Software Requirements Specification for Game Physics

Alex Halliwushka, Luthfi Mawarid, and Olu Owojaiye

September 14, 2019

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

The unit system used throughout is SI (Système International d'Unités). In addition to the basic units, several derived units are also used. For each unit, Tab: ToU lists the symbol, a description and the SI name.

Symbol	Description	SI Name
J	energy	joule
kg	mass	kilogram
m	length	metre
N	force	newton
rad	angle	radian
s	time	second

Table 1

1.2 Table of Symbols

The symbols used in this document are summarized in Tab: ToS along with their units. Throughout the document, symbols in bold will represent vectors, and scalars otherwise. The symbols are listed in alphabetical order. For vector quantities, the units shown are for each component of the vector.

Symbol	Description	Units
$\mathbf{a}$	Acceleration	$\frac{\text{m}}{\text{s}^2}$
$\mathbf{a}(t)$	Linear Acceleration	$\frac{\text{m}}{\text{s}^2}$
$\mathbf{a}_i$	The I-Th Body's Acceleration	$\frac{\text{m}}{\text{s}^2}$
C_{R}	Coefficient of restitution	–
$\mathbf{F}$	Force	N
$\mathbf{F}_1$	Force exerted by the first body (on another body)	N
$\mathbf{F}_2$	Force exerted by the second body (on another body)	N
$\mathbf{F}_i$	Force Applied to the I-Th Body at Time T	N
G	Gravitational constant	$\frac{\text{m}^3}{(\text{kgs}^2)}$
g	Gravitational acceleration	$\frac{\text{m}}{\text{s}^2}$
h	Height	m

Symbol	Description	Units
$\mathbf{I}$	Moment of inertia	kgm^2
$\mathbf{I}_{\text{A}}$	Moment of Inertia of Rigid Body A	kgm^2
$\mathbf{I}_{\text{B}}$	Moment of Inertia of Rigid Body B	kgm^2
$\mathbf{J}$	Impulse (vector)	Ns
j	Impulse (scalar)	Ns
KE	Kinetic energy	J
L	Length	m
M	Mass of the Larger Rigid Body	kg
m	Mass	kg
m_1	Mass of the first body	kg
m_2	Mass of the second body	kg
m_{A}	Mass of Rigid Body A	kg
m_{B}	Mass of Rigid Body B	kg
m_j	Mass of the J-Th Particle	kg
m_T	Total Mass of the Rigid Body	kg
$\mathbf{n}$	Collision Normal Vector	m
PE	Potential energy	J
$\mathbf{p}$	Position	m
$\mathbf{p}_{\text{CM}}$	Center of Mass	m
$\mathbf{p}_j$	Position Vector of the J-Th Particle	m
r_j	Distance Between the J-Th Particle and the Axis of Rotation	m
$\mathbf{r}$	Displacement	m
$\mathbf{r}(t)$	Linear Displacement	m
$\mathbf{r}_{\text{OB}}$	Displacement vector between the origin and point B	m
$\mathbf{r}$	Displacement unit vector	m
t	Time	s
t_{c}	Denotes the time at collision	s
$\mathbf{v}$	Velocity	$\frac{\text{m}}{\text{s}}$
$\Delta\mathbf{v}$	Change in velocity	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}(t)$	Linear Velocity	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}^{\text{AP}}$	Velocity of the Point of Collision P in Body A	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}^{\text{BP}}$	Velocity of the Point of Collision P in Body B	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_1$	Velocity of the First Body	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_2$	Velocity of the Second Body	$\frac{\text{m}}{\text{s}}$

Symbol	Description	Units
$\mathbf{v}_\text{A}$	Velocity at Point A	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_\text{B}$	Velocity at Point B	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_\text{f}{}^\text{AB}$	Final Relative Velocity Between Rigid Bodies of A and B	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_i$	Velocity of the I-Th Body's Velocity	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_\text{i}{}^\text{AB}$	Initial Relative Velocity Between Rigid Bodies of A and B	$\frac{\text{m}}{\text{s}}$
$\mathbf{v}_\text{O}$	Velocity at Point Origin	$\frac{\text{m}}{\text{s}}$
$\|\|\mathbf{n}\|\|$	Length of the Normal Vector	m
$\|\|\mathbf{r}\|\|$	Euclidean norm of the displacement	m
$\|\|\mathbf{r}_\text{AP}{}^*\mathbf{n}\|\|$	Length of the Perpendicular Vector to the Contact Displacement Vector of Rigid Body A	m
$\|\|\mathbf{r}_\text{BP}{}^*\mathbf{n}\|\|$	Length of the Perpendicular Vector to the Contact Displacement Vector of Rigid Body B	m
$\|\|\mathbf{r}\|\|^2$	Squared distance	m^2
α	Angular Acceleration	$\frac{\text{rad}}{\text{s}^2}$
θ	Angular Displacement	rad
τ	Torque	Nm
τ_i	Torque applied to the i-th body	Nm
ω	Angular Velocity	$\frac{\text{rad}}{\text{s}}$
ϕ	Orientation	rad

Table 2

1.3 Abbreviations and Acronyms

Abbreviation	Full Form
2D	Two-Dimensional
3D	Three-Dimensional
A	Assumption
CM	Centre of Mass
DD	Data Definition
GD	General Definition
GS	Goal Statement
Game Physics	game physics library
IM	Instance Model
LC	Likely Change

Abbreviation	Full Form
ODE	Ordinary Differential Equation
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
UC	Unlikely Change
Uncert.	Typical Uncertainty

Table 3

2 Introduction

Due to the rising cost of developing video games, developers are looking for ways to save time and money for their projects. Using an open source physics library that is reliable and free will cut down development costs and lead to better quality products.

