);
      previousTime = currentTime;
      // instead of simply calling drawModel() here, we set this flag so that
      // we redraw only if the event queue is empty; reason: draw is
      // expensive and we don't want to delay everything too much
      redraw = true;
    } else if (ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE) {
      break;
    }

    if (redraw && al_is_event_queue_empty(eventQueue)) {
      drawModel();
      redraw = false;
    }
  }
}
```

5.6.3.3 updateModel()

```
virtual void Simulator::updateModel (
               double dt )  [pure virtual]
```

Updates the state of the objects in the model.

Parameters

dt	Change in time, in seconds, since last update.

Implemented in mySimulator.

The documentation for this class was generated from the following files:

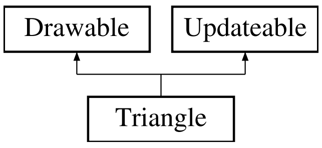
- src/Simulator.h
- src/Simulator.cc

5.7 Triangle Class Reference

represents a triangle that falls from the top of the display to the bottom

```
#include <Triangle.h>
```

Inheritance diagram for Triangle:



Public Member Functions

- Triangle (int x, int y)

  Construct a new Triangle object.
- void draw ()

  draws the triangle object to the display, if out of bounds returns to the top
- void update (double dt)

  Updates the triangle's position over time, i.e. $pt = pt + crtSpeed * dt$.

5.7.1 Detailed Description

represents a triangle that falls from the top of the display to the bottom

Definition at line 29 of file Triangle.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Triangle()

```
Triangle::Triangle (
              int x,
              int y )   [inline]
```

Construct a new Triangle object.

Parameters

x	Initial x-coordinate for the Triangle object.
y	Initial y-coordinate for the Triangle object.

Definition at line 57 of file Triangle.h.

```
                          : max_x(x), max_y(y), size(30), crtSpeed(0, 100) {
    pt = Point(rand() % x, rand() % y);
  }
```

5.7.3 Member Function Documentation

5.7.3.1 update()

```
void Triangle::update (
            double dt )  [inline], [virtual]
```

Updates the triangle's position over time, i.e. $pt = pt + crtSpeed * dt$.

Parameters

dt	Change in time, in seconds, since last update.

Implements Updateable.

Definition at line 80 of file Triangle.h.

```
80 { pt = pt + crtSpeed * dt; }
```

The documentation for this class was generated from the following file:

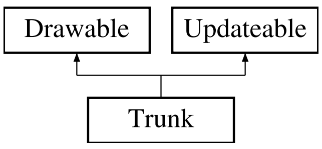
- src/Triangle.h

5.8 Trunk Class Reference

represents an elogating line from Point start to Point end in total_time

`#include <Trunk.h>`

Inheritance diagram for Trunk:



Public Member Functions

- Trunk (Point p1, Point p2, double time, double bf, double a, int rd)

  Constructor.
- void addBranch ()

  called when Point current has reached one third of the distance from start to end randomly chooses to add between 1-4 child Trunk objects to the branch list
- void draw ()

  draws a straight line from start to current, then iteratively calls draw on each of the child Trunk objects in the branch container
- void update (double dt)

  updates the position of current according to the Trunk objects growth rate and time passed dt; updates the position of all the child Trunk objects in the branch container;

5.8.1 Detailed Description

represents an elogating line from Point start to Point end in total_time

Definition at line 33 of file Trunk.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Trunk()

```
Trunk::Trunk (
            Point p1,
            Point p2,
            double time,
            double bf,
            double a,
            int rd )  [inline]
```

Constructor.

Point current is initalized to Point start growth is initalized to the distance between start and end divided by the total time expanded is initalized to false

Definition at line 99 of file Trunk.h.

```
        : start(p1),
          end(p2),
          total_time(time),
          branch_factor(bf),
          angle(a),
          rec_depth(rd) {
      current = start;
      L = sqrt(pow(end.y - start.y, 2) + pow(end.x - start.x, 2));
      growth =
          Vector((end.x - start.x) / total_time, (end.y - start.y) / total_time);
      expanded = false;
    }
```

5.8.3 Member Function Documentation

5.8.3.1 addBranch()

```
void Trunk::addBranch ( )  [inline]
```

called when Point current has reached one third of the distance from start to end randomly chooses to add between 1-4 child Trunk objects to the branch list

child Trunk objects have the possibility to appear in 4 positions as follows: 1/3 of the max length, angled to the left 1/3 of the max length, angled to the right 1/6 of the max length, angled to the left 1/6 of the max length, angled to the right < position 1 : @1/3 L, angled to left of trunk

< position 2 : @1/6 L, angled to the right of the trunk

< position 3 : @1/6 L, angled to the left of the trunk

< position 4 : @1/3 L, angled to right of trunk

Definition at line 123 of file Trunk.h.

