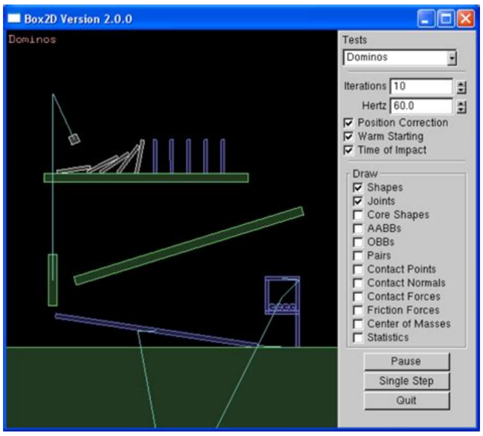
box2d manual



File Name: box2d manual.pdf
Size: 2796 KB
Type: PDF, ePub, eBook
Category: Book
Uploaded: 13 May 2019, 20:19 PM
Rating: 4.6/5 from 640 votes.

Book Descriptions:

We have made it easy for you to find a PDF Ebooks without any digging. And by having access to our ebooks online or by storing it on your computer, you have convenient answers with box2d manual . To get started finding box2d manual , you are right to find our website which has a comprehensive collection of manuals listed.
Our library is the biggest of these that have literally hundreds of thousands of different products represented.

Book Descriptions:

box2d manual

Programmers can use it in their games to make objects move in realistic ways and make the game world more interactive. From the game engines point of view, a physics engine is just a system for procedural animation. Most of the types defined in the engine begin with the b2 prefix. Hopefully this is sufficient to avoid name clashing with your game engine. If not, please first consult Google search and Wikipedia. You can get these tutorials from the download section of box2d.org. You should be comfortable with compiling, linking, and debugging. However, not every aspect is covered. Please look at the testbed included with Box2D to learn more. The latest version of Box2D may be out of sync with this manual. A testbed example that reproduces the problem is ideal. You can read about the testbed later in this document. We briefly define these objects here and more details are given later in this document. They are hard like a diamond. In the following discussion we use body interchangeably with rigid body. A fixture puts a shape into the collision system broadphase so that it can collide with other shapes. A 2D body has 3 degrees of freedom two translation coordinates and one rotation coordinate. If we take a body and pin it to the wall like a pendulum we have constrained the body to the wall. At this point the body can only rotate about the pin, so the constraint has removed 2 degrees of freedom. You do not create contact constraints; they are created automatically by Box2D. Box2D supports several joint types revolute, prismatic, distance, and more. Some joints may have limits and motors. For example, the human elbow only allows a certain range of angles. For example, you can use a motor to drive the rotation of an elbow. Box2D supports the creation of multiple worlds, but this is usually not necessary or desirable. The Box2D solver is a high performance iterative solver that operates in order N time, where N is the number of constraints.http://www.aeroklub-jihlava.cz/userfiles/corvette-1999-manual.xml

- **box2d manual, box2d manual, box2d manual pdf, box2d manual free, box2d manuals, box2d manual downloads, box2d manual online, box2d manual instructions, box2d manual 2017, box2d manual software, box2d manual.**