The following section provides an overview of the Software Requirements Specification (SRS) for Game Physics. This section explains the purpose of this document, the scope of the system, the characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

This document describes the modeling of an open source 2D rigid body physics library used for games. The theoretical models and goal statements used in Game Physics are provided. This document is intended to be used as a reference to provide all necessary information to understand and verify the model.

This document will be used as a starting point for subsequent development phases, including writing the design specification and the software verification and validation plan. The design document will show how the requirements are to be realized, including decisions on the numerical algorithms and programming environment. The verification and validation plan will show the steps that will be used to increase confidence in the software documentation and the implementation. Although the SRS fits in a series of documents that follow the so-called waterfall model, the actual development process is not constrained in any way. Even when the waterfall model is not followed, as Parnas and Clements point out [8], the most logical way to present the documentation is still to “fake” a rational design process.

2.2 Scope of Requirements

The scope of the requirements includes the physical simulation of 2D rigid bodies acted on by forces.

Figure 1: System Context

2.3 Characteristics of Intended Reader

Reviewers of this documentation should have an understanding of rigid body dynamics and high school calculus. The users of Game Physics can have a lower level of expertise, as explained in Section: User Characteristics.

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by [7] and [9]. The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the instance models in Section: Instance Models and trace back to find any additional information they require.

The goal statements (Section: Goal Statements) are refined to the theoretical models and the theoretical models (Section: Theoretical Models) to the instance models (Section: Instance Models).

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics, and lists the system constraints.

3.1 System Context

Fig:sysCtxDiag shows the system context. A circle represents an external entity outside the software, the user in this case. A rectangle represents the software system itself (Game Physics). Arrows are used to show the data flow between the system and its environment.

The interaction between the product and the user is through an application programming interface. The responsibilities of the user and the system are as follows:

- User Responsibilities
  - Provide initial conditions of the physical state of the simulation, rigid bodies present, and forces applied to them.

- Ensure application programming interface use complies with the user guide.
  - Ensure required software assumptions (FIXME REF) are appropriate for any particular problem the software addresses.

- Game Physics Responsibilities
  - Determine if the inputs and simulation state satisfy the required physical and system constraints (FIXME REF).
  - Calculate the new state of all rigid bodies within the simulation at each simulation step.
  - Provide updated physical state of all rigid bodies at the end of a simulation step.

3.2 User Characteristics

The end user of Game Physics should have an understanding of first year programming concepts and an understanding of high school physics.

3.3 System Constraints

There are no system constraints.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, and definitions that are used.

4.1 Problem Description

A system is needed to create a simple, lightweight, fast, and portable 2D rigid body physics library, which will allow for more accessible game development and the production of higher quality products. Creating a gaming physics library is a difficult task. Games need physics libraries that simulate objects acting under various physical conditions, while simultaneously being fast and efficient enough to work in soft real-time during the game. Developing a physics library from scratch takes a long period of time and is very costly, presenting barriers of entry which make it difficult for game developers to include physics in their products. There are a few free, open source and high quality physics libraries available to be used for consumer products (Section: Off-The-Shelf Solutions).

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements.

- Rigid body: A solid body in which deformation is neglected.
- Elasticity: The ratio of the relative velocities of two colliding objects after and before a collision.
- Centre of mass: The mean location of the distribution of mass of the object.
- Cartesian coordinate system: A coordinate system that specifies each point uniquely in a plane by a set of numerical coordinates, which are the signed distances to the point from two fixed perpendicular oriented lines, measured in the same unit of length (from [2]).
- Right-handed coordinate system: A coordinate system where the positive z-axis comes out of the screen..
- Line: An interval between two points (from [4]).
- Point: An exact location, it has no size, only position (from [10]).

4.1.2 Goal Statements

Given the kinematic properties, and forces including any (collision forces) applied on a set of rigid bodies, the goal statements are:

-Linear-Properties: Determine their new positions and velocities over a period of time.

ngular-Properties: Determine their new orientations and angular velocities over a period of time.

