Software Requirements Specification for Chipmunk2D

Alex Halliwushka and Luthfi Mawarid

Reference Material

This section records information for easy reference.

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 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
$\boldsymbol{a}$	Acceleration	m/s^2
$\boldsymbol{a}(t)$	Linear Acceleration	m/s^2
$\boldsymbol{a}_i$	The I-Th Body's Acceleration	m/s^2
C_R	Coefficient of restitution	--
$\boldsymbol{F}$	Force	N
$\boldsymbol{F}_1$	Force exerted by the first body (on another body)	N
$\boldsymbol{F}_2$	Force exerted by the second body (on another body)	N
$\boldsymbol{F}_i$	Force Applied to the I-Th Body at Time T	N
G	Gravitational constant	$m^3/(kg \cdot s^2)$
g	Gravitational acceleration	m/s^2
h	Height	m
$\boldsymbol{I}$	Moment of inertia	$kg \cdot m^2$
$\boldsymbol{I}_A$	Moment of Inertia Of Rigid Body A	$kg \cdot m^2$
$\boldsymbol{I}_B$	Moment of Inertia Of Rigid Body B	$kg \cdot m^2$
$\boldsymbol{J}$	Impulse (vector)	$N \cdot s$
j	Impulse (scalar)	$N \cdot s$
KE	Kinetic energy	J
L	Length	m
M	Total Mass of the 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
$\mathbf{n}$	Collision Normal Vector	m
PE	Potential energy	J
$\mathbf{p}$	Position	m
$\mathbf{p}_{CM}$	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
$\mathbf{r}_{OB}$	Displacement vector between the origin and point B	m
$\hat{\mathbf{r}}$	Displacement unit vector	m
t	Time	s
t_c	Denotes the time at collision	s
$\mathbf{v}$	Velocity	m/s
$\Delta\mathbf{v}$	Change in velocity	m/s
$\mathbf{v}(t)$	Linear Velocity	m/s
$\mathbf{v}^{AP}$	Velocity Of the Point of Collision P in Body A	m/s
$\mathbf{v}^{BP}$	Velocity Of the Point of Collision P in Body B	m/s
$\mathbf{v}_1$	Velocity Of the First Body	m/s
$\mathbf{v}_2$	Velocity Of the Second Body	m/s
$\mathbf{v}_A$	Velocity At Point A	m/s
$\mathbf{v}_B$	Velocity At Point B	m/s
$\mathbf{v}_f^{AB}$	Final Relative Velocity Between Rigid Bodies of A and B	m/s
$\mathbf{v}_i$	Velocity Of the I-Th Body's Velocity	m/s
$\mathbf{v}_i^{AB}$	Initial Relative Velocity Between Rigid Bodies of A and B	m/s
$\mathbf{v}_O$	Velocity At Point Origin	m/s
$\lVert\mathbf{n}\rVert$	Length of the Normal Vector	m
$\lVert\mathbf{r}\rVert$	Euclidean norm of the displacement	m
$\lVert\mathbf{r}_{AP}{}^{*}\mathbf{n}\rVert$	Length of the Perpendicular Vector To the Contact Displacement Vector of Rigid Body A	m
$\lVert\mathbf{r}_{BP}{}^{*}\mathbf{n}\rVert$	Length of the Perpendicular Vector To the Contact Displacement Vector of Rigid Body B	m
$\lVert\mathbf{r}\rVert^2$	Squared distance	m^2
α	Angular Acceleration	rad/s^2
θ	Angular Displacement	rad
τ	Torque	N·m
τ_i	Torque applied to the i-th body	N·m
ω	Angular Velocity	rad/s
ϕ	Orientation	rad

Abbreviations and Acronyms

Abbreviation	Full Form
2D	Two-Dimensional
3D	Three-Dimensional

A	Assumption
CM	Centre of Mass
Chipmunk2D	Chipmunk2D game physics library
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
ODE	Ordinary Differential Equation
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
UC	Unlikely Change
Uncert.	Typical Uncertainty

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 Chipmunk2D. 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.

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 Chipmunk2D 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 parnasClements1986, the most logical way to present the documentation is still to "fake" a rational design process.