Without intervention this can lead to tunneling. First, the collision algorithms can interpolate the motion of two bodies to find the first time of impact TOI. Second, there is a substepping solver that moves bodies to their first time of impact and then resolves the collision. The Common module has code for allocation, math, and settings. Finally the Dynamics module provides the simulation world, bodies, fixtures, and joints. These tolerances have been tuned to work well with meterskilogramsecond MKS units. In particular, Box2D has been tuned to work well with moving shapes between 0.1 and 10 meters. So this means objects between soup cans and buses in size should work well. Static shapes may be up to 50 meters long without trouble. Unfortunately this will lead to a poor simulation and possibly weird behavior. An object of length 200 pixels would be seen by Box2D as the size of a 45 story building. Keep the size of moving objects roughly between 0.1 and 10 meters. Youll need to use some scaling system when you render your environment and actors. The Box2D testbed does this by using an OpenGL viewport transform.The billboard may move in a unit system of meters, but you can convert that to pixel coordinates with a simple scaling factor. You can then use those pixel coordinates to place your sprites, etc. You can also account for flipped coordinate axes. If your world units become larger than 2 kilometers or so, then the lost precision can affect stability. Use b2WorldShiftOrigin to support larger worlds. I recommend to use grid lines along with some hysteresis for triggering calls to ShiftOrigin. This call should be made infrequently because it is has CPU cost. You may need to store a physics offset when translating between game units and Box2D units. The body rotation is stored in radians and may grow unbounded. Consider normalizing the angle of your bodies if the magnitude of the angle becomes too large use b2BodySetAngle.http://skibetjagtforening.damgruppen.dk/userfiles/corvec-britony-ii-t-manual.xml

So when you create a b2Body or a b2Joint, you need to call the factory functions on b2World. You should never try to allocate these types in another manner. These definitions contain all the information needed to build the body or joint. By using this approach we can prevent construction errors, keep the number of function parameters small, provide sensible defaults, and reduce the number of accessors. So you can create definitions on the stack and keep them in temporary resources. These are created via b2WorldCreateBody. Programmers can use it in t heir games to make object s move in realistic ways and make the game wor ld more interac tive. From th e game engines point of view, a p hysics engine is just a system for procedural animation. Most of the types def ined in th e engine begin with the b2 prefix. Hopefully t his is sufficient t o avoid name c lashing with your game engin e. 1.2 Prere quisites In this manual Il l assume you are familiar with basic physics con cepts, suc h as mass, force, torque, and impulses. If not, please first consult Google searc h and Wikipedia. Box2D was created as part of a physics tut orial at the Game Developer Conference. You should be comfo rtable with compiling, li nkin g, and debugging. There are many resources for this o n the net. 1.3 About t his Manual This manual c overs the majority of t he Box2D API. However, not every asp ect is c overed. You are encouraged to look at the testbed inc luded with Box2D t o learn more. Also, the Box2 D code base has comments formatted for Doxygen, so it is easy to create a hyperlinked API doc ument. This manual is on ly updated with new releases. T he version in sou rce co n trol is likely t o be out of date. 1.4 Fe edback and Reportin g Bugs If you have a question o r feedback about Box2D, please leave a comment in the forum. This is also a great place for co mmunity discussion. A testbed example that reproduc es the pro blem is ideal.

You can read about t he test bed later in t his document. 1.5 Core Concepts Box2D works with several fundamental c oncept s and objects. We briefly define these o bjects here and more details are gi ven later in t his document.They are hard like a diamond. In the following discussion we use body interchangeably with rigid body.A fixture put s a shape into the collisi on syst em broadphase so that it c an collide with other shapes.A 2D body has 3 degrees of freedom two t ranslation co ordinates and one rotat ion coordinate. If we t ake a bod y and pin it to the wall li ke a pendulum we h ave const rained the b ody to the wall. At t his point t he body can only rotat e about th e pin, so the constraint has removed 2 degrees of freedom.Y ou do not c reate contact constraint s; they are creat ed automatically by Box2D.Box2D supports several joint types revolute, prismatic, distan ce, an d more. Some joints may have limits and mot ors.For example, the h uman elbow only allows a c ertain range of an gles. For example, you c an use a motor to drive the rotation of an elbow.Box2D supports the c reation of multiple worlds, but t his is usually not necessary or desirable.The Bo x2D solver is a high performance iterat ive solver t hat operates in order N time, where N is the number of const raints.Without intervention t his can lead to tunneling. Box2D contains specialized algor ithms to deal with tunneling. First, the c olli sion algorithm s can interpolate t he motion of two bodies to find t he first time of impact T OI. Second, t here is a sub stepping solver th at moves bo dies to t heir first time of impact and th en resolves the collision. 1.6 Mo dules Box2D is composed of three modules Common, Colli s ion, and Dynamics. The Common module has c ode for allocat ion, math, and settings. Fina lly th e Dynamics module pro vides t he simulation world, bodies, fixtures, and joints. These t olerances have been tuned to work well with meterskil o gramsecond MKS units.

