

Generated by Doxygen 1.9.5

1 General Information	1
1.0.0.1 C++ library for numerical calculations related to basic spacecraft motion issues, tak-	_
ing into account influence of gravity and vehicle propulsion.	1
1.1 Instalation	1
1.3 Units	2
1.4 json file	2
1.4.1 control	2
1.4.2 data	2
1.4.3 planets	3
1.4.4 ship	3
1.5 Example of Json file (Ship on the Earth orbit with a constant velocity):	3
2 Class Index	5
2.1 Class List	5
3 File Index	7
3.1 File List	7
4 Class Documentation	9
4.1 Control Class Reference	9
4.1.1 Detailed Description	9
4.1.2 Constructor & Destructor Documentation	10
4.1.2.1 Control()	10
4.1.3 Member Function Documentation	10
4.1.3.1 Check_input()	10
4.1.3.2 Print_Interval()	10
4.1.4 Member Data Documentation	10
4.1.4.1 engforce	
4.1.4.2 timeend	11
4.1.4.3 timestart	11
4.2 Planet Class Reference	11
4.2.1 Detailed Description	12
4.2.2 Constructor & Destructor Documentation	12
4.2.2.1 Planet()	12
4.2.3 Member Function Documentation	12
4.2.3.1 Move_Planet()	12
4.2.3.2 Print_info()	12
4.2.4 Member Data Documentation	12 13
4.2.4.1 ang_velocity	13
4.2.4.2 ISOrb	13
4.2.4.4 name	13
H.Z.H.H HQHIC	13

4.2.4.5 orb_data	 13
4.2.4.6 orb_pos	 13
4.2.4.7 orb_radius	 14
4.2.4.8 position	 14
4.2.4.9 radius	 14
4.2.4.10 start_ang	 14
4.3 Solver Class Reference	 14
4.3.1 Detailed Description	 16
4.3.2 Member Enumeration Documentation	 17
4.3.2.1 ode	 17
4.3.3 Member Function Documentation	 17
4.3.3.1 Adams_Bashford()	 17
4.3.3.2 Calculate_Grav()	 17
4.3.3.3 Calculate_Net()	 18
4.3.3.4 Check_Collision()	 18
4.3.3.5 dvdt()	 18
4.3.3.6 dxdt()	 19
4.3.3.7 Euler()	 19
4.3.3.8 Load_data()	 20
4.3.3.9 Load_file()	 20
4.3.3.10 Midpoint()	 21
4.3.3.11 Move_Orbit()	 21
4.3.3.12 Populate()	 21
4.3.3.13 Print_Pauses()	 21
4.3.3.14 Push_Back()	 22
4.3.3.15 Runge_Kutta()	 22
4.3.3.16 Save_data()	 22
4.3.3.17 Save_json()	 22
4.3.3.18 Save_Planets()	 23
4.3.3.19 Setup()	 23
4.3.3.20 Solve()	 23
4.3.3.21 UseEngine()	 23
4.3.3.22 Validate_Json()	 23
4.3.4 Member Data Documentation	 24
4.3.4.1 distance	 24
4.3.4.2 engine_data	 24
4.3.4.3 engine_used	 24
4.3.4.4 force_data	 24
4.3.4.5 fuel_data	 24
4.3.4.6 fuel_used	 25
4.3.4.7 G	 25
4.3.4.8 grav_forces	 25

4.3.4.9 kinetic_data	 . 25
4.3.4.10 mass_data	 . 25
4.3.4.11 method	 . 25
4.3.4.12 Planets	 . 26
4.3.4.13 position_data	 . 26
4.3.4.14 potential_data	 . 26
4.3.4.15 Ship	 . 26
4.3.4.16 step	 . 26
4.3.4.17 T	 . 26
4.3.4.18 time	 . 27
4.3.4.19 time_data	 . 27
4.3.4.20 TimeVect	 . 27
4.3.4.21 velocity_data	 . 27
4.4 Vector3D Class Reference	 . 27
4.4.1 Detailed Description	 . 29
4.4.2 Constructor & Destructor Documentation	 . 29
4.4.2.1 Vector3D() [1/2]	 . 29
4.4.2.2 Vector3D() [2/2]	 . 29
4.4.3 Member Function Documentation	 . 29
4.4.3.1 Add()	 . 29
4.4.3.2 Divide()	 . 30
4.4.3.3 Multiply()	 . 30
4.4.3.4 operator*()	 . 30
4.4.3.5 operator*=()	 . 31
4.4.3.6 operator+=()	 . 31
4.4.3.7 operator-=()	 . 32
4.4.3.8 operator/=()	 . 32
4.4.3.9 Subtract()	 . 32
4.4.3.10 VectorsEqual()	 . 33
4.4.3.11 Zero()	 . 33
4.4.4 Friends And Related Function Documentation	 . 33
4.4.4.1 operator*	 . 33
4.4.4.2 operator+	 . 34
4.4.4.3 operator	 . 34
4.4.4.4 operator/	 . 35
4.4.4.5 operator <<	 . 35
4.4.5 Member Data Documentation	 . 35
4.4.5.1 x	 . 35
4.4.5.2 y	 . 36
4.4.5.3 z	 . 36
4.5 Vehicle Class Reference	 . 36
4.5.1 Detailed Description	 . 37

4.5.2	Constructor & Destructor Documentation	37
	4.5.2.1 Vehicle() [1/2]	37
	4.5.2.2 Vehicle() [2/2]	38
4.5.3	Member Function Documentation	38
	4.5.3.1 Print_info()	38
	4.5.3.2 User_set()	38
4.5.4	Member Data Documentation	39
	4.5.4.1 CalculatedEnergy	39
	4.5.4.2 displacement	39
	4.5.4.3 engine	39
	4.5.4.4 force	39
	4.5.4.5 fuel	39
	4.5.4.6 fuel_usage	40
	4.5.4.7 KineticEnergy	40
	4.5.4.8 mass	40
	4.5.4.9 name	40
	4.5.4.10 position	40
	4.5.4.11 PotentialEnergy	40
	4.5.4.12 velocity	40
5 File Docume	ntation	41
5.1 Enginee	er_Thesis/Engineer_Thesis/Engineer_Thesis.cpp File Reference	41
5.1.1	Function Documentation	41
	5.1.1.1 main()	41
5.2 Enginee	er_Thesis/Engineer_Thesis/Instructions.md File Reference	41
5.3 Enginee	er_Thesis/Engineer_Thesis/Planet.h File Reference	41
5.4 Planet.h	1	42
5.5 Enginee	er_Thesis/Engineer_Thesis/Solver.cpp File Reference	42
5.6 Enginee	er_Thesis/Engineer_Thesis/Solver.h File Reference	42
5.7 Solver.h		43
5.8 Enginee	er_Thesis/Engineer_Thesis/Vector3D.cpp File Reference	44
5.8.1	Function Documentation	44
	5.8.1.1 operator*()	44
	5.8.1.2 operator+()	45
	5.8.1.3 operator-()	45
	5.8.1.4 operator/()	45
	5.8.1.5 operator<<()	46
5.9 Enginee	er_Thesis/Engineer_Thesis/Vector3D.h File Reference	46
5.10 Vector	3D.h	46
5.11 Engine	eer_Thesis/Engineer_Thesis/Vehicle.cpp File Reference	47
5.12 Engine	er_Thesis/Engineer_Thesis/Vehicle.h File Reference	47
5 13 Vehicle	e.h	47

Index 49

Chapter 1

General Information

1.0.0.1 C++ library for numerical calculations related to basic spacecraft motion issues, taking into account influence of gravity and vehicle propulsion.

1.1 Instalation

To use the library you need to:

- 1. Clone repository using: git clone https://github.com/Spectyte5/Engineer_Thesis
- 2. Make sure your compiler "sees" the libraries necessary(which are located in the external folder):
- json for modern C++: https://github.com/tristanpenman/valijson
- valijson: https://github.com/nlohmann/json

1.2 Operation

There are two options of using the library:

- 1. Provide a command line argument of type string, which is a path to your *.json* file for simulation. (examplary *.json* file can be found lower in this document) *example:*
 - .\Engineer_Thesis.cpp "./JSON_files/Sim1.json"
- 2. Run the code with your compiler of choice and then choose if you want to create a new simulation or load a *.json* from this directory:
 - <Library directory>/JSON_files/

2 General Information

1.3 Units

Units used are SI units:

- · mass in kilograms [kg]
- · position in meters [m]
- · velocity in meters per second [m/s]
- · force in Newtons [N]
- energy in Joule [J]
- · angle in radians [rad]
- angular velocity in radians per second [rad/s]

1.4 json file

.json file has one object with 4 seperate parts:

1.4.1 control

- **starttime** *array* of three double type elemetents, time when engine will be turned on for x, y, z axis, allowing user to set each one independently.
- endtime array of three double type elemetents, time when engine will be turned off for x, y, z axis, allowing user to set each one independently.
- force array of three double type elemetents, magnitude and the direction of engine thrust for x, y, z axis, allowing user to set each one independently.

1.4.2 data

- ode integer type number meaning which ODE solving method should be used: 0. Adams-Bashford
- 1. Euler
- 2. Midpoint
- 3. Runge-Kutta IV
- **step** double type *number* equal to timesteps used for simulation.
- time double type *number*, the final time of simulation.