4.2 Solution Characteristics Specification

The instance models that govern Game Physics are presented in Section: Instance Models. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical models by filling in the missing information for the physical system. The assumptions refine the scope by providing more detail.

objectTy: All objects are rigid bodies. (RefBy: DD: chaslesThm, DD: reVeInColl, IM: transMot, IM: rotMot, DD: potEnergy, DD: ctrOfMass, DD: momentOfInertia, DD: linVel, DD: linDisp, DD: linAcc, DD: kEnergy, DD: impulseV, GD: impulse, IM: col2D, DD: angVel, DD: angDisp, and DD: angAccel.)

objectDimension: All objects are 2D. (RefBy: IM: transMot, IM: rotMot, DD: potEnergy, TM: NewtonSecLawRotMot, DD: kEnergy, GD: impulse, IM: col2D, DD: angVel, DD: angDisp, and DD: angAccel.)

ordinateSystemTy: The library uses a Cartesian coordinate system.

axesDefined: The axes are defined using right-handed coordinate system. (RefBy: IM: rotMot, GD: impulse, and IM: col2D.)

collisionType: All rigid bodies collisions are vertex-to-edge collisions. (RefBy: LC: Expanded-Collisions, GD: impulse, and IM: col2D.)

mpingInvolvement: There is no damping involved throughout the simulation and this implies that there are no friction forces. (RefBy: IM: transMot, DD: potEnergy, LC: Include-Dampening, DD: kEnergy, and IM: col2D.)

JointsInvolvement: There are no constraints and joints involved throughout the simulation. (RefBy: IM: transMot, LC: Include-Joints-Constraints, and IM: col2D.)

4.2.2 Theoretical Models

This section focuses on the general equations and laws that Game Physics is based on.

Refname	**TM:NewtonSecLawMot**
Label	Newton's second law of motion
Equation	$$\mathbf{F} = m\mathbf{a}$$
Description	$\mathbf{F}$ is the force (N) m is the mass (kg) $\mathbf{a}$ is the acceleration ($\frac{\text{m}}{\text{s}^2}$)
Notes	The net force $\mathbf{F}$ on a body is proportional to the acceleration $\mathbf{a}$ of the body, where m denotes the mass of the body as the constant of proportionality.
Source	–
RefBy	IM: transMot

Refname	TM:NewtonThirdLawMot
Label	Newton's third law of motion
Equation	$$\mathbf{F}_1 = -\mathbf{F}_2$$
Description	$\mathbf{F}_1$ is the force exerted by the first body (on another body) (N) $\mathbf{F}_2$ is the force exerted by the second body (on another body) (N)
Notes	Every action has an equal and opposite reaction. In other words, the force $\mathbf{F}_1$ exerted on the second rigid body by the first is equal in magnitude and in the opposite direction to the force $\mathbf{F}_2$ exerted on the first rigid body by the second.
Source	–
RefBy	

Refname	TM:UniversalGravLaw
Label	Newton's law of universal gravitation
Equation	$$\mathbf{F} = G\frac{m_1 m_2}{\|\|\mathbf{r}\|\|^2}\mathbf{r} = G\frac{m_1 m_2}{\|\|\mathbf{r}\|\|^2}\frac{\mathbf{r}}{\|\|\mathbf{r}\|\|}$$
Description	$\mathbf{F}$ is the force (N) G is the gravitational constant ($\frac{\text{m}^3}{(\text{kgs}^2)}$) m_1 is the mass of the first body (kg) m_2 is the mass of the second body (kg) $\|\|\mathbf{r}\|\|$ is the Euclidean norm of the displacement (m) $\mathbf{r}$ is the displacement unit vector (m) $\mathbf{r}$ is the displacement (m)
Notes	Two rigid bodies in the universe attract each other with a force $\mathbf{F}$ that is directly proportional to the product of their masses, m_1 and m_2, and inversely proportional to the squared distance $\|\|\mathbf{r}\|\|^2$ between them. The vector $\mathbf{r}$ is the displacement between the centres of the rigid bodies and $\|\|\mathbf{r}\|\|$ is the absolute distance between the two. $\mathbf{r}$ is equivalent to the displacement divided by the Euclidean norm of the displacement, as shown above.
Source	–
RefBy	

Refname	TM:NewtonSecLawRotMot
Label	Newton's second law for rotational motion
Equation	$$\tau = \mathbf{I}\alpha$$
Description	τ is the torque (Nm) $\mathbf{I}$ is the moment of inertia (kgm^2) α is the angular acceleration ($\frac{\text{rad}}{\text{s}^2}$)
Notes	The net torque τ on a rigid body is proportional to its angular acceleration α, where $\mathbf{I}$ denotes the moment of inertia of the rigid body as the constant of proportionality. We also assume that all rigid bodies involved are two-dimensional (from A: objectDimension).
Source	–
RefBy	IM: rotMot

4.2.3 General Definitions

This section collects the laws and equations that will be used to build the instance models.

Refname	**GD:accelGravity**
Label	Acceleration due to gravity
Units	$\frac{\text{m}}{\text{s}^2}$
Equation	$$g = \frac{GM}{\lvert\lvert\mathbf{r}\rvert\rvert^2}\mathbf{r}$$
Description	g is the gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$) G is the gravitational constant ($\frac{\text{m}^3}{(\text{kgs}^2)}$) M is the mass of the larger rigid body (kg) $\lvert\lvert\mathbf{r}\rvert\rvert$ is the Euclidean norm of the displacement (m) $\mathbf{r}$ is the displacement unit vector (m)
Source	Definition of Gravitational Acceleration
RefBy	IM: transMot