https://formations.fondationmironroyer.com/en/node/8549

In particu lar, Box2D has been tuned to work well with moving shapes bet ween 0.1 and 10 meters. So t his means object s between so up cans and buses in size sho uld work well. Static shapes may b e up to 50 met ers long without trouble. Being a 2D physics engine, it is tempting to use pixels as your units. Unfortunat ely this will lead to a poor simulation and possibly weird behavior. An o bject o f length 20 0 pixe ls would be seen by Box2D as the size of a 45 story bu il ding. Caution Box2D is tuned fo r MKS units. Keep th e size of moving objects roughly between 0.1 and 1 0 meters. Yo ull need to use so me scaling system when you render your environment and actors. T he Box2D t estbed does this by using an O penGL viewport transform. DO NOT USE PIXELS. It is best to think of Box2D bodies as moving billboards upon which you att a ch your artwork. The billboard may mo ve in a unit system of meters, but you can convert that to pixel coo rdinates with a simple scaling fac tor. You can then use those pixel coordinates t o place yo ur sprites, etc. Y ou c an also acco unt for flipped coo rdinate axes. Box2D uses radians for angles. The body rotation is stored in radians an d may grow un bounded. Consider normalizin g the angle of your bo dies if the magnitude of the angle becomes to o large use b2BodySetAngle. So when you create a b2Body o r a b2Joint, you need to call the factory functions o n b2World. You should never t ry to allocate these t ypes in anot her manner. Th ese definitions cont ain all the information needed to b uild the bod y or joint. By using this ap proach we can prevent const ruction errors, keep the n umber of funct ion parameters small, provide sensible defau lts, and redu ce the number of accessors. So you can creat e definitions on the stack and keep them in t emporary resourc es. T his code does not contain any graphics. All you will se e is text output in t he console of the boxs po sition over t ime.

https://hardwareusato.com/images/casio-ctk-450-keyboard-manual.pdf

This is a good example of how to get up an d running with Box2D. 2.1 Creat ing a Worl d Every Box2 D program begins with the c reation of a b2World objec t. b2World is t he physics hub that manages memory, object s, and simulation. You can allocate th e physics wor ld on t h e stack, h eap, or data sec tion. It is easy t o creat e a Box2D world. First, we define t he gravi ty vec tor.Note that we are c re ating th e world on the stack, so the world must remain in scope.For st ep 1 we creat e the ground bo dy. For this we need a body definition. With the bo dy definition we specify the initial position of t he ground body.The world objec t does not keep a reference to the bod y definition. Bodies are static by default. Static bod ies dont collide with ot her static bodies an d are immovable. So in this case t he ground box is 100 units wide xaxis and 2 0 units tall yaxis. Box2D is tu ned for meters, kilograms, and seco nds. So you c an consider the extents to b e in meters. Box2 D generally works best when objects are the size of typical real world o bjects. Fo r example, a barr el is about 1 meter tall. Due to the limitations o f floating point arit hmetic, u sing Box2D to model the movement of glaciers or dust p ar ticles is not a good idea. We finish the ground bod y in step 4 by creating th e shape fixture. For t his step we h ave a shortcut. We do no t have a need t o alter the d efault fixture material pro perties, so we can pass the sha pe directly t o the bo dy without creat ing a fixture definition. L ater we will see how t o use a fixture definition for custo mized material properties. The seco nd parameter is the shape density in kilograms per meter squared. A stat ic body has zero mass by definition, so the density is no t used in this case. It c lone s th e data into a new b 2Shape object. Note t hat every fixture must have a parent body, even fixtures that are static. However, yo u can attach all static fixtures to a single static b ody.