Scope of Requirements

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

Characteristics of Intended Reader

Reviewers of this documentation should have an understanding of rigid body dynamics and high school calculus. The users of Chipmunk2D can have a lower level of expertise, as explained in Section: User Characteristics.

Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by dParnas1972 and parnasClements1984. 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.

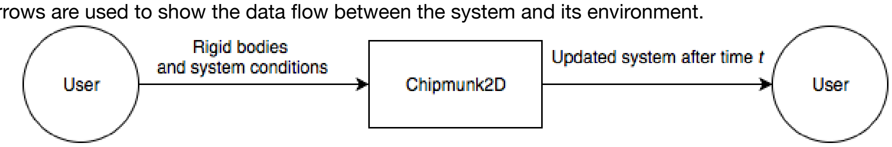
The goal statements (Section: Goal Statements) are refined to the theoretical models and the theoretical models (Section: Theoretical Models) to the instance models (Section: Instance Models).

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.

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 (Chipmunk2D). Arrows are used to show the data flow between the system and its environment.



System Context

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.
- Chipmunk2D 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.

User Characteristics

The end user of Chipmunk2D should have an understanding of first year programming concepts and an understanding of high school physics.

System Constraints

There are no system constraints.

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.

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).

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: 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 cartesianWiki).
- Right-handed coordinate system: A coordinate system where the positive z-axis comes out of the screen.

Goal Statements

Given the kinematic properties, and forces including any (collision forces) applied on a set of rigid bodies, the goal statements are:
Determine-Linear-Properties: Determine their new positions and velocities over a period of time.

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

Solution Characteristics Specification

The instance models that govern Chipmunk2D 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.

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: chalses, DD: reVelnColl, IM: transMot, IM: rotMot, DD: potEnergy, DD: ctrOfMass, DD: momentOfInertia, DD: linVel, DD: linDisp, DD: linAcc, DD: kEnergy, DD: impulseV, DD: impulse, IM: col2D, TM: ChaslesThm, DD: angVel, DD: angDisp, and DD: angAccel.)

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

coordinateSystemTy: The library uses a Cartesian coordinate system.

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

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

dampingInvolvement: There is no damping involved throughout the simulation and this implies that there are no friction forces. (RefBy: DD: chalses, IM: transMot, IM: rotMot, DD: potEnergy, DD: linVel, DD: linDisp, DD: linAcc, LC: Include-Dampening, DD: kEnergy, IM: col2D, DD: angVel, DD: angDisp, and DD: angAccel.)

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

Theoretical Models

This section focuses on the general equations and laws that Chipmunk2D is based on.

Refname **TM:NewtonSecLawMot**

Label Newton's second law of motion

Equation $\boldsymbol{F} = m\,\boldsymbol{a}$

Description

- $\boldsymbol{F}$ is the force (N)
- m is the mass (kg)
- $\boldsymbol{a}$ is the acceleration (m/s^2)

Notes The net force $\boldsymbol{F}$ (N) on a body is proportional to the acceleration $\boldsymbol{a}$ (m/s^2) of the body, where m (kg) denotes the mass of the body as the constant of proportionality.

Source --

RefBy

Refname **TM:NewtonThirdLawMot**

Label Newton's third law of motion

Equation $\boldsymbol{F_1} = -\boldsymbol{F_2}$

Description

- $\boldsymbol{F_1}$ is the force exerted by the first body (on another body) (N)
- $\boldsymbol{F_2}$ 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 $\boldsymbol{F_1}$ (N) exerted on the second rigid body by the first is equal in magnitude and in the opposite direction to the force $\boldsymbol{F_2}$ (N) exerted on the first rigid body by the second.