Refname	GD:impulse
Label	Impulse for Collision
Units	Ns
Equation	$$j = \frac{-\left(1+C_{\text{R}}\right)\mathbf{v}_{\text{i}}{}^{\text{AB}}\cdot\mathbf{n}}{\left(\frac{1}{m_{\text{A}}}+\frac{1}{m_{\text{B}}}\right)\|\mathbf{n}\|^{2}+\frac{\|\mathbf{r}_{\text{AP}}{*}\mathbf{n}\|^{2}}{\mathbf{I}_{\text{A}}}+\frac{\|\mathbf{r}_{\text{BP}}{*}\mathbf{n}\|^{2}}{\mathbf{I}_{\text{B}}}}$$
Description	j is the impulse (scalar) (Ns) C_{R} is the coefficient of restitution (Unitless) $\mathbf{v}_{\text{i}}{}^{\text{AB}}$ is the initial relative velocity between rigid bodies of A and B ($\frac{\text{m}}{\text{s}}$) $\mathbf{n}$ is the collision normal vector (m) m_{A} is the mass of rigid body A (kg) m_{B} is the mass of rigid body B (kg) $\|\mathbf{n}\|$ is the length of the normal vector (m) $\|\mathbf{r}_{\text{AP}}{*}\mathbf{n}\|$ is the length of the perpendicular vector to the contact displacement vector of rigid body A (m) $\mathbf{I}_{\text{A}}$ is the moment of inertia of rigid body A (kgm^2) $\|\mathbf{r}_{\text{BP}}{*}\mathbf{n}\|$ is the length of the perpendicular vector to the contact displacement vector of rigid body B (m) $\mathbf{I}_{\text{B}}$ is the moment of inertia of rigid body B (kgm^2)
Notes	All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension). A right-handed coordinate system is used (from A: axesDefined). All collisions are vertex-to-edge (from A: collisionType).
Source	Impulse for Collision Ref
RefBy	IM: col2D

4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models.

Refname	**DD:ctrOfMass**
Label	Center of Mass
Symbol	$\mathbf{p}_{\text{CM}}$
Units	m
Equation	$$\mathbf{p}_{\text{CM}} = \frac{\sum m_j \mathbf{p}_j}{m_T}$$
Description	$\mathbf{p}_{\text{CM}}$ is the Center of Mass (m) m_j is the mass of the j-th particle (kg) $\mathbf{p}_j$ is the position vector of the j-th particle (m) m_T is the total mass of the rigid body (kg)
Notes	All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	IM: transMot and IM: col2D

Refname	**DD:linDisp**
Label	Linear Displacement
Symbol	$\mathbf{r}(t)$
Units	m
Equation	$$\mathbf{r}(t) = \frac{d\mathbf{p}\,(t)}{dt}$$
Description	$\mathbf{r}(t)$ is the linear displacement (m) t is the time (s) $\mathbf{p}$ is the position (m)
Notes	All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	IM: transMot

Refname	**DD:linVel**
Label	Linear Velocity
Symbol	$\mathbf{v}(t)$
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{v}(t) = \frac{d\mathbf{r}\,(t)}{dt}$$
Description	$\mathbf{v}(t)$ is the linear velocity ($\frac{\text{m}}{\text{s}}$) t is the time (s) $\mathbf{r}$ is the displacement (m)
Notes	All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	IM: transMot

Refname	**DD:linAcc**
Label	Linear Acceleration
Symbol	$\mathbf{a}(t)$
Units	$\frac{\text{m}}{\text{s}^2}$
Equation	$$\mathbf{a}(t) = \frac{d\mathbf{v}\,(t)}{dt}$$
Description	$\mathbf{a}(t)$ is the linear acceleration ($\frac{\text{m}}{\text{s}^2}$) t is the time (s) $\mathbf{v}$ is the velocity ($\frac{\text{m}}{\text{s}}$)
Notes	All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	IM: transMot

Refname	**DD:angDisp**
Label	Angular Displacement
Symbol	θ
Units	rad
Equation	$$\theta = \frac{d\phi\,(t)}{dt}$$
Description	θ is the angular displacement (rad) t is the time (s) ϕ is the orientation (rad)
Notes	All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension).
Source	–
RefBy	IM: rotMot

Refname	**DD:angVel**
Label	Angular Velocity
Symbol	ω
Units	$\frac{\text{rad}}{\text{s}}$
Equation	$$\omega = \frac{d\theta\left(t\right)}{dt}$$
Description	ω is the angular velocity ($\frac{\text{rad}}{\text{s}}$) t is the time (s) θ is the angular displacement (rad)
Notes	All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension).
Source	–
RefBy	IM: rotMot

Refname	DD:angAccel
Label	Angular Acceleration
Symbol	α
Units	$\frac{\text{rad}}{\text{s}^2}$
Equation	$$\alpha = \frac{d\omega\left(t\right)}{dt}$$
Description	α is the angular acceleration ($\frac{\text{rad}}{\text{s}^2}$) t is the time (s) ω is the angular velocity ($\frac{\text{rad}}{\text{s}}$)
Notes	All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension).
Source	–
RefBy	IM: rotMot