```
                       {
    Point current_2 = start + growth * (total_time / 6);

    Point end_extended = end + growth * (total_time / 3);
    Point end_extended_2 = end + growth * (total_time / 6);

    _line end_ext(current, end_extended);
    _line end_ext_2(current_2, end_extended_2);
    double radians = angle * PI / 180.0;

    Point p;
    std::vector<Point> pts;

    p = Point(current.x, current.y);
    pts.push_back(p);
    p = end_ext.get_endpoint_ccw(radians, branch_factor * L);
    pts.push_back(p);

    p = Point((current.x + start.x) / 2, (current.y + start.y) / 2);
    pts.push_back(p);
    p = end_ext_2.get_endpoint_cw(radians, branch_factor * L);
    pts.push_back(p);

    p = Point((current.x + start.x) / 2, (current.y + start.y) / 2);
    pts.push_back(p);
    p = end_ext_2.get_endpoint_ccw(radians, branch_factor * L);
    pts.push_back(p);

    p = Point(current.x, current.y);
    pts.push_back(p);
    p = end_ext.get_endpoint_cw(radians, branch_factor * L);
    pts.push_back(p);

    int branch_config = rand() % 4 + 1;
    std::vector<Point>::iterator it1 = pts.begin();
    std::vector<Point>::iterator it2 = pts.begin() + 1;

    for (int i = 0; i < branch_config; i++) {
      branch.push_back(std::make_shared<Trunk>(
          (*it1), (*it2), total_time, branch_factor, angle, rec_depth - 1));
      ++it1;
      ++it1;
      ++it2;
      ++it2;
    }
  }
```

5.8.3.2 update()

```
void Trunk::update (
             double dt )  [inline], [virtual]
```

updates the position of current according to the Trunk objects growth rate and time passed dt; updates the position of all the child Trunk objects in the branch container;

Implements Updateable.

Definition at line 198 of file Trunk.h.

```
                        {
    double start_to_end =
        sqrt(pow(start.x - end.x, 2) + pow(start.y - end.y, 2));
    double start_to_current =
        sqrt(pow(start.x - current.x, 2) + pow(start.y - current.y, 2));

    if (start_to_current < start_to_end) {
      current = current + growth * dt;
    } else {
      current = end;
    }

    if (!branch.empty()) {
      for (auto it = branch.begin(); it != branch.end(); ++it) {
        (*it)->update(dt);
      }
    }

    if (!expanded && rec_depth > 0) {
      start_to_current =
          sqrt(pow(start.x - current.x, 2) + pow(start.y - current.y, 2));
      if (start_to_current > (L / 3)) {
        addBranch();
        expanded = true;
      }
    }
  }
```

The documentation for this class was generated from the following file:

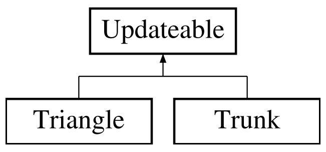
- src/Trunk.h

5.9 Updateable Class Reference

interface for updateable objects

`#include <Updateable.h>`

Inheritance diagram for Updateable:



Public Member Functions

- virtual void update (double t)=0

    virtual function declaration of update, updates the position of derived objects

5.9.1 Detailed Description

interface for updateable objects

provides the declaration of the update function

Definition at line 17 of file Updateable.h.

5.9.2 Member Function Documentation

5.9.2.1 update()

```
virtual void Updateable::update (
             double t )  [pure virtual]
```

virtual function declaration of update, updates the position of derived objects

Parameters

t	the change in time since the last position update

Implemented in Trunk, and Triangle.

The documentation for this class was generated from the following file:

- src/Updateable.h

5.10 Vector Struct Reference

represents the change in position of x and y

```
#include <Vector.h>
```

Public Member Functions

- **Vector** (double a=0.0, double b=0.0)
- Vector operator ∗ (double scalar)

  takes a scalar value and multiplies x and y by it

Public Attributes

- double **x**
- double **y**

5.10.1 Detailed Description

represents the change in position of x and y

Definition at line 18 of file Vector.h.

5.10.2 Member Function Documentation

5.10.2.1 operator ∗()

```
Vector Vector::operator * (
             double scalar )  [inline]
```

takes a scalar value and multiplies x and y by it

Parameters

scalar	

Returns

Vector

Definition at line 28 of file Vector.h.