https://juanguillermocadena.com/images/casio-ctk-491-keyboard-manual.pdf

When you attac h a shape to a body using a fixture, the shape’ s coordinates b ecome local t o the body. So when th e body moves, so does the shape. A fixture’s w orld tra nsform is inherited fro m the p arent bo dy. A fixture does not h ave a transform independent of the body. So we don’t move a shape arou n d on th e body. Moving or modifying a sha pe that is on a body is not supported. The reason is simpl e a body with morphing shapes is not a rigid body, but Box2D is a rigid body engine. Many of the assumptions made in Box2D are based on t he rigid body model. If this is viol ated many t hings will break 2.3 Creat ing a Dyn amic Body So no w we have a grou nd body. We can use the same tec hni que to create a dynamic body. The main difference, besides dimensions, is that we must est ablish the dynamic bodys mass p roperties. First we c reate t he body using CreateBody. By default bodies are st atic, so we sho uld set the b2BodyT ype at const ruction time to make the body dynamic. First we creat e a box shape b2PolygonShape dynamicBox; dynamicBox.SetAsBox1.0f, 1.0f;

Next we c reate a fixtu re definition using the box. Notice t hat we set density to 1. Th e d efault density is zero. Using th e fixture definition we can no w create the fixture. This aut omatically updates the mass o f the body. Yo u can add as many fixtures as you li ke to a body. Each one contribut es to the total mass. We are now ready to begin simulating. 2.4 Simulat ing the Wo rld of Box2D So we have initialized the ground box and a dynamic box. Now we are ready to set Newton loose to do his thing. We just have a cou ple more issues to consider. Box2D uses a computat ional algorithm called an int egrator. Integrators simulate t he physics equat ions at disc rete points of time. This goes along with t he traditional game loop where we essent ially have a flip book of movement on t he screen. So we n eed to pick a time step fo r Box2D.

You can get away with larger time steps, but you wil l have to be more c areful about setting up the def initions for your world. We also dont like the time st ep to chan ge much. A variable time step prod uces variable results, which makes it difficult to debug. So dont tie th e time step to your frame rate un less you really, really have to. Without further ad o, here is the time step. A single c onstra int can be solved perfect ly. How ever, when we solve one co nstraint, we slightly disrupt ot her co nstraints. To get a good solut ion, we n eed to iterate o ver all constraint s a number of t imes. There are two phases in the constraint so lver a velocity phase and a position p hase. In the velocity phase th e solver comput es the impulses necessary for the bod ies to move correc tly. In the posit ion phase th e solver adjusts the positions of the bodies to reduce overlap and joint detachment. Eac h phase has its o wn iteration c ount. In addition, the position phase may exit iterations early if the erro rs are small. The suggested iteration c ount f or Box2 D is 8 fo r velocity and 3 for position. Y ou can tune th is number to your liking, just keep in mind that this has a trade of f b etween performanc e and ac curacy. Using fewer iterations inc reases performance but ac curacy suffers. Likewi s e, using more it erations decreases performance but improves t he quality of your simulation. For t his simple example, we dont need much iteration. Here are our chosen iteration c ounts. An iteration is not a sub step. One solver iterat ion is a single pass over al l the c onstraints within a time step. You can have multiple passes over t he con straints within a single time step. We are no w ready to begin the simulation loop. In your game th e simulation loop c an be merged with your game loop. In eac h pass t hrough your game loop you call b2World Step. Just one call is usually enough, depending on you r frame rate and your physics t ime step.