Refname	**DD:chaslesThm**
Label	Chasles' theorem
Symbol	$\mathbf{v}_{\text{B}}$
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{v}_{\text{B}} = \mathbf{v}_{\text{O}} + \omega \times \mathbf{r}_{\text{OB}}$$
Description	$\mathbf{v}_{\text{B}}$ is the velocity at point B ($\frac{\text{m}}{\text{s}}$) $\mathbf{v}_{\text{O}}$ is the velocity at point origin ($\frac{\text{m}}{\text{s}}$) ω is the angular velocity ($\frac{\text{rad}}{\text{s}}$) $\mathbf{r}_{\text{OB}}$ is the displacement vector between the origin and point B (m)
Notes	The linear velocity $\mathbf{v}_{\text{B}}$ of any point B in a rigid body is the sum of the linear velocity $\mathbf{v}_{\text{O}}$ of the rigid body at the origin (axis of rotation) and the resultant vector from the cross product of the rigid body's angular velocity ω and the displacement vector between the origin and point B $\mathbf{r}_{\text{OB}}$. All bodies are assumed to be rigid (from A: objectTy).
Source	[3]
RefBy	

Refname	**DD:torque**
Label	Torque
Symbol	τ
Units	Nm
Equation	$$\tau = \mathbf{r} \times \mathbf{F}$$
Description	τ is the torque (Nm) $\mathbf{r}$ is the displacement (m) $\mathbf{F}$ is the force (N)
Notes	The torque on a body measures the the tendency of a force to rotate the body around an axis or pivot.
Source	–
RefBy	

Refname	**DD:kEnergy**
Label	Kinetic energy
Symbol	KE
Units	J
Equation	$$KE = \frac{m\mathbf{v}^2}{2}$$
Description	KE is the kinetic energy (J) m is the mass (kg) $\mathbf{v}$ is the velocity ($\frac{\text{m}}{\text{s}}$)
Notes	Kinetic energy is the measure of the energy a body possesses due to its motion. All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension). No damping occurs during the simulation (from A: dampingInvolvement).
Source	–
RefBy	

Refname	**DD:coeffRestitution**
Label	Coefficient of restitution
Symbol	C_{R}
Units	Unitless
Equation	$$C_{\text{R}} = -\left(\frac{\mathbf{v}_{\text{f}}{}^{\text{AB}} \cdot \mathbf{n}}{\mathbf{v}_{\text{i}}{}^{\text{AB}} \cdot \mathbf{n}}\right)$$
Description	C_{R} is the coefficient of restitution (Unitless) $\mathbf{v}_{\text{f}}{}^{\text{AB}}$ is the final relative velocity between rigid bodies of A and B ($\frac{\text{m}}{\text{s}}$) $\mathbf{n}$ is the collision normal vector (m) $\mathbf{v}_{\text{i}}{}^{\text{AB}}$ is the initial relative velocity between rigid bodies of A and B ($\frac{\text{m}}{\text{s}}$)
Notes	The coefficient of restitution C_{R} determines the elasticity of a collision between two rigid bodies. $C_{\text{R}} = 1$ results in an elastic collision, $C_{\text{R}} < 1$ results in an inelastic collision, and $C_{\text{R}} = 0$ results in a totally inelastic collision.
Source	–
RefBy	

Refname	DD:reVeInColl
Label	Initial Relative Velocity Between Rigid Bodies of A and B
Symbol	$\mathbf{v}_{\mathrm{i}}{}^{\mathrm{AB}}$
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$$\mathbf{v}_{\mathrm{i}}{}^{\mathrm{AB}} = \mathbf{v}^{\mathrm{AP}} - \mathbf{v}^{\mathrm{BP}}$$
Description	$\mathbf{v}_{\mathrm{i}}{}^{\mathrm{AB}}$ is the initial relative velocity between rigid bodies of A and B ($\frac{\mathrm{m}}{\mathrm{s}}$) $\mathbf{v}^{\mathrm{AP}}$ is the velocity of the point of collision P in body A ($\frac{\mathrm{m}}{\mathrm{s}}$) $\mathbf{v}^{\mathrm{BP}}$ is the velocity of the point of collision P in body B ($\frac{\mathrm{m}}{\mathrm{s}}$)
Notes	In a collision, the velocity of a rigid body A colliding with another rigid body B relative to that body $\mathbf{v}_{\mathrm{i}}{}^{\mathrm{AB}}$ is the difference between the velocities of A and B at point P. All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	

Refname	**DD:impulseV**
Label	Impulse (vector)
Symbol	$\mathbf{J}$
Units	Ns
Equation	$$\mathbf{J} = m\Delta\mathbf{v}$$
Description	$\mathbf{J}$ is the impulse (vector) (Ns) m is the mass (kg) $\Delta\mathbf{v}$ is the change in velocity ($\frac{\text{m}}{\text{s}}$)
Notes	An impulse (vector) $\mathbf{J}$ occurs when a force $\mathbf{F}$ acts over a body over an interval of time. All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	

Detailed derivation of impulse (vector): Newton's second law of motion states:

$$\mathbf{F} = m\mathbf{a} = m\frac{d\mathbf{v}}{dt}$$

Rearranging :

$$\int_{t_1}^{t_2} \mathbf{F}\,dt = m\left(\int_{\mathbf{v}_1}^{\mathbf{v}_2} 1\,d\mathbf{v}\right)$$