1.4.3 planets

- · name string type, name of the planet
- mass double type number, mass of the planet
- radius double type number, radius of the planet
- orbit boolean type, is orbitting or strationary [true/false] For orbitting Planets (orbit = true):
- **start_angle** double type *number*, phase used for orbital motion (start angle)
- orbit_radius double type number, radius of the orbit
- ang_velocity double type number, constant angular velocity of the planet
- **orbit_pos** *array* of three double type elemetents, position of the center of the orbit (x,y,z) **NOTE**: only x,z are taken into account y value is always 0 For stationary Planets (orbit = false):
- position array of three double type elemetents, magnitude and the direction of position vector (x,y,z)

1.4.4 ship

- fuel double type number, fuel mass of the ship
- fuel usage double type number, constant ammount of fuel used when engines are turned on
- mass double type number, total mass of the ship
- · name string type, name of the ship
- position array of three double type elemetents, magnitude and the direction of position vector (x,y,z)
- velocity array of three double type elemetents, magnitude and the direction of velocity vector (x,y,z)

1.5 Example of Json file (Ship on the Earth orbit with a constant velocity):

4 General Information

Chapter 2

Class Index

2.1 Class List

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

Control		
	Class for engine intervals used in simulation	9
Planet		
	Class for different Planets	11
Solver		
	Main class, used for solving	14
Vector3D		
	Class for Three-Dimensional Vectors	27
Vehicle		
	Class for different spaceship objects	36

6 Class Index

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

ingineer_Thesis/Engineer_Thesis/Engineer_Thesis.cpp	. 41
ingineer_Thesis/Engineer_Thesis/Planet.h	. 41
ingineer_Thesis/Engineer_Thesis/Solver.cpp	. 42
ingineer_Thesis/Engineer_Thesis/Solver.h	. 42
ingineer_Thesis/Engineer_Thesis/Vector3D.cpp	. 44
ingineer_Thesis/Engineer_Thesis/Vector3D.h	. 46
ingineer_Thesis/Engineer_Thesis/Vehicle.cpp	. 47
ingineer_Thesis/Engineer_Thesis/Vehicle.h	. 47

8 File Index

Chapter 4

Class Documentation

4.1 Control Class Reference

Class for engine intervals used in simulation.

```
#include <Engineer_Thesis/Engineer_Thesis/Solver.h>
```

Public Member Functions

- Control ()
- void Print_Interval ()
- bool Check_input (Solver &method)

Function for checking if intervals given by user are possible to be implemented.

Public Attributes

- Vector3D timestart = { 0,0,0 }
- Vector3D timeend = { 0,0,0 }
- Vector3D engforce = { 0,0,0 }

4.1.1 Detailed Description

Class for engine intervals used in simulation.

This class is used to create engine intervals, print and check input given by user for them.

Parameters

timestart	is Vector3D object at what time the engine will be turned on
timeend	is Vector3D object at what time the engine will be turned off
engforce	is Vector3D object the direction and magnitude of thrust force vector

See also

Vector3D for information about three-dimenstional vectors class

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Control()

```
Control::Control ( ) [inline]
```

4.1.3 Member Function Documentation

4.1.3.1 Check_input()

Function for checking if intervals given by user are possible to be implemented.

Iterates through all intervals given by the user, checks if the engine start, end and thrust values are correct and don't intersect each other.

Returns

true if all conditions are satisfied

4.1.3.2 Print_Interval()

```
void Control::Print_Interval ( ) [inline]
```

4.1.4 Member Data Documentation

4.1.4.1 engforce

```
Vector3D Control::engforce = { 0,0,0 }
```

4.2 Planet Class Reference 11

4.1.4.2 timeend

```
Vector3D Control::timeend = { 0,0,0 }
```

4.1.4.3 timestart

```
Vector3D Control::timestart = { 0,0,0 }
```

4.2 Planet Class Reference

Class for different Planets.

```
#include <Engineer_Thesis/Engineer_Thesis/Planet.h>
```

Public Member Functions

• Planet ()

default constructor

void Print_info ()

Function printing information about planet.

• void Move_Planet (double time)

Move Planet around orbit.

Public Attributes

```
    std::vector< Vector3D > orb_data
```

vector used for storing position data of orbitting planets

• double mass = 0

mass of the planet [kg]

• double radius = 0

radius of the planet [m]

• double orb_radius = 0

radius of the orbit [m]

• double ang_velocity = 0

angular velocity [rad/s]

• double start_ang = 0

phase [rad]

• Vector3D position = { 0,0,0 }

position of the ship

• Vector3D orb_pos = { 0,0,0 }

position of the orbit in space

• std::string name = ""

name of the planet

• bool isOrb = false

Variable connected to the orbitting of planet if true it means that the Planet orbits around a given point.

4.2.1 Detailed Description

Class for different Planets.

This class handles creating, moving and printing information about a planet

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Planet()

```
Planet::Planet ( ) [inline]
```

default constructor

4.2.3 Member Function Documentation

4.2.3.1 Move_Planet()

Move Planet around orbit.

Function moving planet around orbit with given angular velocity starting from given angle

Parameters

```
time is current time of simulation
```

4.2.3.2 Print_info()

```
void Planet::Print_info ( ) [inline]
```

Function printing information about planet.

4.2.4 Member Data Documentation

4.2 Planet Class Reference

4.2.4.1 ang_velocity

```
double Planet::ang_velocity = 0
angular velocity [rad/s]
```

4.2.4.2 isOrb

```
bool Planet::isOrb = false
```

Variable connected to the orbitting of planet if true it means that the Planet orbits around a given point.

4.2.4.3 mass

```
double Planet::mass = 0
mass of the planet [kg]
```

4.2.4.4 name

```
std::string Planet::name = ""
name of the planet
```

4.2.4.5 orb_data

```
std::vector<Vector3D> Planet::orb_data
```

vector used for storing position data of orbitting planets

4.2.4.6 orb_pos

position of the orbit in space

```
Vector3D Planet::orb_pos = { 0,0,0 }
```

Note

orbit is actually only x,z so the y component will always be 0

4.2.4.7 orb_radius

```
double Planet::orb_radius = 0
radius of the orbit[m]
```

4.2.4.8 position

```
Vector3D Planet::position = { 0,0,0 }
position of the ship
```

4.2.4.9 radius

```
double Planet::radius = 0
radius of the planet [m]
```

4.2.4.10 start_ang

```
double Planet::start_ang = 0
phase [rad]
```

4.3 Solver Class Reference

Main class, used for solving.

```
#include <Engineer_Thesis/Engineer_Thesis/Solver.h>
```

Public Types

• enum ode { adams , euler , midpoint , runge } Enum of different solving ODEs methods.

Public Member Functions

• double dvdt (double t, double v, double f, double m)

Derivative of Velocity.

• double dxdt (double t, double v, double x)

Derivative of Position.

• void Populate ()

Define planets in simulation.

• void Setup ()

Create all simulation elements.

• bool Validate_Json (std::string &filename)

Json Validation function.

• void Save_json ()

Save simulation as a Json file.

• std::ifstream Load_file (std::string sys_path, std::string filepath, std::string extenstion)

Function for Loading file from a directory.

void Load_data (std::string &filename)

Function Setting parameters from the file.

· bool Check Collision (Planet &Planet)

Collistion checking function.

• bool UseEngine ()

Applying the thrust force from the engines.

void Calculate_Grav ()

Function for calculating gravity.

void Calculate_Net ()

Function for Calculating Net force.

void Euler (double &velocity, double &position, double force, double mass)

Euler method.

• void Runge_Kutta (double &velocity, double &position, double force, double mass)

Runge-Kutta IV method.

• void Midpoint (double &velocity, double &position, double force, double mass)

Midpoint method.

• void Adams_Bashford (double &velocity, double &position, double force, double mass)

Adams-Bashforth's method.

· void Solve ()

Main solving function.

• void Push Back ()

Put all parameters in vectors.

void Move_Orbit ()

Method for changing position of orbitting planets.

• void Save_Planets ()

Function for saving planets.

void Save_data ()

Save simulation data.

• void Print_Pauses ()

Pauses between simulation elements printing.

Public Attributes

• const double G = 6.67259e-11

Gravitational constant.

• std::vector< double > time_data

vectors for storing current time value

• std::vector< double > mass_data

vectors for storing current mass

std::vector< double > fuel data

vectors for storing current fuel value

• std::vector< double > kinetic_data

vectors for storing current kinetic energy value

std::vector< double > potential_data

vectors for storing current potential energy value

std::vector< Vector3D > position_data

vector used for storing current position data

std::vector< Vector3D > velocity_data

vector used for storing current velocity data

std::vector< Vector3D > engine data

vector used for storing current engine data

std::vector< Vector3D > force_data

vector used for storing current force data

std::vector< Planet > Planets

vector storing planets in the simulation

std::vector < Control > TimeVect

vector storing force intervals of type Control

Vehicle Ship = Vehicle("", 0, 0, 0, 0, 0, 0, 0, 0, 0)

Ship member used in simulation.

Vector3D grav_forces

Gravitational force and distance from Planet at the time T.

- · Vector3D distance
- bool engine_used = false

Boolean used for removing fuel used.

• int method =0

Method stored as an int used for enum.

• double T = 0

time of simulation

• double step = 0

time step between increments

• double time = 0

current simulation time

• double fuel_used = 0

ammount of fuel_used at the iteration

4.3.1 Detailed Description

Main class, used for solving.

Class taking care of loading, validating, saving files and calculating results using different solvers

4.3.2 Member Enumeration Documentation

4.3.2.1 ode

enum Solver::ode

Enum of different solving ODEs methods.

Enumerator

adams	
euler	
midpoint	
runge	

4.3.3 Member Function Documentation

4.3.3.1 Adams_Bashford()

Adams-Bashforth's method.

Function solving and ODE using the Adams-Bashforth's predictor and corrector method for calculating and setting parameters of the ship

Parameters

velocity	is velocity at current time
position	is position at current time
force	is force acting on the spaceship
mass	is mass of the spaceship

4.3.3.2 Calculate_Grav()

```
void Solver::Calculate_Grav ( )
```

Function for calculating gravity.