Source --

RefBy

Refname **TM:UniversalGravLaw**

Label Newton's law of universal gravitation

Equation

$\boldsymbol{F} = G$
$m_1 m_2 ||\boldsymbol{r}||^2$
$\hat{\boldsymbol{r}} = G$
$m_1 m_2 ||\boldsymbol{r}||^2$

$\boldsymbol{r} ||\boldsymbol{r}||$

Description

- $\boldsymbol{F}$ is the force (N)
- G is the gravitational constant (m^3/(kg·s^2))
- m_1 is the mass of the first body (kg)
- m_2 is the mass of the second body (kg)
- $||\boldsymbol{r}||$ is the Euclidean norm of the displacement (m)
- $\hat{\boldsymbol{r}}$ is the displacement unit vector (m)
- $\boldsymbol{r}$ is the displacement (m)

Notes Two rigid bodies in the universe attract each other with a force $\boldsymbol{F}$ (N) that is directly proportional to the product of their masses, m_1 and m_2 (kg), and inversely proportional to the squared distance $||\boldsymbol{r}||^2$ (m^2) between them. The vector $\boldsymbol{r}$ (m) is the displacement between the centres of the rigid bodies and $||\boldsymbol{r}||$ (m) represents the Euclidean norm of the displacement, or absolute distance between the two. $\hat{\boldsymbol{r}}$ denotes the displacement unit vector, equivalent to the displacement divided by the Euclidean norm of the displacement, as shown above. Finally, G is the gravitational constant (6.673 * 10E-11) (m^3/(kg·s^2)).

Source --

RefBy

Refname **TM:ChaslesThm**

Label Chasles' theorem

Equation $\boldsymbol{v_B} = \boldsymbol{v_O} + \omega \times \boldsymbol{r_{OB}}$

Description

- $\boldsymbol{v_B}$ is the velocity at point B (m/s)
- $\boldsymbol{v_O}$ is the velocity at point origin (m/s)
- ω is the angular velocity (rad/s)
- $\boldsymbol{r_{OB}}$ is the displacement vector between the origin and point B (m)

Notes The linear velocity $\boldsymbol{v_B}$ (m/s) of any point B in a rigid body A: objectTy is the sum of the linear velocity $\boldsymbol{v_O}$ (m/s) 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 (rad/s) and the displacement vector between the origin and point B, $\boldsymbol{r_{OB}}$ (m).

Source --

RefBy

Refname **TM:NewtonSecLawRotMot**

Label Newton's second law for rotational motion

Equation $\tau = \boldsymbol{I}\,\alpha$

Description

- τ is the torque (N·m)
- $\boldsymbol{I}$ is the moment of inertia (kg·m^2)
- α is the angular acceleration (rad/s^2)

Notes The net torque τ (N·m) on a rigid body is proportional to its angular acceleration α (rad/s^2). Here, $\boldsymbol{I}$ (kg·m^2) denotes the moment of inertia of the rigid body. We also assume that all rigid bodies involved are two-dimensional A: objectDimension.

Source --

RefBy

General Definitions

There are no general definitions.

Data Definitions

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

Refname	DD:ctrOfMass
Label	Center of Mass
Symbol	$\mathbf{p}_{CM}$
Units	m
Equation	$\mathbf{p}_{CM} = \sum m_j \mathbf{p}_j M$
Description	• $\mathbf{p}_{CM}$ 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 is the total mass of the rigid body (kg)
Notes	A: objectTy A: objectDimension
Source	--
RefBy	IM: transMot and IM: col2D

Refname	DD:linDisp
Label	Linear Displacement
Symbol	$\mathbf{r}(t)$
Units	m
Equation	$\mathbf{r}(t) = d\mathbf{p}(t)\,dt$
Description	• $\mathbf{r}(t)$ is the linear displacement (m) • t is the time (s) • $\mathbf{p}$ is the position (m)
Notes	A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	IM: transMot

Refname	DD:linVel
Label	Linear Velocity
Symbol	$\mathbf{v}(t)$
Units	m/s
Equation	$\mathbf{v}(t) = d\mathbf{r}(t)\,dt$
Description	• $\mathbf{v}(t)$ is the linear velocity (m/s) • t is the time (s) • $\mathbf{r}$ is the displacement (m)
Notes	A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	IM: transMot

Refname	DD:linAcc
Label	Linear Acceleration
Symbol	$\mathbf{a}(t)$
Units	m/s^2
Equation	$\mathbf{a}(t) = d\mathbf{v}(t)\,dt$
Description	• $\mathbf{a}(t)$ is the linear acceleration (m/s^2) • t is the time (s) • $\mathbf{v}$ is the velocity (m/s)
Notes	A: objectTy A: objectDimension A: dampingInvolvement
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	A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	IM: rotMot