The Hello World pro gram was des igned to be simple, so it h as no graphical outp ut. The code p rints out the po sition and rotat ion of the dynamic body. Here is the simulation loop that simulates 60 time steps for a total of 1 secon d of simulated time. Your out put should look like this T his is done to improve performance a nd make your life easier. However, you will need to nullify any body, fixture, o r joint pointers you have becau se they wil l become invalid. 2.6 T he Testbed Onc e you have conqu ered the Hell oWorld exa mple, yo u sho uld start looking at Box2Ds testb ed. The testbed is a unittest ing framework and demo environment.I encourage you t o explore and tinker with the t estbed as you learn Bo x2D. Note the testbed is writt en using freeglut and GLUI. The testbed is not part of th e Box2D library. The Box2D library is agnostic abou t rendering. As shown by the HelloWorld examp le, you dont need a renderer to use Box2 D. Constant s Allocation wrappers. The version number Typ es Box2D defines various types suc h as float 32, int8, etc.Constants Box2D defines several constan ts. These are all documented in b2Sett ings.h. Normally you do no t need to adjust these constant s. Box2D uses floating point math fo r co lli sion and simul a tion. Due t o round off erro r some n umerical tolerances are defined. Some toleranc es are absolute and some are relative. Absolute t olerances use MKS unit s. Allocation wr appers The set tings file defines b2Alloc and b2 Free for large allocat ions. You may fo rward these calls to your own memory manage ment system. Versi on The b2 Version structure holds the curren t version so you can query this at run time. 3.3 Me mory Man agement A large number of the dec isi ons ab out t he desig n of Box2D were based on the need for q uick and efficient use o f memory. In this sect ion I will discuss how and why Box2D allocates memory.

Using the syst em heap through mall oc or new fo r small objects is inefficient and c an cause fragmentat ion. Many of these small object s may have a short life span, such as c ontacts, but c an persist for several t ime steps. So we n eed an allocat or that c an efficiently provide heap memor y for t hese objects. Box2Ds solution is t o use a small object allocator S OA called b2 BlockAll ocat or. The SOA keeps a number of growable pools of varying sizes. When a requ est is made for memory, t he SOA ret urns a block of memory t hat best fits the request ed size. When a b l ock is freed, it is retu rned to the pool. Bot h of these o perations are fast and cause little heap traffic. Since Box2 D uses a SOA, you should never new or malloc a body, fixture, or joint. Howev er, you do have to all ocat e a b2 World on you r own. The b2World class provides factories for you to c reate bodies, fixtures, an d joints. T his allows Box2D t o us e th e SOA a nd hide t he gory det ails from you. Never, c all delete or free on a body, fixture, o r joint. While execut ing a time step, Box2D needs some temporary workspace memory. For this, it uses a st a ck allocator called b2Stac kAll ocat or to avoid p er step h ea p allocations. Y ou do nt need to interac t with the stack allocator, but its good to know its there. 3.4 Mat h Box2D includes a simple small vect or and matrix module. This has b een designed to suit the internal needs of Box2D an d th e API. All the memb ers are exposed, so you may u se them freely in your application. The mat h library is kept simple to make Box2D easy to port and maintain. T he module also c ontains a dynamic tree and broadph ase to acceleration co lli sion pro cessing of large systems. The c ollisi on modu le is designed to be usable o utside of t he dynamic syst em. For example, you can u se the dynamic tree for other aspect s of your game besides physics.

However, the main purpose of Box2D is to provide a rigid body physics engine, so th e using the c ollisi on module by itself may fe el limited fo r some applications. Likewise, I will not make a st rong effort to document it or polish th e APIs. 4.2 Shapes Shapes desc ribe collisi on geo metry an d may be used independently of physics simulation. At a minimum, you should un derstand ho w to create shapes that can be later a ttac hed to rigi d bod ies. Box2D shapes implement the b2Shape base class. In addition, each sha pe has a type member and a radius. The radius even applies to po lygons, as discussed below. Keep in mind t hat a shape does not know about bodies and st and apart fro m the dynamics system. Shapes are stored in a compact form t hat is optimized for size and performanc e. As such, shapes are not easily moved aro und. You have to manually set the sh ape vertex positions t o move a shape. However, when a sha pe is attac hed to a body using a fixture, the shapes move rigid ly with the ho st bod y. In summary When a shape is not attac hed to a b ody, you can view it's vertices as being expressed in world space. When a shape is att ached to a body, you can view it's vertices as being expressed in loc al coord inates. Circles are so lid. You cannot make a ho ll ow c ircle using the circle shape. Polygon s are solid and never hollow. A polygon must have 3 or more vertices. Polygons vertices are st ored with a counter cloc kwise windi ng CCW. We must be careful bec ause the notion of CCW is with respect to a righthanded c oordinate system with the zaxis pointing ou t of the plane. This might turn out to be cloc kwise on your screen, depending o n your coordinat e system convent ions. The po lygon members are public, but yo u should use i nitialization fu nctions t o create a polygon. The initialization fun ctions c reate no rmal v ectors and perform val idation. You can create a polygon shap e by passing in a vertex array.