Iterate through planets and calculate Gravitation forces acting on the ship and it's potential energy

4.3.3.3 Calculate_Net()

```
void Solver::Calculate_Net ( )
```

Function for Calculating Net force.

Sets the value of net force taking in to account gravitational and thrust forces

See also

Calculate_Grav() and UseEngine() for more information about forces calculation

4.3.3.4 Check_Collision()

Collistion checking function.

Function calculating distance between the Ship and Planet.

Parameters

```
Planet is a Planet Class object which we are checking ships collistion with
```

Returns

true if we have a collistion and false if there not

Attention

If Ship is exactly on the Planet's surface it does not count as a collistion

4.3.3.5 dvdt()

Derivative of Velocity.

Function returning the derivative of velocity.

Parameters

t	is current time of simulation
V	is velocity at time t
f	is force at time t
m	is mass of the object

Returns

double discreibing velocity change in the last interval

4.3.3.6 dxdt()

Derivative of Position.

Function returning the derivative of position.

Parameters

t	is current time of simulation
V	is velocity at time t
Х	is position at time t

Returns

double discreibing position change in the last interval

4.3.3.7 Euler()

Euler method.

Function solving and ODE using the Euler's method and setting parameters of the ship

Parameters

velocity	is velocity at current time
position	is position at current time
force	is force acting on the spaceship
mass	is mass of the spaceship

4.3.3.8 Load_data()

Function Setting parameters from the file.

Loaded file is used to set paramaters

See also

Load_file() for information about loading file

4.3.3.9 Load_file()

Function for Loading file from a directory.

Display files in directory and open file with a given filepath

Parameters

sys_path	is directory in which we are looking for files
filepath	is path to the file
extension	is extension of the file ex. "txt"

Returns

loaded file as ifstream

4.3.3.10 Midpoint()

Midpoint method.

Function solving and ODE using the modified Euler's method (Midpoint method) and setting parameters of the ship

Parameters

velocity	is velocity at current time
position	is position at current time
force	is force acting on the spaceship
mass	is mass of the spaceship

4.3.3.11 Move_Orbit()

```
void Solver::Move_Orbit ( )
```

Method for changing position of orbitting planets.

Checks if planets is orbitting around a point and if yes changes its position and saves it to vector

4.3.3.12 Populate()

```
void Solver::Populate ( )
```

Define planets in simulation.

Gets ammount of planets in simulation, sets parameters for planet and puts it in the planets vector

4.3.3.13 Print_Pauses()

```
void Solver::Print_Pauses ( ) [inline]
```

Pauses between simulation elements printing.

Function printing '=' signs to allow better seperation between simulation elements and improve comfort of reading the text displayed.

4.3.3.14 Push_Back()

```
void Solver::Push_Back ( )
```

Put all parameters in vectors.

Save all neccessary values at time t into corresponding vectors

4.3.3.15 Runge_Kutta()

Runge-Kutta IV method.

Function solving and ODE using the Runge-Kutta IV-order method and setting parameters of the ship

Parameters

velocity	is velocity at current time
position	is position at current time
force	is force acting on the spaceship
mass	is mass of the spaceship

4.3.3.16 Save_data()

```
void Solver::Save_data ( )
```

Save simulation data.

Saves all parameters and calls the function for saving planets' data.

See also

Save_Planets() for more information about saving planets

4.3.3.17 Save_json()

```
void Solver::Save_json ( )
```

Save simulation as a Json file.

Function used in create a simulation mode to save all parameters of the ship and planets in a json file which then can be reloaded in load mode.

4.3.3.18 Save_Planets()

```
void Solver::Save_Planets ( )
```

Function for saving planets.

Save all planets' paramaters to a seprate file

4.3.3.19 Setup()

```
void Solver::Setup ( )
```

Create all simulation elements.

Setup Particle and planets in simulation, fill all engine intervals

4.3.3.20 Solve()

```
void Solver::Solve ( )
```

Main solving function.

This function loops through time interval calling all functions used for calculation and prints result on screen.

See also

Adams_Bashford(), Midpoint(), Euler(), Runge_Kutta() for more information about solving ODE's

4.3.3.21 UseEngine()

```
bool Solver::UseEngine ( )
```

Applying the thrust force from the engines.

Checks if we are in any of intervals defined by user and if yes and fuel is available it applies engine force

Returns

true if engine was used and no if not

4.3.3.22 Validate_Json()

Json Validation function.

Check if vector file validates against the schema

Parameters

filename is a filepath for the json file that will be validated	filename
---	----------

4.3.4 Member Data Documentation

4.3.4.1 distance

Vector3D Solver::distance

4.3.4.2 engine_data

std::vector<Vector3D> Solver::engine_data

vector used for storing current engine data

4.3.4.3 engine_used

bool Solver::engine_used = false

Boolean used for removing fuel used.

4.3.4.4 force_data

std::vector<Vector3D> Solver::force_data

vector used for storing current force data

4.3.4.5 fuel_data

std::vector<double> Solver::fuel_data

vectors for storing current fuel value

4.3.4.6 fuel_used

```
double Solver::fuel_used = 0
```

ammount of fuel_used at the iteration

4.3.4.7 G

```
const double Solver::G = 6.67259e-11
```

Gravitational constant.

4.3.4.8 grav_forces

```
Vector3D Solver::grav_forces
```

Gravitational force and distance from Planet at the time T.

4.3.4.9 kinetic_data

```
std::vector<double> Solver::kinetic_data
```

vectors for storing current kinetic energy value

4.3.4.10 mass data

```
std::vector<double> Solver::mass_data
```

vectors for storing current mass

4.3.4.11 method

```
int Solver::method =0
```

Method stored as an int used for enum.

4.3.4.12 Planets

```
std::vector<Planet> Solver::Planets
```

vector storing planets in the simulation

4.3.4.13 position_data

```
std::vector<Vector3D> Solver::position_data
```

vector used for storing current position data

4.3.4.14 potential_data

```
std::vector<double> Solver::potential_data
```

vectors for storing current potential energy value

4.3.4.15 Ship

```
Vehicle Solver::Ship = Vehicle("", 0, 0, 0, 0, 0, 0, 0, 0)
```

Ship member used in simulation.

4.3.4.16 step

```
double Solver::step = 0
```

time step between increments

4.3.4.17 T

```
double Solver::T = 0
```

time of simulation

4.3.4.18 time

double Solver::time = 0

current simulation time

4.3.4.19 time_data

std::vector<double> Solver::time_data

vectors for storing current time value

4.3.4.20 TimeVect

std::vector<Control> Solver::TimeVect

vector storing force intervals of type Control

4.3.4.21 velocity_data

std::vector<Vector3D> Solver::velocity_data

vector used for storing current velocity data

4.4 Vector3D Class Reference

Class for Three-Dimensional Vectors.

#include <Engineer_Thesis/Engineer_Thesis/Vector3D.h>

Public Member Functions

· Vector3D ()

default constructor

Vector3D (double x, double y, double z)

constructor with x, y and z values

Vector3D & Add (const Vector3D &vect)

Add two vectors.

Vector3D & Subtract (const Vector3D &vect)

Substact two vectors.

Vector3D & Multiply (const Vector3D &vect)

Multiply two vectors.

Vector3D & Divide (const Vector3D &vect)

Divide two vectors.

Vector3D & operator+= (const Vector3D &vect)

Add two vectors with += operator.

Vector3D & operator-= (const Vector3D &vect)

Substract two vectors with -= operator.

Vector3D & operator*= (const Vector3D &vect)

Multiply two vectors with *= operator.

Vector3D & operator/= (const Vector3D &vect)

Divide two vectors with /= operator.

• Vector3D & operator* (const double &i)

Multiply vector by scale.

Vector3D & Zero ()

Sets values of the x,y,z to 0.

bool VectorsEqual (const Vector3D &vect)

Function checking if two vectors have the same x,y,z components.

Public Attributes

double x

x component of the vector

· double y

y component of the vector

• double z

z component of the vector

Friends

Vector3D operator+ (const Vector3D &v1, const Vector3D &v2)

Add two vectors with + operator.

Vector3D operator- (const Vector3D &v1, const Vector3D &v2)

Substract two vectors with - operator.

Vector3D operator* (const Vector3D &v1, const Vector3D &v2)

Multiply two vectors with * operator.

• Vector3D operator/ (const Vector3D &v1, const Vector3D &v2)

Divide two vectors with / operator.

std::ostream & operator<< (std::ostream &output, const Vector3D &vect)

overload of << opearator for printing vectors

4.4.1 Detailed Description

Class for Three-Dimensional Vectors.

This class is used for operations and storing parameters of the three-dimensional vectors

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Vector3D() [1/2]

```
Vector3D::Vector3D ( )
```

default constructor

4.4.2.2 Vector3D() [2/2]

```
Vector3D::Vector3D ( \label{eq:double x, double y, double z } \mbox{double } z \mbox{ )}
```

constructor with x, y and z values

4.4.3 Member Function Documentation

4.4.3.1 Add()

Add two vectors.

```
\textbf{4.4.3.1.1} \quad \textbf{Example} \quad \texttt{v1.Add(v2)} \ \textit{// which equals to v1 + v2}
```

Parameters

vect is vector being added to the vector calling this method

Returns

Vector3D that is a vector on which method was called with vect value added to it

4.4.3.2 Divide()

Divide two vectors.

```
\textbf{4.4.3.2.1} \quad \textbf{Example} \quad \texttt{v1.Divide(v2)} \ \textit{// which equals to v1 / v2}
```

Parameters

Vect is vector writer the vector calling this method is divided by		vect	is vector which the vector calling this method is divided by
--	--	------	--

Returns

Vector3D that is a vector on which method was called divided by vect value

4.4.3.3 Multiply()

Multiply two vectors.

```
4.4.3.3.1 Example v1.Multiply(v2) // which equals to v1 \star v2
```

Parameters

vect is vector which the vector calling this method is multiplyied by

Returns

Vector3D that is a vector on which method was called multiplyied by vect value

4.4.3.4 operator*()

```
Vector3D & Vector3D::operator* ( const double & i )
```

Multiply vector by scale.