Refname	**DD:angVel**
Label	Angular Velocity
Symbol	ω
Units	rad/s
Equation	$\omega = \frac{d\theta(t)}{dt}$
Description	• ω is the angular velocity (rad/s) • t is the time (s) • θ is the angular displacement (rad)
Notes	A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	IM: rotMot

Refname	**DD:angAccel**
Label	Angular Acceleration
Symbol	α
Units	rad/s^2
Equation	$\alpha = \frac{d\omega(t)}{dt}$
Description	• α is the angular acceleration (rad/s^2) • t is the time (s) • ω is the angular velocity (rad/s)
Notes	A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	IM: rotMot

Refname	**DD:impulse**
Label	Impulse (scalar)
Symbol	j
Units	N·s
Equation	$j = \frac{-(1+C_R)\,\mathbf{v}_i^{AB}\cdot\mathbf{n}}{\left(\frac{1}{m_A}+\frac{1}{m_B}\right)\|\mathbf{n}\|^2+\frac{\|\mathbf{r}_{AP}*\mathbf{n}\|^2}{\mathbf{I}_A}+\frac{\|\mathbf{r}_{BP}*\mathbf{n}\|^2}{\mathbf{I}_B}}$

Description	• j is the impulse (scalar) (N·s) • C_R is the coefficient of restitution (Unitless) • $\boldsymbol{v}_i^{AB}$ is the initial relative velocity between rigid bodies of A and B (m/s) • $\boldsymbol{n}$ 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) • $\lVert\boldsymbol{n}\rVert$ is the length of the normal vector (m) • $\lVert\boldsymbol{r}_{AP}{}^{*}\boldsymbol{n}\rVert$ is the length of the perpendicular vector to the contact displacement vector of rigid body A (m) • $\boldsymbol{I}_A$ is the moment of inertia of rigid body A (kg·m^2) • $\lVert\boldsymbol{r}_{BP}{}^{*}\boldsymbol{n}\rVert$ is the length of the perpendicular vector to the contact displacement vector of rigid body B (m) • $\boldsymbol{I}_B$ is the moment of inertia of rigid body B (kg·m^2)
Notes	A: objectTy A: objectDimension A: axesDefined A: collisionType
Source	--
RefBy	IM: col2D

Refname	**DD:chalses**
Label	Velocity At Point B
Symbol	$\boldsymbol{v}_B$
Units	m/s
Equation	$\boldsymbol{v}_B = \boldsymbol{v}_O + \omega \times \boldsymbol{r}_{OB}$
Description	• $\boldsymbol{v}_B$ is the velocity at point B (m/s) • $\boldsymbol{v}_O$ is the velocity at point origin (m/s) • ω is the angular velocity (rad/s) • $\boldsymbol{r}_{OB}$ is the displacement vector between the origin and point B (m)
Notes	The linear velocity $\boldsymbol{v}_B$ (m/s) of any point B in a rigid body A: objectTy is the sum of the linear velocity $\boldsymbol{v}_O$ (m/s) 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 (rad/s) and the displacement vector between the origin and point B, $\boldsymbol{r}_{OB}$ (m). A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	

Refname	**DD:torque**
Label	Torque
Symbol	τ
Units	N·m
Equation	$\tau = \boldsymbol{r} \times \boldsymbol{F}$
Description	• τ is the torque (N·m) • $\boldsymbol{r}$ is the displacement (m) • $\boldsymbol{F}$ 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	$KE = m\boldsymbol{v}^2 2$
Description	• KE is the kinetic energy (J) • m is the mass (kg) • $\boldsymbol{v}$ is the velocity (m/s)
Notes	The kinetic energy of an object is the energy it possess due to its motion. A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	

Refname	**DD:coeffRestitution**
Label	Coefficient of restitution
Symbol	C_R
Units	Unitless
Equation	$C_R = -(\boldsymbol{v}_f^{AB} \cdot \boldsymbol{n} \boldsymbol{v}_i^{AB} \cdot \boldsymbol{n})$
Description	• C_R is the coefficient of restitution (Unitless) • $\boldsymbol{v}_f^{AB}$ is the final relative velocity between rigid bodies of A and B (m/s) • $\boldsymbol{n}$ is the collision normal vector (m) • $\boldsymbol{v}_i^{AB}$ is the initial relative velocity between rigid bodies of A and B (m/s)
Notes	The coefficient of restitution C_R is a unitless, dimensionless quantity that determines the elasticity of a collision between two rigid bodies. $C_R = 1$ results in an elastic collision, while $C_R < 1$ results in an inelastic collision, and $C_R = 0$ results in a totally inelastic collision.
Source	--
RefBy	