The b2 PolygonShapeSet function auto matically computes the convex hull and establishes the p roper winding order. This func tion is fast when t he number of vertices is low. If you increase The skin is u sed in stacking sc enarios to keep polygons slightly separated. This allows co ntinuous c ollisi on t o work against the c ore polygon. The po lygon skin helps prevent t unneling by keep ing the polygons separat ed. This results in small gaps between t he shapes. Yo ur visual representat ion can be larger than the po lygon to hide any gaps. These are provided t o assist in making a free form stat ic environment for your game. A major limitation of edge shapes is t hat t hey can co ll ide with c ircles and polygons b ut not with themselves. The co lli sion algorit hms used by Bo x2D require t hat at least one of two colliding shapes have volume.This can giv e rise to an unexpec ted art ifact when a polygon slides along the c hain of edges. In t he figure below we see a box co ll iding with an internal vert ex. These g host collisi ons are caused when the polygon collides with an internal vertex generating an internal collision norma l. If edge1 did no t exist t his collision would seem fine. With edge1 present, the internal c olli sion seems l ike a bug. But normally when Box2D collides two shapes, it vi ews them in isolation. Fortun ately, the edge shape pro vides a mechanism for eliminating ghost c ollisio ns by st oring the adjacent ghost vertices. Box2D uses these ghost vertices to prevent internal co lli sions. Chain Shap es The c hain shape p rovides an efficient way to connect many edges to gether to const r uct your static game worlds. You can connec t chains to gether using ghost vertices, like we did with b2 EdgeShape. It mig ht work, it mig ht not. T he code that prevent s ghost c ollisi ons assumes t here are no selfi ntersect ions of th e chain. Each edge in the chain is treated as a child shape and c an be accessed by index.

Shape Po int Test You can test a point for overlap with a sha pe. You provide a t ransform for the shape and a world point. No hit wil l register if the ray starts inside the shape. A child index is inc luded for c hain shapes bec ause t he ray cast will only check a single edge at a time.If we consider circle circle or circlepolygon, we can on ly get one c ontact point an d normal. In the c ase of po lyg onpolyg on we can get two p oints. These p oints share the same n ormal vect or so Box2D groups t hem into a manifold stru cture. The c ontact solver takes ad vantage of t his to improve stacking st abili ty. Normally you d on't need to compute contact manifolds direct ly, however you wil l like ly use the result s produc ed in the simulation. The b2 Manifold stru cture holds a normal vecto r and up t o two contact points. The normal and points are held in loc al coordinat es. As a convenience for th e contact solver, each point st ores the normal and tangential friction impulses. The dat a stored in b2 Manifold is opt imi zed for internal use. If you need this dat a, it is usually best to use the b2 WorldManifold struc ture to generat e the world coordinates o f the contact no rmal and point s. You need to provide a b 2Manifold and the shape transforms a nd radii. During simulat ion shapes may move an d the manifolds may c hange. Points may be ad ded or removed. You can detect th is using b2GetPointSt ates. Th ere is also some c aching used t o warm start the d istance function for repeated calls. You c an see the details in b2 Distance.h. Time of Impact If two sha pes are moving fast, they may tunnel through eac h other in a singl e time step. The b2 TimeOfImpact function is used to determine the time when two moving shapes collide. This is called the t ime of impact TO I. Th e main purpose of b2TimeOfImpact is fo r tunn el prevention. In particular, it is designed to prevent mo ving objec ts from tunn eli ng out side of stat ic level geometry.