Parameters

i is integer value by which we want to multiply our vector

4.4.3.5 operator*=()

Multiply two vectors with *= operator.

4.4.3.5.1 Example $v1 \neq v2$

Parameters

v1	is vector multiplyied
v2	is vector we are multiplying by

Returns

v1 multiplied by v2 value

4.4.3.6 operator+=()

Add two vectors with += operator.

4.4.3.6.1 Example v1 += v2

Parameters

v1	is vector which we are adding into
v2	is vector being added

Returns

v1 increased by v2 value

4.4.3.7 operator-=()

Substract two vectors with -= operator.

4.4.3.7.1 Example v1 -= v2

Parameters

v1	is vector which we are substracting from
v2	is vector being substracted

Returns

v1 decreased by v2 value

4.4.3.8 operator/=()

Divide two vectors with /= operator.

4.4.3.8.1 Example v1 /= v2

Parameters

v1	is vector divided
v2	is vector we are dividing by

Returns

v1 divided by v2 value

4.4.3.9 Subtract()

Substact two vectors.

$\textbf{4.4.3.9.1} \quad \textbf{Example} \quad \text{v1.Substract(v2)} \ \ // \ \ \text{which equals to v1 - v2}$

Parameters

vect	is vector being substracted from the vector calling this method

Returns

Vector3D that is a vector on which method was called with vect value substracted from it

4.4.3.10 VectorsEqual()

Function checking if two vectors have the same x,y,z components.

Parameters

vect is vector compared to the vector calling this method

Returns

true if two vectors are the same, false if not

4.4.3.11 Zero()

```
Vector3D & Vector3D::Zero ( )
```

Sets values of the x,y,z to 0.

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator*

Multiply two vectors with * operator.

4.4.4.1.1 Example result = v1 * v2

Parameters

v1	is first vector being multiplyied	
v2 is second vector we are multiplying		

Returns

```
vector equal to v1 * v2
```

4.4.4.2 operator+

Add two vectors with + operator.

4.4.4.2.1 Example result = v1 + v2

Parameters

v1	is first vector being addded
v2	is second vector being added

Returns

```
vector equal to v1 + v2
```

4.4.4.3 operator-

Substract two vectors with - operator.

4.4.4.3.1 Example result = v1 - v2

Parameters

v1 is first vector being substracted	
v2	is second vector being substracted

Returns

vector equal to v1 - v2

4.4.4.4 operator/

Divide two vectors with / operator.

4.4.4.1 Example result = v1 / v2

Parameters

v1	is first vector being devided
v2	is second vector we are deviding by

Returns

vector equal to v1 / v2

4.4.4.5 operator <<

overload of << opearator for printing vectors

Parameters

output	is ofstream where we will print data
vect	is a vector being printed

4.4.5 Member Data Documentation

4.4.5.1 x

double Vector3D::x

x component of the vector

4.4.5.2 y

double Vector3D::y

y component of the vector

4.4.5.3 z

double Vector3D::z

z component of the vector

4.5 Vehicle Class Reference

Class for different spaceship objects.

#include <Engineer_Thesis/Engineer_Thesis/Vehicle.h>

Public Member Functions

• Vehicle ()

Default Constructor.

• Vehicle (std::string n, double rx, double ry, double rz, double vx, double vy, double vz, double m, double fuel, double fuel_usage)

Constructor assiging given paramaters.

void Print_info ()

Print information about the Ship.

• void User_set ()

User set ships parameters.

Public Attributes

```
• std::string name
```

name of the ship

· Vector3D position

Vector3D position on x,y,z axis [m].

Vector3D velocity

Vector3D velocity on x,y,z axis [m/s].

• Vector3D engine = { 0,0,0 }

engine is a Vector3D thrust force on x,y,z axis [N]

Vector3D force = { 0,0,0 }

Vector3D net force acting on spaceship on x,y,z axis [N].

• Vector3D displacement = { 0,0,0 }

displacement of the ship from initial position[m]

• double mass = 0

total mass of the ship with fuel [kg]

• double fuel = 0

mass of fuel carried by the ship[kg]

• double fuel_usage = 0

ammount of fuel used by engines [kg/s]

• double PotentialEnergy = 0

total potential energy from all planets acting on the spaceship [J]

• double KineticEnergy = 0

energy from velocity whith which spaceship is moving [J]

• bool CalculatedEnergy = 0

true or false depending on whether the planets where already initialized

4.5.1 Detailed Description

Class for different spaceship objects.

Ship is a body having no size, no rotation (Point-mass)

Note

In few places the mass actually is a mass without fuel, inputed by user that then has the fuel mass added to it.

See also

Vector3D for more information about three-dimenstional vector objects

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Vehicle() [1/2]

Vehicle::Vehicle () [inline]

Default Constructor.

4.5.2.2 Vehicle() [2/2]

```
Vehicle::Vehicle (
std::string n,
double rx,
double ry,
double rz,
double vx,
double vy,
double vz,
double m,
double fuel,
double fuel_usage)
```

Constructor assiging given paramaters.

Parameters

n	is name of the ship
rx	is position on x axis [m]
ry	is position on y axis [m]
rz	is position on z axis [m]
VX	is velocity on x axis [m/s]
vy	is velocity on y axis [m/s]
VZ	is velocity on z axis [m/s]
m	is mass of the ship [kg]
fuel	is mass of fuel carried by the ship [kg]
fuel_usage	is ammount of fuel used in [kg/s]

4.5.3 Member Function Documentation

4.5.3.1 Print_info()

```
void Vehicle::Print_info ( )
```

Print information about the Ship.

Function for printing each paramter of the ship on screen

4.5.3.2 User_set()

```
void Vehicle::User_set ( )
```

User set ships parameters.

Function allowing user to set values of the Ship, used in create a simulation mode.

4.5.4 Member Data Documentation

4.5.4.1 CalculatedEnergy

```
bool Vehicle::CalculatedEnergy = 0
```

true or false depending on whether the planets where already initialized

See also

Planet more info about planets

4.5.4.2 displacement

```
Vector3D Vehicle::displacement = { 0,0,0 }
```

displacement of the ship from initial position[m]

4.5.4.3 engine

```
Vector3D Vehicle::engine = { 0,0,0 }
```

engine is a Vector3D thrust force on x,y,z axis [N]

4.5.4.4 force

```
Vector3D Vehicle::force = { 0,0,0 }
```

Vector3D net force acting on spaceship on x,y,z axis [N].

4.5.4.5 fuel

```
double Vehicle::fuel = 0
```

mass of fuel carried by the ship[kg]

4.5.4.6 fuel_usage

```
double Vehicle::fuel_usage = 0
ammount of fuel used by engines [kg/s]
```

4.5.4.7 KineticEnergy

```
double Vehicle::KineticEnergy = 0
```

energy from velocity whith which spaceship is moving [J]

4.5.4.8 mass

```
double Vehicle::mass = 0
```

total mass of the ship with fuel [kg]

4.5.4.9 name

```
std::string Vehicle::name
```

name of the ship

4.5.4.10 position

```
Vector3D Vehicle::position
```

Vector3D position on x,y,z axis [m].

4.5.4.11 PotentialEnergy

```
double Vehicle::PotentialEnergy = 0
```

total potential energy from all planets acting on the spaceship [J]

4.5.4.12 velocity

Vector3D Vehicle::velocity

Vector3D velocity on x,y,z axis [m/s].

Chapter 5

File Documentation

5.1 Engineer_Thesis/Engineer_Thesis/Engineer_Thesis.cpp File Reference

```
#include <iostream>
#include "Solver.h"
```

Functions

```
• int main (int argc, char *argv[])
```

5.1.1 Function Documentation

5.1.1.1 main()

```
int main (
          int argc,
          char * argv[] )
```

- 5.2 Engineer_Thesis/Engineer_Thesis/Instructions.md File Reference
- 5.3 Engineer_Thesis/Engineer_Thesis/Planet.h File Reference

```
#include "Vector3D.h"
```

42 File Documentation

Classes

class Planet

Class for different Planets.

5.4 Planet.h

Go to the documentation of this file.

```
1 #pragma once
2 #include "Vector3D.h"
7 class Planet {
9 public:
        std::vector <Vector3D> orb_data;
12
14
        double mass = 0;
    double radius = 0;
double orb_radius = 0;
double ang_velocity = 0;
16
20
      double start_ang = 0;
23
   Vector3D position = { 0,0,0 };
Vector3D orb_pos = { 0,0,0 };
std::string name = "";
bool isOrb = false;
2.5
28
30
33
35
      Planet() {}
36
       void Print_info() {
38
39
       40
41
42
             if (isOrb) {
       std::cout « "\nOrbit Radius: " « orb_radius « " m" « "\nOrbit Velocity: " « ang_velocity « " rad/s" « "\nOrbit Center: " « orb_pos « " m";
43
44
45
            std::cout « "\n";
47
        void Move Planet (double time) {
52
53
            position.x = orb_pos.x + orb_radius * cos(start_ang + ang_velocity * time);
position.z = orb_pos.z + orb_radius * sin(start_ang + ang_velocity * time);
56
             orb_data.push_back(position);
57
58 };
```

5.5 Engineer_Thesis/Engineer_Thesis/Solver.cpp File Reference

```
#include "Solver.h"
#include <filesystem>
#include <json.hpp>
#include <valijson_nlohmann_bundled.hpp>
```

5.6 Engineer_Thesis/Engineer_Thesis/Solver.h File Reference

```
#include <fstream>
#include <iomanip>
#include "Vehicle.h"
#include "Planet.h"
```

5.7 Solver.h 43

Classes

class Solver

Main class, used for solving.

class Control

Class for engine intervals used in simulation.