Refname	**DD:reVelnColl**
Label	Initial Relative Velocity Between Rigid Bodies of A and B
Symbol	$\boldsymbol{v}_i^{AB}$
Units	m/s
Equation	$\boldsymbol{v}_i^{AB} = \boldsymbol{v}^{AP} - \boldsymbol{v}^{BP}$
Description	• $\boldsymbol{v}_i^{AB}$ is the initial relative velocity between rigid bodies of A and B (m/s) • $\boldsymbol{v}^{AP}$ is the velocity of the point of collision P in body A (m/s) • $\boldsymbol{v}^{BP}$ is the velocity of the point of collision P in body B (m/s)
Notes	In a collision, the velocity of a rigid body A: objectTy A colliding with another rigid body B relative to that body $\boldsymbol{v}_i^{AB}$ is the difference between the velocities of A and B at point P.
Source	--
RefBy	

Refname	**DD:impulseV**
Label	Impulse (vector)
Symbol	$\boldsymbol{J}$
Units	N·s
Equation	$\boldsymbol{J} = m \Delta \boldsymbol{v}$
Description	• $\boldsymbol{J}$ is the impulse (vector) (N·s) • m is the mass (kg) • $\Delta \boldsymbol{v}$ is the change in velocity (m/s)
Notes	An impulse (vector) $\boldsymbol{J}$ occurs when a force $\boldsymbol{F}$ acts over a body over an interval of time. A: objectTy
Source	--
RefBy	

Detailed derivation of impulse (vector):

Newton's second law of motion states:

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

Rearranging :

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

Integrating the right hand side :

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

Refname	**DD:potEnergy**
Label	Potential energy
Symbol	PE
Units	J
Equation	$PE = m g h$
Description	• PE is the potential energy (J) • m is the mass (kg) • g is the gravitational acceleration (m/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. A: objectTy A: objectDimension A: dampingInvolvement
Source	--
RefBy	

Refname	**DD:momentOfInertia**
Label	Moment of inertia
Symbol	$\boldsymbol{I}$
Units	$kg \cdot m^2$
Equation	$\boldsymbol{I} = \sum m_j r_j^2$
Description	• $\boldsymbol{I}$ is the moment of inertia ($kg \cdot m^2$) • 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 $\boldsymbol{I}$ of a body measures how much torque is needed for the body to achieve angular acceleration about the axis of rotation. A: objectTy
Source	--
RefBy	

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	$\boldsymbol{v}_i$, t, g, $\boldsymbol{F}_i$, m_j
Output	$\boldsymbol{a}_i$
Input Constraints	$\boldsymbol{v}_i>0$ $t>0$ $g>0$ $\boldsymbol{F}_i>0$ $m_j>0$
Output Constraints	
Equation	$\boldsymbol{a}_i = d\boldsymbol{v}_i(t)\,dt = g + \boldsymbol{F}_i(t) m_j$
Description	• $\boldsymbol{a}_i$ is the the i-th body's acceleration (m/s^2) • t is the time (s) • $\boldsymbol{v}_i$ is the velocity of the i-th body's velocity (m/s) • g is the gravitational acceleration (m/s^2) • $\boldsymbol{F}_i$ 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 A: objectTy A: objectDimension i as the sum of gravitational acceleration (GD3) and acceleration due to applied force Fi(t) (T1). The resultant outputs are then obtained from this equation using DD: linDisp DD: linVel DD: linAcc. It is currently assumed that there is no damping A: dampingInvolvement or constraints A: constraintsAndJointsInvolvement involved. DD: ctrOfMass.
Source	--
RefBy	

Refname	**IM:rotMot**
Label	Force on the rotational motion of a set of 2D rigid body
Input	ω, t, τ_i, $\boldsymbol{I}$
Output	α
Input Constraints	$\omega>0$ $t>0$ $\tau_i>0$ $\boldsymbol{I}>0$
Output Constraints	$\alpha>0$
Equation	$\alpha = d\omega(t)\,dt = \tau_i(t)\boldsymbol{I}$