Integrating the right hand side :

$$\int_{t_1}^{t_2} \mathbf{F}\,dt = m\mathbf{v}_2 - m\mathbf{v}_1 = m\Delta\mathbf{v}$$

Refname	**DD:potEnergy**
Label	Potential energy
Symbol	PE
Units	J
Equation	$$PE = mgh$$
Description	PE is the potential energy (J) m is the mass (kg) g is the gravitational acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) h is the height (m)
Notes	The potential energy of an object is the energy held by an object because of its position to other objects. All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension). No damping occurs during the simulation (from A: dampingInvolvement).
Source	–
RefBy	

Refname	**DD:momentOfInertia**
Label	Moment of inertia
Symbol	$\mathbf{I}$
Units	kgm^2
Equation	$$\mathbf{I} = \sum m_j {r_j}^2$$
Description	$\mathbf{I}$ is the moment of inertia (kgm^2) m_j is the mass of the j-th particle (kg) r_j is the distance between the j-th particle and the axis of rotation (m)
Notes	The moment of inertia $\mathbf{I}$ of a body measures how much torque is needed for the body to achieve angular acceleration about the axis of rotation. All bodies are assumed to be rigid (from A: objectTy).
Source	–
RefBy	

4.2.5 Instance Models

This section transforms the problem defined in Section: Problem Description into one which is expressed in mathematical terms. It uses concrete symbols defined in Section: Data Definitions to replace the abstract symbols in the models identified in Section: Theoretical Models and Section: General Definitions.

The goal GS: Determine-Linear-Properties is met by IM: transMot and IM: col2D. The goal GS: Determine-Angular-Properties is met by IM: rotMot and IM: col2D.

Refname	**IM:transMot**
Label	Force on the translational motion of a set of 2d rigid bodies
Input	$\mathbf{v}_i$, t, g, $\mathbf{F}_i$, m_j
Output	$\mathbf{a}_i$
Input Constraints	$$\mathbf{v}_i > 0$$ $$t > 0$$ $$g > 0$$ $$\mathbf{F}_i > 0$$ $$m_j > 0$$
Output Constraints	
Equation	$$\mathbf{a}_i = \frac{d\mathbf{v}_i\left(t\right)}{dt} = g + \frac{\mathbf{F}_i\left(t\right)}{m_j}$$
Description	$\mathbf{a}_i$ is the the i-th body's acceleration ($\frac{\text{m}}{\text{s}^2}$) t is the time (s) $\mathbf{v}_i$ is the velocity of the i-th body's velocity ($\frac{\text{m}}{\text{s}}$) g is the gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$) $\mathbf{F}_i$ is the force applied to the i-th body at time t (N) m_j is the mass of the j-th particle (kg)
Notes	The above equation expresses the total acceleration of the rigid body i as the sum of gravitational acceleration (from GD: accelGravity) and acceleration due to applied force $\mathbf{F}_i\left(t\right)$ (from TM: NewtonSecLawMot). The resultant outputs are then obtained from this equation using DD: linDisp, DD: linVel, and DD: linAcc. The output of the instance model will be the functions of position and velocity over time that satisfy the ODE for the acceleration, with the given initial conditions for position and velocity. The motion is translational, so the position and velocity functions are for the centre of mass (from DD:

Refname	IM:rotMot
Label	Force on the rotational motion of a set of 2D rigid body
Input	ω, t, τ_i, $\mathbf{I}$
Output	α
Input Constraints	$\omega > 0$ $t > 0$ $\tau_i > 0$ $\mathbf{I} > 0$
Output Constraints	$\alpha > 0$
Equation	$$\alpha = \frac{d\omega\left(t\right)}{dt} = \frac{\tau_i\left(t\right)}{\mathbf{I}}$$
Description	α is the angular acceleration ($\frac{\text{rad}}{\text{s}^2}$) t is the time (s) ω is the angular velocity ($\frac{\text{rad}}{\text{s}}$) τ_i is the torque applied to the i-th body (Nm) $\mathbf{I}$ is the moment of inertia (kgm^2)
Notes	The above equation for the total angular acceleration of the rigid body i is derived from TM: NewtonSecLawRotMot, and the resultant outputs are then obtained from this equation using DD: angDisp, DD: angVel, and DD: angAccel. All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension). A right-handed coordinate system is used (from A: axesDefined).

Refname	**IM:col2D**
Label	Collisions on 2D rigid bodies
Input	t, j, m_{A}, $\mathbf{n}$
Output	t_{c}
Input Constraints	$t > 0$ $j > 0$ $m_{\text{A}} > 0$ $\mathbf{n} > 0$
Output Constraints	$\mathbf{v}_{\text{A}} > 0$ $t_{\text{c}} > 0$
Equation	$\mathbf{v}_{\text{A}}\left(t_{\text{c}}\right) = \mathbf{v}_{\text{A}}\left(t\right) + \frac{j}{m_{\text{A}}}\mathbf{n}$
Description	$\mathbf{v}_{\text{A}}$ is the velocity at point A ($\frac{\text{m}}{\text{s}}$) t_{c} is the denotes the time at collision (s) t is the time (s) j is the impulse (scalar) (Ns) m_{A} is the mass of rigid body A (kg) $\mathbf{n}$ is the collision normal vector (m)
Notes	The output of the instance model will be the functions of position, velocity, orientation, and angular acceleration over time that satisfy the equations for the velocity and angular acceleration, with the given initial conditions for position, velocity, orientation, and angular acceleration. The motion is translational, so the position, velocity, orientation, and angular acceleration functions are for the centre of mass (from DD: ctrOfMass). All bodies are assumed to be rigid (from A: objectTy) and two-dimensional (from A: objectDimension).