5.7 Solver.h

Go to the documentation of this file.

```
#pragma once
2 #include <fstream>
3 #include <iomanip>
4 #include "Vehicle.h"
5 #include "Planet.h"
8 //forward declare class
9 class Control:
10
14 class Solver {
15
16 public:
18
       const double G = 6.67259e-11;
       std::vector <double> time_data;
20
       std::vector <double> mass_data;
22
       std::vector <double> fuel_data;
24
       std::vector <double> kinetic_data;
28
       std::vector <double> potential_data;
29
       std::vector <Vector3D> position_data;
std::vector <Vector3D>velocity_data;
31
33
35
       std::vector <Vector3D> engine_data;
       std::vector <Vector3D> force_data;
37
39
       std::vector <Planet> Planets;
       std::vector <Control> TimeVect;
Vehicle Ship = Vehicle("", 0, 0, 0, 0, 0, 0, 0, 0, 0);
41
43
       Vector3D grav_forces, distance;
45
       bool engine_used = false;
47
49
       enum ode { adams, euler, midpoint, runge};
51
       int method=0;
52
       double T = 0;
54
       double step = 0;
double time = 0;
56
58
       double fuel_used = 0;
60
       double dvdt(double t, double v, double f, double m) { return f / m; } double dxdt(double t, double v, double x) { return v; }
70
78
82
       void Populate();
       void Setup();
86
       bool Validate_Json(std::string& filename);
       void Save_json();
95
103
        std::ifstream Load_file(std::string sys_path, std::string filepath, std::string extenstion);
108
        void Load_data(std::string& filename);
115
        bool Check_Collision(Planet& Planet);
120
        bool UseEngine();
124
        void Calculate_Grav();
129
        void Calculate_Net();
137
         void Euler(double& velocity, double& position, double force, double mass);
145
        void Runge_Kutta(double& velocity, double& position, double force, double mass);
153
        void Midpoint(double& velocity, double& position, double force, double mass);
        void Adams_Bashford(double& velocity, double& position, double force, double mass);
161
        void Solve();
166
170
        void Push_Back();
174
         void Move_Orbit();
178
        void Save_Planets();
183
        void Save_data();
184
        void Print_Pauses() {
188
             std::cout « std::setfill('=') « std::setw(120) « "\n";
189
190
191 };
192
200 class Control{
201
202 public:
         Vector3D timestart= { 0,0,0 }, timeend = { 0,0,0 }, engforce = { 0,0,0 };
```

44 File Documentation

```
204
205
        Control() {};
206
207
        void Print_Interval() {
            208
209
210
211
216
        bool Check_input(Solver& method) {
217
                //initial check:
            if (method.TimeVect.empty()) {
218
219
                //timestart
                if (timestart.x < 0 || timestart.y < 0 || timestart.z < 0) return false;</pre>
220
221
222
                 if (timeend.x > method.T || timeend.y > method.T || timeend.z > method.T) return false;
223
                if (timeend.x < timestart.x || timeend.y < timestart.y || timeend.z < timestart.z) return</pre>
      false;
224
225
            else {
226
                if (timestart.x < 0 || timestart.y < 0 || timestart.z < 0) return false;</pre>
                if (timeend.x > method.T || timeend.y > method.T || timeend.z > method.T) return false;
228
                if (timeend.x < timestart.x || timeend.y < timestart.y || timeend.z < timestart.z) return</pre>
      false;
                //check if interval does not intersect previous interval
if (timestart.x < method.TimeVect.back().timeend.x || timestart.y <</pre>
229
230
      method.TimeVect.back().timeend.y || timestart.z < method.TimeVect.back().timeend.z) return false;</pre>
231
232
            return true;
233
234 1:
```

5.8 Engineer_Thesis/Engineer_Thesis/Vector3D.cpp File Reference

```
#include "Vector3D.h"
```

Functions

- Vector3D operator+ (const Vector3D &v1, const Vector3D &v2)
- Vector3D operator- (const Vector3D &v1, const Vector3D &v2)
- Vector3D operator* (const Vector3D &v1, const Vector3D &v2)
- Vector3D operator/ (const Vector3D &v1, const Vector3D &v2)
- std::ostream & operator<< (std::ostream &output, const Vector3D &vect)

5.8.1 Function Documentation

5.8.1.1 operator*()

```
Vector3D operator* (  {\rm const~Vector3D~\&~v1,}   {\rm const~Vector3D~\&~v2~)}
```

5.8.1.1.1 Example result = v1 * v2

Parameters

v1	is first vector being multiplyied
v2	is second vector we are multiplying by

Returns

```
vector equal to v1 * v2
```

5.8.1.2 operator+()

```
Vector3D operator+ (  {\rm const~Vector3D~\&~v1,}   {\rm const~Vector3D~\&~v2~)}
```

5.8.1.2.1 Example result = v1 + v2

Parameters

v1	is first vector being addded
v2	is second vector being added

Returns

vector equal to v1 + v2

5.8.1.3 operator-()

```
Vector3D operator- (  {\rm const~Vector3D~\&~v1,}   {\rm const~Vector3D~\&~v2~)}
```

5.8.1.3.1 Example result = v1 - v2

Parameters

v1	is first vector being substracted
v2	is second vector being substracted

Returns

vector equal to v1 - v2

5.8.1.4 operator/()

```
Vector3D operator/ (  {\rm const~Vector3D~\&~v1,}    {\rm const~Vector3D~\&~v2~)}
```

$\textbf{5.8.1.4.1} \quad \textbf{Example} \quad \texttt{result} \; = \; \texttt{v1} \; / \; \; \texttt{v2}$

46 File Documentation

Parameters

v1	is first vector being devided
v2	is second vector we are deviding by

Returns

vector equal to v1 / v2

5.8.1.5 operator << ()

Parameters

output	is ofstream where we will print data
vect	is a vector being printed

5.9 Engineer_Thesis/Engineer_Thesis/Vector3D.h File Reference

#include <iostream>

Classes

• class Vector3D

Class for Three-Dimensional Vectors.

5.10 Vector3D.h

Go to the documentation of this file.

```
1 #pragma once
2 #include <iostream>
7 class Vector3D {
9 public:
       double x;
11
       double y;
double z;
13
15
16
       Vector3D();
      Vector3D(double x, double y, double z);
20
21
30
       Vector3D& Add(const Vector3D& vect);
39
       Vector3D& Subtract(const Vector3D& vect);
       Vector3D& Multiply(const Vector3D& vect);
```

```
Vector3D& Divide(const Vector3D& vect);
         friend Vector3D operator+ (const Vector3D& v1, const Vector3D& v2);
friend Vector3D operator- (const Vector3D& v1, const Vector3D& v2);
friend Vector3D operator* (const Vector3D& v1, const Vector3D& v2);
friend Vector3D operator* (const Vector3D& v1, const Vector3D& v2);
87
97
107
           Vector3D& operator+=(const Vector3D& vect);
           Vector3D& operator-=(const Vector3D& vect);
117
           Vector3D& operator*=(const Vector3D& vect);
137
           Vector3D& operator/=(const Vector3D& vect);
138
142
           Vector3D& operator*(const double& i);
143
           Vector3D& Zero();
145
146
151
           friend std::ostream& operator « (std::ostream& output, const Vector3D& vect);
152
157
           bool VectorsEqual (const Vector3D& vect);
158 };
```

5.11 Engineer_Thesis/Engineer_Thesis/Vehicle.cpp File Reference

```
#include "Vehicle.h"
```

5.12 Engineer_Thesis/Engineer_Thesis/Vehicle.h File Reference

```
#include "Vector3D.h"
#include <vector>
```

Classes

· class Vehicle

Class for different spaceship objects.