Description	• α is the angular acceleration (rad/s^2) • t is the time (s) • ω is the angular velocity (rad/s) • τ_i is the torque applied to the i-th body (N·m) • $\boldsymbol{I}$ is the moment of inertia ($kg \cdot m^2$)
Notes	The above equation for the total angular acceleration of the rigid body A: objectTy A: objectDimension i is derived from T5, and the resultant outputs are then obtained from this equation using DD: angDisp DD: angVel DD: angAccel. It is currently assumed that there is no damping A: dampingInvolvement or constraints A: constraintsAndJointsInvolvement involved. A: axesDefined
Source	--
RefBy	

Refname	**IM:col2D**
Label	Collisions on 2D rigid bodies
Input	$t, j, m_A, \boldsymbol{n}$
Output	t_c
Input Constraints	$t>0$ $j>0$ $m_A>0$ $\boldsymbol{n}>0$
Output Constraints	$\boldsymbol{v}_A>0$ $t_c>0$
Equation	$\boldsymbol{v}_A(t_c) = \boldsymbol{v}_A(t)+$ jm_A $\boldsymbol{n}$
Description	• $\boldsymbol{v}_A$ is the velocity at point A (m/s) • t_c is the denotes the time at collision (s) • t is the time (s) • j is the impulse (scalar) (N·s) • m_A is the mass of rigid body A (kg) • $\boldsymbol{n}$ is the collision normal vector (m)
Notes	This instance model is based on our assumptions regarding rigid body A: objectTy A: objectDimension collisions A: collisionType Again, this does not take damping A: dampingInvolvement or constraints A: constraintsAndJointsInvolvement into account. A: axesDefined. DD: ctrOfMass DD: impulse
Source	--
RefBy	

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	$0 \leq C_R \leq 1$	0.8	10%
$\boldsymbol{F}$	--	98.1 N	10%
G	--	9.8 $m^3/(kg \cdot s^2)$	10%
$\boldsymbol{I}$	$\boldsymbol{I} \geq 0$	74.5 $kg \cdot m^2$	10%
L	$L \geq 0$	44.2 m	10%
m	$m \geq 0$	56.2 kg	10%
$\boldsymbol{p}$	--	0.412 m	10%
$\boldsymbol{v}$	--	2.51 m/s	10%
τ	--	200.0 N·m	10%
ω	--	2.1 rad/s	10%
ϕ	--	$\pi 2$ rad	10%

Input Data Constraints

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
$\boldsymbol{p}$

	v
	ϕ
	ω

Output Data Constraints

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.

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.

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

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

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

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

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

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

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

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.

Likely Changes

This section lists the likely changes to be made to the software.
Variable-ODE-Solver: The internal ODE-solving algorithm used by the library may be changed in the future.

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

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

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

Unlikely Changes

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

Cartesian-Coordinate-System: A Cartesian Coordinate system is used.

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

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/

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: coordinateSystemTy	A: axesDefined	A: collisionType	A: dampingInvolvement	A: constraintsAndJointsI
DD: ctrOfMass	X	X					
DD: linDisp	X	X				X	
DD: linVel	X	X				X	
DD: linAcc	X	X				X	
DD: angDisp	X	X				X	
DD: angVel	X	X				X	
DD: angAccel	X	X				X	
DD: impulse	X	X		X	X		
DD: chalses	X	X				X	
DD: torque							
DD: kEnergy	X	X				X	
DD: coeffRestitution							
DD: reVelnColl	X						
DD: impulseV	X						
DD: potEnergy	X	X				X	
DD: momentOfInertia	X						
TM: NewtonSecLawMot							
TM: NewtonThirdLawMot							
TM: UniversalGravLaw							
TM: ChaslesThm	X						
TM: NewtonSecLawRotMot		X					
IM: transMot	X	X				X	X
IM: rotMot	X	X		X		X	X
IM: col2D	X	X		X	X	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															X						
LC: Include-Dampening																		X			
LC: Include-Joints-Constraints																					X
UC: Simulate-Rigid-Bodies																					
UC: External-Input																					
UC: Cartesian-Coordinate-System																					
UC: Objects-Rigid-Bodies																					