4.2.6 Data Constraints

Table:InDataConstraints shows the data constraints on the input variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. FIXME

Var	Physical Constraints	Typical Value	Uncert.
C_{R}	$0 \leq C_{\mathrm{R}} \leq 1$	0.8	10%
$\mathbf{F}$	–	98.1 N	10%
G	–	$66.743 \cdot 10^{-12}$ $\frac{\mathrm{m}^3}{(\mathrm{kgs}^2)}$	10%
$\mathbf{I}$	$\mathbf{I} > 0$	74.5 kgm^2	10%
L	$L > 0$	44.2 m	10%
m	$m > 0$	56.2 kg	10%
$\mathbf{p}$	–	0.412 m	10%
$\mathbf{v}$	–	2.51 $\frac{\mathrm{m}}{\mathrm{s}}$	10%
τ	–	200.0 Nm	10%
ω	–	2.1 $\frac{\mathrm{rad}}{\mathrm{s}}$	10%
ϕ	$0 \leq \phi \leq 2\pi$	$\frac{\pi}{2}$ rad	10%

Table 4: Input Data Constraints

4.2.7 Properties of a Correct Solution

Table:OutDataConstraints shows the data constraints on the output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable.

Var	Physical Constraints
$\mathbf{p}$	–
$\mathbf{v}$	–
ϕ	$0 \leq \phi \leq 2\pi$
ω	–

Table 5: Output Data Constraints

5 Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete, and the non-functional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete.

Simulation-Space: Create a space for all of the rigid bodies in the physical simulation to interact in.

-Initial-Conditions: Input the initial masses, velocities, orientations, angular velocities of, and forces applied on rigid bodies.

Surface-Properties: Input the surface properties of the bodies such as friction or elasticity.

ysical_Constraints: Verify that the inputs satisfy the required physical constraints from Section: Solution Characteristics Specification.

slation-Over-Time: Determine the positions and velocities over a period of time of the 2D rigid bodies acted upon by a force.

otation-Over-Time: Determine the orientations and angular velocities over a period of time of the 2D rigid bodies.

termine-Collisions: Determine if any of the rigid bodies in the space have collided.

sponse-Over-Time: Determine the positions and velocities over a period of time of the 2D rigid bodies that have undergone a collision.

5.2 Non-Functional Requirements

This section provides the non-functional requirements, the qualities that the software is expected to exhibit.

High-Performance: The code has a short reponse time when performing computation.

Correct: The outputs of the code have the properties described in Section: Properties of a Correct Solution.

Understandable: The code is modularized with complete module guide and module interface specification.

Portable: The code is able to be run in different environments.

Reliable: The code gives consistent outputs.

Reusable: The code is modularized.

Maintainable: The traceability between requirements, assumptions, theoretical models, general definitions, data definitions, instance models, likely changes, unlikely changes, and modules is completely recorded in traceability matrices in the SRS and module guide.

6 Likely Changes

This section lists the likely changes to be made to the software.

riable-ODE-Solver: The internal ODE-solving algorithm used by the library may be changed in the future.

xpanded-Collisions: A: collisionType - The library may be expanded to deal with edge-to-edge and vertex-to-vertex collisions.

nclude-Dampening: A: dampingInvolvement - The library may be expanded to include motion with damping.

Joints-Constraints: A: constraintsAndJointsInvolvement - The library may be expanded to include joints and constraints.

7 Unlikely Changes

This section lists the unlikely changes to be made to the software.

ulate-Rigid-Bodies: The goal of the system is to simulate the interactions of rigid bodies.

External-Input: There will always be a source of input data external to the software.

Coordinate-System: A Cartesian Coordinate system is used.

jects-Rigid-Bodies: All objects are rigid bodies.

8 Off-The-Shelf Solutions

As mentioned in Section: Problem Description, there already exist free open source game physics libraries. Similar 2D physics libraries are:

- Box2D: http://box2d.org/
- Nape Physics Engine: http://napephys.com/

Free open source 3D game physics libraries include:

- Bullet: http://bulletphysics.org/
- Open Dynamics Engine: http://www.ode.org/
- Newton Game Dynamics: http://newtondynamics.com/

9 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" should be modified as well. Table:TraceMatAvsAll shows the dependencies of data definitions, theoretical models, general definitions, instance models, requirements, likely changes, and unlikely changes on the assumptions. Table:TraceMatRefvsRef shows the dependencies of data definitions, theoretical models, general definitions, and instance models with each other. Table:TraceMatAllvsR shows the dependencies of requirements, goal statements on the data definitions, theoretical models, general definitions, and instance models.