5.13 Vehicle.h

Go to the documentation of this file.

```
1 #pragma once
2 #include "Vector3D.h"
3 #include <vector>
10 class Vehicle {
12 public:
      std::string name;
       Vector3D position;
Vector3D velocity;
18
       Vector3D engine = { 0,0,0 };
Vector3D force = { 0,0,0 };
2.0
22
       Vector3D displacement = { 0,0,0 };
24
      double mass = 0;
double fuel = 0;
26
30
        double fuel_usage = 0;
32
       double PotentialEnergy = 0;
        double KineticEnergy = 0;
34
38
       bool CalculatedEnergy = 0;
       Vehicle() {};
55
       Vehicle(std::string n, double rx, double ry, double rz, double vx, double vy, double vz, double m,
      double fuel, double fuel_usage);
void Print_info();
59
63
       void User_set();
64 };
```

48 File Documentation

Index

adams	Engineer_Thesis/Engineer_Thesis/Instructions.md, 41
Solver, 17	Engineer_Thesis/Engineer_Thesis/Planet.h, 41, 42
Adams_Bashford	Engineer_Thesis/Engineer_Thesis/Solver.cpp, 42
Solver, 17	Engineer_Thesis/Engineer_Thesis/Solver.h, 42, 43
Add	Engineer_Thesis/Engineer_Thesis/Vector3D.cpp, 44
Vector3D, 29	Engineer_Thesis/Engineer_Thesis/Vector3D.h, 46
ang_velocity	Engineer_Thesis/Engineer_Thesis/Vehicle.cpp, 47
Planet, 12	Engineer_Thesis/Engineer_Thesis/Vehicle.h, 47
	Euler
Calculate_Grav	Solver, 19
Solver, 17	euler
Calculate_Net	Solver, 17
Solver, 18	
CalculatedEnergy	force
Vehicle, 39	Vehicle, 39
Check_Collision	force_data
Solver, 18	Solver, 24
Check_input	fuel
Control, 10	Vehicle, 39
Control, 9	fuel data
Check_input, 10	Solver, 24
Control, 10	fuel_usage
engforce, 10	Vehicle, 39
Print_Interval, 10	fuel_used
timeend, 10	Solver, 24
timestart, 11	,
	G
displacement	Solver, 25
Vehicle, 39	grav_forces
distance	Solver, 25
Solver, 24	
Divide	isOrb
Vector3D, 30	Planet, 13
dvdt	
Solver, 18	kinetic_data
dxdt	Solver, 25
Solver, 19	KineticEnergy
GOIVEI, 10	Vehicle, 40
engforce	
Control, 10	Load_data
engine	Solver, 20
Vehicle, 39	Load_file
engine_data	Solver, 20
Solver, 24	
engine_used	main
Solver, 24	Engineer_Thesis.cpp, 41
Engineer_Thesis.cpp	mass
main, 41	Planet, 13
	Vehicle, 40
Engineer_Thesis/Engineer_Thesis/Engineer_Thesis.cpp,	mass_data
41	Solver, 25