Traceability Matrix Showing the Connections Between Assumptions and Other Items

	DD: ctrOfMass	**DD: linDisp**	**DD: linVel**	**DD: linAcc**	**DD: angDisp**	**DD: angVel**	**DD: angAccel**	**DD: impulse**	**DD: chalses**	**DD: torque**	**DD: kEnergy**	**DD: coeffRestitution**	**DD: reVeloc**	**DD: impulseV**	**DD: potEnergy**	**DD: momentOfInertia**	**TM: NewtonSecLawMot**	**TM: NewtonThirdLawMot**	**TM: UniversalGravLaw**	**TM: ChaslesThm**	**TM: NewtonSecLawRotMot**	**IM: transMot**	**IM: rotMot**	**IM: col2D**
DD: ctrOfMass																								
DD: linDisp																								
DD: linVel																								
DD: linAcc																								
DD: angDisp																								
DD: angVel																								
DD: angAccel																								
DD: impulse																								
DD: chalses																								
DD: torque																								

DD: kEnergy								
DD: coeffRestitution								
DD: reVelInColl								
DD: impulseV								
DD: potEnergy								
DD: momentOfInertia								
TM: NewtonSecLawMot								
TM: NewtonThirdLawMot								
TM: UniversalGravLaw								
TM: ChaslesThm								
TM: NewtonSecLawRotMot								
IM: transMot	X	X	X	X				
IM: rotMot					X	X	X	
IM: col2D	X							X

Traceability Matrix Showing the Connections Between Items and Other Sections

	DD: ctrOfMass	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	DD:	TM:	TM:	TM:	TM:	TM:	IM:	IM:	IM:	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:	NFR:	NFR:	NFR:	NFR:	NFR:	NFR:
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

Traceability Matrix Showing the Connections Between Requirements, Goal Statements and Other Items

Values of Auxiliary Constants

There are no auxiliary constants.

References

- [1]: Bueche, J. Frederick. *Introduction to Physics for Scientists, Fourth Edition*. Mcgraw-Hill College, 1986.
- [2]: Koothoor, Nirmitha. *A document drive approach to certifying scientific computing software*. McMaster University, Hamilton, ON, Canada: 2013. Print.
- [3]: Parnas, David L. "On the Criteria To Be Used in Decomposing Systems into Modules." *Communications of the ACM*, 1972. pp. 1053–1058. Print.
- [4]: Parnas, David L. "Designing Software for Ease of Extension and Contraction.." *ICSE '78: Proceedings of the 3rd international conference on Software engineering*. 1978. pp. 264–277.
- [5]: Parnas, David L. and Clements, P. C. "A rational design process: How and why to fake it." *IEEE Transactions on Software Engineering*, vol. 12, no. 2, Washington, USA: February, 1986. pp. 251–257. Print.
- [6]: Parnas, David L., Clements, P. C., and Wiess. "The Modular Structure of Complex Systems." *ICSE '84: Proceedings of the 7th international conference on Software engineering*. 1984. pp. 408–417.
- [7]: Smith, W. Spencer and Lai, Lei. "A new requirements template for scientific computing." *Proceedings of the First International Workshop on Situational Requirements Engineering Processes - Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*. Edited by PJ Agerfalk, N. Kraiem, and J. Ralyte, Paris, France: 2005. pp. 107–121. In conjunction with 13th IEEE International Requirements Engineering Conference,
- [8]: Wikipedia Contributors. *Cartesian coordinate system*. June, 2019. https://en.wikipedia.org/wiki/Cartesian_coordinate_system (https://en.wikipedia.org/wiki/Cartesian_coordinate_system).
- [9]: Wilson, Greg, Aruliah, D. A., Titus, C., Chue Hong, Neil P., Davis, Matt, Guy, Richard T., Haddock, Steven H. D., Huff, Kathryn D., Mitchell, Ian M., Plumblet, Mark D., Waugh, Ben, White, Ethan P., and Wilson, Paul. "Best Practices for Scientific Computing, 2013." *PLoS Biol*, vol. 12, no. 1, 2013. Print.


```
In [ ]:
```