	A: objectTy	**A: objectDimension**	**A: coordinat**
DD: ctrOfMass	X		
DD: linDisp	X		
DD: linVel	X		
DD: linAcc	X		
DD: angDisp	X	X	
DD: angVel	X	X	
DD: angAccel	X	X	
DD: chaslesThm	X		
DD: torque			
DD: kEnergy	X	X	
DD: coeffRestitution			
DD: reVeInColl	X		
DD: impulseV	X		
DD: potEnergy	X	X	
DD: momentOfInertia	X		
TM: NewtonSecLawMot			
TM: NewtonThirdLawMot			
TM: UniversalGravLaw			
TM: NewtonSecLawRotMot		X	
GD: accelGravity			
GD: impulse	X	X	
IM: transMot	X	X	
IM: rotMot	X	X	

	A: objectTy	A: objectDimension	A: coordinat
IM: col2D	X	X	
FR: Simulation-Space			
FR: Input-Initial-Conditions			
FR: Input-Surface-Properties			
FR: Verify-Physical_Constraints			
FR: Calculate-Translation-Over-Time			
FR: Calculate-Rotation-Over-Time			
FR: Determine-Collisions			
FR: Determine-Collision-Response-Over-Time			
NFR: High-Performance			
NFR: Correct			
NFR: Understandable			
NFR: Portable			
NFR: Reliable			
NFR: Reusable			
NFR: Maintainable			
LC: Variable-ODE-Solver			
LC: Expanded-Collisions			
LC: Include-Dampening			
LC: Include-Joints-Constraints			
UC: Simulate-Rigid-Bodies			
UC: External-Input			
UC: Cartesian-Coordinate-System			
UC: Objects-Rigid-Bodies			

Table 6: Traceability Matrix Sho
Items

	DD: ctrOfMass	DD: linDisp	DD: linVel	DD: linAcc	DD: an
DD: ctrOfMass					
DD: linDisp					
DD: linVel					
DD: linAcc					
DD: angDisp					
DD: angVel					
DD: angAccel					
DD: chaslesThm					
DD: torque					
DD: kEnergy					
DD: coeffRestitution					

	DD: ctrOfMass	DD: linDisp	DD: linVel	DD: linAcc	DD: an
DD: reVeInColl					
DD: impulseV					
DD: potEnergy					
DD: momentOfInertia					
TM: NewtonSecLawMot					
TM: NewtonThirdLawMot					
TM: UniversalGravLaw					
TM: NewtonSecLawRotMot					
GD: accelGravity					
GD: impulse					
IM: transMot	X	X	X	X	
IM: rotMot					X
IM: col2D	X				

	DD: ctrOfMass	DD: linDisp	DD: linVel	DD:
GS: Determine-Linear-Properties				
GS: Determine-Angular-Properties				
FR: Simulation-Space				
FR: Input-Initial-Conditions				
FR: Input-Surface-Properties				
FR: Verify-Physical_Constraints				
FR: Calculate-Translation-Over-Time				
FR: Calculate-Rotation-Over-Time				
FR: Determine-Collisions				
FR: Determine-Collision-Response-Over-Time				
NFR: High-Performance				
NFR: Correct				
NFR: Understandable				
NFR: Portable				
NFR: Reliable				
NFR: Reusable				
NFR: Maintainable				

10 Values of Auxiliary Constants

There are no auxiliary constants.

11 References

[1] J. Frederick Bueche. *Introduction to Physics for Scientists, Fourth Edition*. 1986.

[2] Wikipedia Contributors. *Cartesian coordinate system*. https://en.wikipedia.org/wiki/Cartesian_coordinate_system. June 2019.

[3] Wikipedia Contributors. *Chasles' theorem (kinematics)*. https://en.wikipedia.org/wiki/Chasles'_theorem_(kinematics). Nov. 2018.

[4] The Editors of Encyclopaedia Britannica. *Line*. https://www.britannica.com/science/line-mathematics. June 2019.

[5] Nirmitha Koothoor. "A document drive approach to certifying scientific computing software". MA thesis. Hamilton, ON, Canada: McMaster University, 2013.

[6] David L. Parnas. "Designing Software for Ease of Extension and Contraction". In: *ICSE '78: Proceedings of the 3rd international conference on Software engineering*. 1978, pp. 264–277.

[7] David L. Parnas. "On the Criteria To Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* (1972), pp. 1053–1058.

[8] David L. Parnas and P. C. Clements. "A rational design process: How and why to fake it". In: *IEEE Transactions on Software Engineering* 12.2 (Feb. 1986), pp. 251–257.

[9] David L. Parnas, P. C. Clements, and Wiess. "The Modular Structure of Complex Systems". In: *ICSE '84: Proceedings of the 7th international conference on Software engineering*. 1984, pp. 408–417.

[10] Rod Pierce. *Point*. https://www.mathsisfun.com/geometry/point.html. May 2017.

[11] W. Spencer Smith and Lei Lai. "A new requirements template for scientific computing". In: *Proceedings of the First International Workshop on Situational Requirements Engineering Processes - Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*. Ed. by PJ Agerfalk, N. Kraiem, and J. Ralyte. In conjunction with 13th IEEE International Requirements Engineering Conference, Paris, France, 2005, pp. 107–121.

[12] Greg Wilson et al. "Best Practices for Scientific Computing, 2013". In: *PLoS Biol* 12.1 (2013).