50 INDEX

method Solver, 25 Print_info, 12 radius, 14 Solver, 20 midpoint Planets Solver, 20 solver, 20 start_ang, 14 midpoint Planets Solver, 21 Solver, 25 Populate Solver, 21 Solver, 21 Populate Planet, 12 Populate Planet, 12 Populate Solver, 21 Populate Planet, 13 Solver, 26 Populate Solver, 26 Populate Planet, 13 Solver, 26 Populate Solver, 26 Populate Solver, 26 Populate, 40 Populat		
Solver, 25 Midpoint Solver, 20 midpoint Solver, 20 midpoint Solver, 17 Move_Orbit Solver, 21 Move_Planet Planet, 12 Multiply Vector3D, 30 Operator< Vector3D, 35 Vector3D, 30, 33 Vector3D, 31 Operator Vector3D, 31 Operator Vector3D, 34 Vector3D, 35 Vector3D, 35 Vector3D, 36 Vector3D, 37 Vector3D, 38 Vector3D, 39 Vector3D, 39 Vector3D, 39 Vector3D, 30 Vector3D, 30 Vector3D, 30 Vector3D, 31 Operator Vector3D, 35 Vector3D, 36 Vector3D, 36 Vector3D, 36 Vector3D, 37 Vector3D, 38 Vector3D, 39 Vector3D, 30 Vector3D,	method	position, 14
Midpoint radius, 14 Solver, 20 start_ang, 14 midpoint Planets Solver, 17 Solver, 25 Move_Orbit Populate Solver, 21 Solver, 21 Move_Planet position Planet, 12 Planet, 14 Multiply Vehicle, 40 Vector3D, 30 position_data Solver, 26 Solver, 26 Planet, 13 Solver, 26 Vehicle, 40 Potential_data Planet, 3 Solver, 26 Vehicle, 40 Potential_energy Vehicle, 40 Print_info Ode Print_info Solver, 26 Potential_energy Vehicle, 40 Print_lnergy Ode Print_lnergy Solver, 26 Potential_energy Vector3D, 35 Print_lnergy Vector3D, 35 Print_lnergy Vector3D, 30, 33 Solver, 21 Vector3D, 31 Planet, 14 Vector3D, 31 Planet, 14 Vector3D, 34 Puctoralergy<	Solver, 25	-
Solver, 20 start_ang, 14 Planets Solver, 25		- ·
midpoint	•	
Solver, 17		_ •
Move Orbit Solver, 21 Populate Solver, 21 Move Planet Planet, 12 position Multiply Vector3D, 30 position Nethorized, 40 position data Solver, 26 name potential, data Solver, 26 Planet, 13 Vehicle, 40 PotentialEnergy Vehicle, 40 Ode Solver, 17 PotentialEnergy Vehicle, 40 Ode Solver, 17 Print_info Planet, 12 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 2pp, 46 Print_Pauses Solver, 21 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 31 Solver, 21 Vector3D, 34 Vector3D, 34 Vector3D, 34 Planet, 14 Vector3D, 34 Vector3D, 34 Vector3D, 34 Vector3D, 34 Solver, 22 Vector3D, 34 Vector3D, 34 Vector3D, 34 Vector3D, 34 Solver, 22 Vector3D, 35 Vector3D, 36 Vector3D, 37 Vector3D, 39 Save_data Solver, 22 Operator/= Vector3D, 35 Vector3D, 30 Vector3D, 31 Vector3D, 32 Vector3D, 32	•	
Solver, 21 Move, Planet Planet, 12 Multiply Vector3D, 30 name Planet, 13 Vehicle, 40 Ode Solver, 17 Operator		
Move_Planet, 12 position Planet, 12 Planet, 14 Multiply Vehicle, 40 Vector3D, 30 position_data Solver, 26 potential_data Planet, 13 Solver, 26 Vehicle, 40 PotentialEnergy Ode Vehicle, 40 Solver, 17 Planet, 12 Operator Vehicle, 30 Vector3D, 35 Vector3D, 55 Vector3D, 55 Print_Interval Control, 10 Print_Pauses Vector3D, 0, 33 Solver, 21 Vector3D, 2pp, 44 Push_Back Operator+ Vector3D, 31 Operator+ Planet, 14 Vector3D, 34 Planet, 14 Vector3D, 31 Planet, 14 Operator+ Solver, 22 Vector3D, 34 Planet, 14 Vector3D, 34 Solver, 22 Vector3D, 34 Solver, 22 Vector3D, 35 Save_jan Operator- Solver, 23 Vector3D, 35 Solver, 23 Vector3D, 35 <td< td=""><td>-</td><td>•</td></td<>	-	•
Planet, 12 Multiply Vector3D, 30 Position_data Solver, 26 name Planet, 13 Vehicle, 40 Potential_data Solver, 26 Potential_fata Solver, 26 Potential_fata Solver, 26 PotentialEnergy Vehicle, 40 PotentialEnergy Vehicle, 40 Print_info Solver, 17 Planet, 12 Vector3D, 35 Vector3D, 35 Vector3D, pp, 46 Operator* Vector3D, 30, 33 Vector3D, 2p, 44 Operator* Vector3D, 31 Operator* Vector3D, 34 Vector3D, 34 Vector3D, 34 Vector3D, 34 Vector3D, 31 Operator- Vector3D, 31 Operator- Vector3D, 34 Vector3D, 35 Vector3D, 35 Vector3D, 35 Operator- Vector3D, 35 Solver, 22 Vector3D, 35 Operator- Vector3D, 35 Solver, 22 Operator- Vector3D, 32 Operator- Vector3D, 35 Solver, 22 Operator- Vector3D, 32 Operator- Vector3D, 35 Solver, 23 Solver, 23 Solver, 23 Solver, 23 Operator- Vector3D, 32 Operator- Vector3D, 33 Operator- Vector3D, 34 Operator- Vector3D, 35 Solver, 26 Operator- Vector3D, 35 Solver, 26 Operator- Vector3D, 36 Vector3D, 37 Operator- Vector3D, 36 Operator-		
Multiply Vehicle, 40 Vector3D, 30 position_data Solver, 26 Solver, 26 name potential_data Planet, 13 Solver, 26 Vehicle, 40 PotentialEnergy Vehicle, 40 PotentialEnergy Vehicle, 40 Vehicle, 40 ode Print_into Solver, 17 Planet, 12 Operator Vehicle, 38 Vector3D, 35 Print_Interval Control, 10 Control, 10 Operator* Print_Pauses Vector3D, 30, 33 Solver, 21 Vector3D, 31 Planet, 14 Vector3D, 31 Planet, 14 Vector3D, 34 Vector3D, 31 Vector3D, 34 Vector3D, 34 Vector3D, 34 Vector3D, 35 Vector3D, 35 Save_data Operator/- Solver, 22 Vector3D, 35 Save_Jane Vector3D, 35 Save_Jane Vector3D, 36 Solver, 23 Vector3D, 37 Solver, 23 Vector3D, 39 <	Move_Planet	position
Vector3D, 30 name Planet, 13 Vehicle, 40 Ode Solver, 26 Potential_Energy Vehicle, 40 Ode Solver, 17 Operator< Vector3D, 35 Vector3D, 35 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 31 Operator+ Vector3D, 34 Vector3D, 34 Vector3D, 31 Operator- Vector3D, 31 Operator- Vector3D, 34 Vector3D, 31 Operator- Vector3D, 34 Vector3D, 31 Operator- Vector3D, 35 Vector3D, 31 Operator- Vector3D, 35 Vector3D, 31 Operator- Vector3D, 35 Vector3D, 31 Operator- Vector3D, 35 Vector3D, 36	Planet, 12	Planet, 14
Vector3D, 30 position_data Solver, 26 potential_data Solver, 26 potential_data Planet, 13 Vehicle, 40 Potential_Energy Vehicle, 40 Potential_Energy Vehicle, 40 Print_info Ode Solver, 17 Operator Pint_info Planet, 12 Vehicle, 38 Print_Interval Control, 10 Print_Pauses Vector3D.cpp, 46 Control, 10 Operator ** Vector3D.cpp, 46 Operator ** Vector3D.cpp, 44 Operator ** Print_Pauses Solver, 21 Vector3D.cpp, 44 Operator ** Vector3D.cpp, 44 Operator ** Vector3D, 31 Operator + Vector3D, 34 Vector3D.cpp, 45 Operator ** Planet, 14 runge Solver, 21 Vector3D, 34 Vector3D, 31 Runge_Kutta Solver, 22 Vector3D, 34 Vector3D.cpp, 45 Operator ** Save_data Solver, 22 Vector3D, 34 Vector3D.cpp, 45 Save_json Solver, 22 Vector3D, 31 Save_json Solver, 22 Vector3D, 35 Vector3D.cpp, 45 Solver, 22 Save_Planets Solver, 22 Operator/** Save_json Solver, 22 Solver, 23 Orb_data Ship Solver, 23 Orb_data Ship Solver, 26 Solver, 26 Operator, 26 Solver, 26 Solver, 26 Operator, 26 Solver, 27 Solver, 28 Solver, 29 Solver, 29 Solver, 29 Solver, 29 Solver, 29 Solver, 21 Solver, 21 Solver, 21 Solver, 22 Solver, 23 Orb_data Ship Solver, 24 Solver, 25 Solver, 26 Solver, 26 Solver, 27 Solver, 28 Solver, 29 Solver	Multiply	Vehicle, 40
name Planet, 13 Vehicle, 40 Potential_data Solver, 26 Potential_Energy Vehicle, 40 Potential_Energy Vehicle, 40 Print_info Solver, 17 Operator<< Vector3D, 35 Vector3D, 35 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 31 Operator+ Vector3D, 31 Operator+ Vector3D, 34 Vector3D, 34 Vector3D, 39 Vector3D, 30 Vector3D, 31 Operator- Vector3D, 31 Operator- Vector3D, 31 Operator- Vector3D, 34 Vector3D, 35 Solver, 22 Vector3D, 31 Operator- Vector3D, 31 Save_json Operator/ Vector3D, 35 Solver, 22 Solver, 22 Solver, 22 Solver, 22 Operator/= Vector3D, 35 Solver, 26 Operator/= Solver, 23 Orb_data Planet, 13 Orb_pos Planet, 13 Orb_pos Planet, 13 Orb_pos Planet, 13 Orb_pos Planet, 13 Orb_pos, 12 Orb_pos, 12 Orb_pos, 12 Orb_pos,	• •	position data
name potential_data Planet, 13 Solver, 26 Vehicle, 40 PotentialEnergy ode Vehicle, 40 Solver, 17 Print_info operator Vehicle, 38 Vector3D, 35 Vector3D.cpp, 46 Control, 10 operator** Print_Interval Control, 10 Operator* Print_Pauses Vector3D, 30, 33 Solver, 21 Vector3D.cpp, 44 Print_Back Operator** Solver, 21 Vector3D, 31 Planet, 14 operator+ radius Vector3D, 34 Planet, 14 Vector3D, 31 Planet, 14 operator- Solver, 22 Vector3D, 34 Vector3D, 22 Vector3D, 34 Vector3D, 35 Vector3D, 35 Save_data Operator- Solver, 22 Vector3D, 31 Save_json Operator- Solver, 22 Vector3D, 35 Save_json Vector3D, 35 Sove_jenate Vector3D, 32 Solver, 22	,	•
Planet, 13 Vehicle, 40 Vehicle, 40 Vehicle, 40 Ode Solver, 17 Operator Vector3D, 35 Vector3D, 35 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 31 Operator+ Vector3D, 31 Operator+ Vector3D, 34 Vector3D, 31 Operator- Vector3D, 34 Vector3D, 31 Operator- Vector3D, 34 Vector3D, 35 Vector3D, 31 Operator- Vector3D, 34 Vector3D, 35 Operator- Vector3D, 34 Vector3D, 35 Operator- Vector3D, 35 Operator- Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 32 Operator/ Vector3D, 32 Operator- Vector3D, 35 Vector3D, 36	name	
Vehicle, 40 PotentialEnergy Vehicle, 40 ode Solver, 17 operator Pinnt_inlo Vector3D, 35 Print_Interval Vector3D, 30, 33 Print_Interval Vector3D, 30, 33 Print_Pauses Vector3D, 30, 33 Solver, 21 Vector3D, 31 Push_Back Operator+ radius Vector3D, 34 Planet, 14 Vector3D, 31 Planet, 14 Operator- Solver, 21 Vector3D, 31 Runge_Kutta Operator- Solver, 22 Vector3D, 34 Vector3D, 22 Vector3D, 34 Solver, 22 Vector3D, 34 Solver, 22 Vector3D, 35 Save_data Operator- Solver, 22 Vector3D, 35 Save_json Operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D, 32 Solver, 23 Orb_data Ship Planet, 13 Solver, 24 Orb_pos Solver, 24 Solver, 23	Planet, 13	•
ode		
ode Print_info Solver, 17 Planet, 12 operator Vehicle, 38 Vector3D, 35 Print_Interval Control, 10 Control, 10 operator** Print_Pauses Vector3D, 30, 33 Solver, 21 operator** Push_Back operator+ radius Vector3D, 31 Planet, 14 operator++ Vector3D, 45 operator- Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Vector3D, 22 Vector3D, 34 Save_data operator-= Solver, 22 Vector3D, 34 Save_json operator-= Solver, 22 Vector3D, 31 Save_json operator/- Solver, 22 Vector3D, 35 Save_json Vector3D, 25 Save_json Operator/- Solver, 22 Vector3D, 35 Solver, 22 Vector3D, 35 Solver, 25 Vector3D, 36 Solver, 25 <td>vernole, 40</td> <td>0,</td>	vernole, 40	0,
Solver, 17 operator <	ode	
operator < Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 30, 33 Vector3D, 30, 33 Vector3D, 2cpp, 44 Operator* Vector3D, 31 Operator+ Vector3D, 31 Operator+ Vector3D, 34 Vector3D, 34 Vector3D, 31 Operator- Vector3D, 34 Vector3D, 34 Vector3D, 35 Vector3D, 36 Vector3D, 37 Vector3D, 31 Operator- Vector3D, 31 Operator- Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 35 Vector3D, 32 Operator/ Vector3D, 31 Operator, 22 Solver, 22 Solver, 23 Operator, 24 Vector3D, 31 Operator, 32 Operator, 32 Operator, 32 Operator, 33 Operator, 44 Operator, 45 Operator, 45 Operator, 45 Operator, 45 Operator, 45 Operator, 45 Operator, 47 Operator, 48 Operator, 49 Operator, 49 Operator, 49 Operator, 40 Ope		Print_info
Vector3D, 35 Print_Interval Operator* Control, 10 Vector3D, 30, 33 Solver, 21 Vector3D.cpp, 44 Push_Back Operator*= Solver, 21 Vector3D, 31 Planet, 14 Operator+ radius Vector3D, 34 Planet, 14 Vector3D, 31 Runge_Kutta Operator- Solver, 22 Vector3D, 34 Vector3D, 22 Vector3D, 34 Save_data Solver, 22 Solver, 22 Vector3D, 34 Save_json Operator- Solver, 22 Vector3D, 34 Save_json Vector3D, 35 Save_json Vector3D, 35 Save_Planets Vector3D, 35 Solver, 22 Vector3D, 32 Solver, 23 Operator/= Setup Vector3D, 32 Solver, 23 Orb_data Solver, 26 Planet, 13 Solver, 26 Orb_pos Planet, 13 Adams_ Bashford, 17 Planet, 11 Calculate_ Grav, 17 Calculate_ Grav, 17 </td <td>•</td> <td>Planet, 12</td>	•	Planet, 12
Vector3D.cpp, 46 Control, 10 operator* Print_Pauses Vector3D, 30, 33 Solver, 21 vector3D.cpp, 44 Push_Back operator*= Vector3D, 31 operator+ radius Vector3D.cpp, 45 runge operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Vector3D, 22 Vector3D, 34 Solver, 22 Vector3D, 34 Solver, 22 Vector3D, 35 Save_data operator/- Solver, 22 Vector3D, 25 Save_Planets Vector3D, 35 Solver, 22 Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 23 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 24 Planet, 11 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18	•	Vehicle, 38
operator*		Print Interval
Operator* Print_Pauses Vector3D, 30, 33 Solver, 21 Operator*= Vector3D, 31 Operator+ radius Vector3D, 34 Planet, 14 Vector3D, 2pp, 45 runge Operator-= Solver, 17 Vector3D, 31 Runge_Kutta Operator- Vector3D, 22 Vector3D, 34 Save_data Operator-= Solver, 22 Vector3D, 31 Save_json Operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D, 35 Solver, 22 Operator/= Setup Vector3D, 32 Solver, 23 Orb_data Ship Planet, 13 Solver, 26 Orb_pos Solve Planet, 13 Solver, 23 Orb_radius Solver, 24 Planet, 13 Calculate_Grav, 17 Planet, 11 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 Move_Planet, 12	Vector3D.cpp, 46	_
Vector3D, 30, 33 Solver, 21 Vector3D.cpp, 44 Push_Back operator*= Vector3D, 31 operator+ radius Vector3D, 34 Planet, 14 Vector3D.cpp, 45 runge operator- Solver, 17 Vector3D, 31 Runge_Kutta operator- Vector3D, 34 Vector3D.cpp, 45 Save_data operator- Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 26 orb_radius Solver, 23 Planet, 11 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 name, 13 dvdt, 18 orb_odata, 13 engine_lused	operator*	
Vector3D.cpp, 44 Push_Back operator**= Solver, 21 Vector3D, 31 radius Vector3D, 34 Planet, 14 Vector3D.cpp, 45 runge operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Solver, 22 Vector3D, 35 Save_data operator/= Solver, 22 Vector3D, 35 Save_Planets Vector3D, 35 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 23 orb_pos Solver, 26 Planet, 13 Solver, 23 orb_radius Solver, 24 Planet, 11 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvvt, 18 dvxt, 19 mae, 13 orb_data, 13 engine_data, 24 orb_radius, 13 engine_used, 24	Vector3D, 30, 33	-
operator*=	Vector3D.cpp, 44	
Vector3D, 31 radius operator+ radius Vector3D, 34 Planet, 14 vector3D.cpp, 45 runge operator-= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Vector3D.cpp, 45 operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 14 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 18 orb_data, 13 orb_data, 13 orb_pos, 13 orb_radius, 13	• •	-
operator+ radius Vector3D, 34 Planet, 14 vector3D.cpp, 45 runge operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 23 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 14 planet, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 Move_Planet, 12 dvdt, 18 name, 13 dvdt, 18 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 2	•	Solver, 21
Vector3D, 34 Planet, 14 Vector3D.cpp, 45 runge operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Save_data Operator-= Solver, 22 Vector3D, 31 Save_json Operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 Operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solve Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 11 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 Move_Planet, 12 dvdt, 18 name, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	•	radius
Vector3D.cpp, 45 runge operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 34 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 23 Planet, 13 Solver, 24 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 adams, 13 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 18 dxdt, 19 orb_data, 13 engine_data, 24 orb_radius, 13 engine_used, 24	•	
operator+= Solver, 17 Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D, 24 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 23 Planet, 13 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 18 dxdt, 19 orb_pos, 13 engine_data, 24 orb_radius, 13 engine_used, 24		,
Vector3D, 31 Runge_Kutta operator- Solver, 22 Vector3D.cpp, 45 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 Adams_Bashford, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 18 orb_pos, 13 engine_data, 24 orb_radius, 13 engine_used, 24	• •	•
operator-	•	Solver, 17
Vector3D, 34 Vector3D.cpp, 45 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	Vector3D, 31	Runge_Kutta
Vector3D.cpp, 45 Save_data operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solve Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	operator-	Solver, 22
operator-= Solver, 22 Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solve Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dvdt, 18 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	Vector3D, 34	
Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solve Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	Vector3D.cpp, 45	Save_data
Vector3D, 31 Save_json operator/ Solver, 22 Vector3D, 35 Save_Planets Vector3D.cpp, 45 Solver, 22 operator/= Setup Vector3D, 32 Solver, 23 orb_data Ship Planet, 13 Solver, 26 orb_pos Solver Planet, 13 Solver, 23 orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	operator-=	Solver, 22
operator/	•	Save ison
Vector3D, 35 Vector3D.cpp, 45 Solver, 22 operator/= Vector3D, 32 Orb_data Planet, 13 Orb_pos Planet, 13 Planet, 13 Planet, 13 Planet, 13 Planet, 13 Planet, 13 Planet, 11 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 mass, 13 Move_Planet, 12 name, 13 Orb_data, 13 Orb_pos, 13 Orb_padata, 13		-
Vector3D.cpp, 45 operator/= Vector3D, 32 orb_data Planet, 13 orb_pos Planet, 13 orb_radius Planet, 11 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Solver, 23 Solver, 23 Solver, 23 Solver, 23 Solver, 23 Solver, 14 adams, 17 Calculate_Grav, 17 Calculate_Crav, 17 Calculate_Net, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19	•	
operator/= Vector3D, 32 orb_data Planet, 13 orb_pos Planet, 13 orb_radius Planet, 11 Orb_cata, 13 Orb_cata, 13 Orb_data, 13 Orb_pos, 13 Orb_radius Orb_radius Orb_radius Orb_radius Orb_radius Orb_radius Orb_radius Orb_radius, 13 Orb_radius Solver, 23 Solver, 23 Solver, 23 Solver, 23 Solver, 23 Solver, 23 Colver, 23 Orb_radius, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Orbeck_Collision, 18 dxdt, 19 engine_data, 24 engine_used, 24 engine_used, 24 Euler, 19		-
Vector3D, 32 orb_data Planet, 13 Solver, 26 orb_pos Planet, 13 Solver, 23 orb_radius Planet, 13 orb_radius Planet, 13 Orb_radius Planet, 13 Orb_radius Planet, 13 Planet, 11 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Solver, 23 Solver, 26 Solve Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 distance, 24 dvdt, 18 dvdt, 19 engine_data, 24 engine_used, 24 Euler, 19	• •	
orb_data Planet, 13 Solver, 26 Solve Planet, 13 Solver, 23 orb_radius Planet, 13 Planet, 13 Planet, 11 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Solver, 23 Solver, 24 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 19 engine_data, 24 engine_used, 24 engine_used, 24 Euler, 19	•	-
Planet, 13 orb_pos Planet, 13 orb_radius Planet, 13 orb_radius Planet, 13 Planet, 13 Planet, 11 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_pos, 13 orb_radius, 13 Solver, 26 Solve Solver, 23 Solver, 23 Solver, 23 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19		· ·
orb_pos Planet, 13 Solver, 23 orb_radius Planet, 13 Solver, 14 Planet, 13 Adams_Bashford, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Solve Solve, 23 Solver, 23 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 distance, 24 dvdt, 18 dvdt, 18 engine_data, 24 engine_data, 24 engine_used, 24 Euler, 19	orb_data	Ship
Planet, 13 orb_radius Planet, 13 Solver, 23 Solver, 14 Planet, 13 Adams_Bashford, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Solver, 23 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 19 engine_data, 24 engine_data, 24 engine_used, 24 Euler, 19	Planet, 13	Solver, 26
orb_radius Solver, 14 Planet, 13 adams, 17 Adams_Bashford, 17 Planet, 11 Calculate_Grav, 17 ang_velocity, 12 Calculate_Net, 18 isOrb, 13 Check_Collision, 18 mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dvdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	orb_pos	Solve
Planet, 13 adams, 17 Adams_Bashford, 17 Planet, 11 ang_velocity, 12 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19	Planet, 13	Solver, 23
Planet, 13 adams, 17 Adams_Bashford, 17 Planet, 11 ang_velocity, 12 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 adams, 17 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19	orb radius	Solver, 14
Planet, 11 ang_velocity, 12 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Adams_Bashford, 17 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dvdt, 19 engine_data, 24 engine_used, 24 Euler, 19	-	
Planet, 11 ang_velocity, 12 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Calculate_Grav, 17 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19		
ang_velocity, 12 isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Calculate_Net, 18 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19	Planet, 11	-
isOrb, 13 mass, 13 Move_Planet, 12 name, 13 orb_data, 13 orb_pos, 13 orb_radius, 13 Check_Collision, 18 distance, 24 dvdt, 18 dxdt, 19 engine_data, 24 engine_used, 24 Euler, 19		- -
mass, 13 distance, 24 Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19		-
Move_Planet, 12 dvdt, 18 name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19		-
name, 13 dxdt, 19 orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19		
orb_data, 13 engine_data, 24 orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	— · · · · · · · · · · · · · · · · · · ·	dvdt, 18
orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	name, 13	dxdt, 19
orb_pos, 13 engine_used, 24 orb_radius, 13 Euler, 19	orb_data, 13	engine data, 24
orb_radius, 13 Euler, 19	orb_pos, 13	- -
	·	<u> </u>
Guici, 17		
	,	Guior, 17