This func tion accounts fo r rotation and translation of b oth shapes, however if the ro tations are large enough, then the function may miss a collision. However t he function will still report a nono verlapped time and will capture all translational collisions. The t ime of impact function identities an initial separat ing axis and ensures the shapes do not cross on that axis. This might miss collisions th at are clear at the final posit ions. While t his approach may miss some co ll isions, it is very fast and adequat e for tunnel prevention. There may be cases where c ollisi ons are missed for small ro tations. Normally, these missed rot ational co lli sions should not harm game play. They tend to be glancing collisions. The fu nction requires two shapes co nverted to b2DistanceProxy and t wo b2Sweep structures. Th e sweep struc ture defines t he initial and final transforms of the shapes. You can use fixed rot ations to perform a sha pe cast. In this c ase, the t ime of impact funct ion wi ll not miss any collisions. 4.5 Dyn amic Tree The b2 DynamicTree c lass is used by Bo x2D to organize large numbers of shapes efficie ntly. T he class does not know about shapes. Instead it operat es on axi saligned bou nding boxes A ABBs wi th u ser data pointers. The dynamic tree is a h ierarchical AABB tree. Eac h internal no de in the tree has two children. A leaf node is a singl e user AA BB. The tree uses rotat ions to keep the tree balanced, even in the c ase of degenerate input. The t ree structure allows for efficient ray casts and region qu eries. For example, you may have hu ndreds of shapes in your sc ene. You could p erform a ray cast agai nst the sc ene in a b rute force manner by ray casting eac h shape. Th is would be inefficient b ecause it d oes not take advantage of shapes being spread This traverses t he ray thro ugh the tree skipping large numbers of shap es. A region query uses the tree to find all leaf AA BBs t hat overlap a query AABB.

T his is faster t han a b ru te force approach becau se many shapes c an be skipped. Normally you wil l not u se the dynamic tree directly. Rather you will go t hrough t he b2World c lass for ray casts and region queries. If you plan t o instant iate your own dynamic t ree, you can learn how to use it by looking at how Box2D uses it. 4.6 Broadphase Collision processing in a ph ysics step c an be div ided into n arrowphase and broadphase. In the narr ow phase we c ompute c ontact points be tween pairs of sh apes. Imagine we have N sha pes.T his great ly reduces t he number of narrowphase calls. Normally you d o not interact with the broadphase directly. Instead, Box2D creates and manages a broadphase internally. Also, b2Bro adPhase is designed with Box2D’s simulation loop in mind, so it is likely not suited for other use cases. Th e Dynamics module sits on to p of the Common and Colli sion modules, so you shou ld be somewhat familiar with those b y now. The D ynamics module con tains.In t he following, you may see some references t o classes t hat h ave not been described yet. T herefore, you may want to quickly skim this ch apter before reading it closely. The dynamics module is covered in t he following chapt ers. Y ou can apply forc e s, t orques, and impulses to bodies. Bodies c an be static, kinematic, or dynamic. Internall y, Box2 D stores z ero for the mass and the inverse mass. Stat ic bodies c an be moved manually by the user. A st atic body has zero velocity. St atic bodies do n ot collide with oth e r static or kinematic bodies. Kinemat ic bodies do not respond to forces. T hey can be moved manually by th e user, but normal ly a kinematic body is moved by sett ing its velocity. A kinematic body behaves as if it has infinite mass, however, Box2 D sto res zero for the mass and t he inverse mass. Kinematic bodies do not collide with ot her ki nematic or static bodies. They c an be moved m anually b y the user, but normally they move acco rding to fo r ces.

http://www.drupalitalia.org/node/79953