```
{ return Vector(x * scalar, y * scalar); }
```

The documentation for this struct was generated from the following file:

- src/Vector.h

Chapter 6

File Documentation

6.1 src/Display.cc File Reference

Display window.

```
#include <allegro5/allegro_primitives.h>
#include <cstdlib>
#include <iostream>
#include "Display.h"
```

6.1.1 Detailed Description

Display window.

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.2 src/Display.h File Reference

```
#include <allegro5/allegro.h>
```

Classes

- class Display

  Class to initialize allegro and open the main window.

6.2.1 Detailed Description

Author

C. Barnson (`cbarnson@outlook.com`)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.3 src/Line.h File Reference

Represents a straight line in 2D.

```
#include <cmath>
#include "Point.h"
```

Classes

- struct _line

  represents a straight line from the Point object _start to the Point object _end with a certain slope

Variables

- const float **PI** = 3.14159265

6.3.1 Detailed Description

Represents a straight line in 2D.

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.4 src/main-line.cc File Reference

Program entry for the "tree" program.

```
#include <ctime>
#include <memory>
#include "Display.h"
#include "Point.h"
#include "Trunk.h"
#include "mySimulator.h"
```

Functions

- int **main** ()

6.4.1 Detailed Description

Program entry for the "tree" program.

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.5 src/main-triangle.cc File Reference

Program entry point for the "rain" program.

```
#include <cstdlib>
#include <ctime>
#include <list>
#include <memory>
#include "Display.h"
#include "Triangle.h"
#include "mySimulator.h"
```

Functions

- int **main** ()

6.5.1 Detailed Description

Program entry point for the "rain" program.

Author

C. Barnson (`cbarnson@outlook.com`)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.6 src/mySimulator.h File Reference

```
#include "Simulator.h"
#include <list>
#include <memory>
#include <allegro5/allegro_primitives.h>
#include "Drawable.h"
#include "Updateable.h"
```

Classes

- class mySimulator

  contains lists of Drawable objects and Updateable objects and calls draw and update for their derived objects respectively

6.6.1 Detailed Description

Author

C. Barnson (`cbarnson@outlook.com`)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.7 src/Point.h File Reference

Describes a position on a 2D grid.

```
#include "Vector.h"
```

Classes

- struct Point

  represents a position on the display (grid)

6.7.1 Detailed Description

Describes a position on a 2D grid.

Author

C. Barnson (`cbarnson@outlook.com`)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.8 src/Simulator.cc File Reference

```
#include "Simulator.h"
#include <allegro5/allegro.h>
#include <allegro5/allegro_primitives.h>
#include <stdexcept>
```

6.8.1 Detailed Description

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.9 src/Simulator.h File Reference

```
#include <allegro5/allegro.h>
#include "Display.h"
```

Classes

- class Simulator
  Simulator object. Sets up Allegro library, and runs the main simulation loop.

6.9.1 Detailed Description

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.10 src/Triangle.h File Reference

Triangle represents a single triangle that moves from the top of the display to the bottom over time.

```
#include <allegro5/allegro_primitives.h>
#include <cstdlib>
#include <ctime>
#include "Drawable.h"
#include "Point.h"
#include "Updateable.h"
#include "Vector.h"
```

Classes

- class Triangle

    represents a triangle that falls from the top of the display to the bottom

6.10.1 Detailed Description

Triangle represents a single triangle that moves from the top of the display to the bottom over time.

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.11 src/Trunk.h File Reference

Trunk of the tree object.

```
#include <cmath>
#include <ctime>
#include <list>
#include <memory>
#include <vector>
#include <allegro5/allegro_primitives.h>
#include "Drawable.h"
#include "Line.h"
#include "Point.h"
#include "Updateable.h"
#include "Vector.h"
```

Classes

- class Trunk

  represents an elogating line from Point start to Point end in total_time

6.11.1 Detailed Description

Trunk of the tree object.

Author

C. Barnson (cbarnson@outlook.com)

Version

0.1

Date

2019-01-11

Copyright

Copyright (c) 2019

6.12 src/Updateable.h File Reference

Definition of the Updateable abstract class.

Classes

- class Updateable

  interface for updateable objects

6.12.1 Detailed Description

Definition of the Updateable abstract class.

Author

Cody Barnson

Bug no known bugs

6.13 src/Vector.h File Reference

Definition of the Vector class.

Classes

- struct Vector

  represents the change in position of x and y

6.13.1 Detailed Description

Definition of the Vector class.

Author

Cody Barnson

Bug No known bugs

Index