INDEX 51

force_data, 24	Vehicle, 38
fuel_data, 24	
fuel_used, 24	Validate_Json
G, 25	Solver, 23
grav_forces, 25	Vector3D, 27
kinetic_data, 25	Add, 29
Load_data, 20	Divide, 30
Load_file, 20	Multiply, 30
mass_data, 25	operator<<, 35
method, 25	operator*, 30, 33
Midpoint, 20	operator*=, 31
midpoint, 17	operator+, 34
Move_Orbit, 21	operator+=, 31
ode, 17	operator-, 34
Planets, 25	operator-=, 31
Populate, 21	operator/, 35
•	operator/=, 32
position_data, 26	Subtract, 32
potential_data, 26	Vector3D, 29
Print_Pauses, 21	VectorsEqual, 33
Push_Back, 21	•
runge, 17	x, 35
Runge_Kutta, 22	y, 36
Save_data, 22	z, 36
Save_json, 22	Zero, 33
Save_Planets, 22	Vector3D.cpp
Setup, 23	operator<<, 46
Ship, 26	operator*, 44
Solve, 23	operator+, 45
step, 26	operator-, 45
T, 26	operator/, 45
time, 26	VectorsEqual
time_data, 27	Vector3D, 33
TimeVect, 27	Vehicle, 36
UseEngine, 23	CalculatedEnergy, 39
Validate Json, 23	displacement, 39
velocity_data, 27	engine, 39
start ang	force, 39
Planet, 14	fuel, 39
step	fuel usage, 39
Solver, 26	KineticEnergy, 40
Subtract	mass, 40
Vector3D, 32	name, 40
Vector3D, 32	position, 40
Т	PotentialEnergy, 40
Solver, 26	Print info, 38
time	User set, 38
	_ ·
Solver, 26	Vehicle, 37
time_data	velocity, 40
Solver, 27	velocity
timeend	Vehicle, 40
Control, 10	velocity_data
timestart	Solver, 27
Control, 11	
TimeVect	X
Solver, 27	Vector3D, 35
UseEngine	y
Solver, 23	Vector3D, 36
User_set	

52 INDEX

z Vector3D, 36 Zero Vector3D, 33