**Conclusion for b2controller**

吴凭飞 2018011366

Path: “liquid\liquidfun\Contributions\Enhancements\Controllers”

1. b2controller.h:
2. headerfile:

" liquid\liquidfun\Box2D\Box2D\Dynamics\b2World.h"

" liquid\liquidfun\Box2D\Box2D\Dynamics\b2Body.h"

2)Struct:

b2ControllerEdge:

A)Introduction:To connect bodies and controllers together (in a bipartite graph).(一个用来表示controller和所控制的粒子的对应关系的类)

B)Data member(data type):

a. body(b2body): The body.

b. controller(b2controller):The controller for body.

c. prevBody(b2ControllerEdge \*):The pointer for the previous ControllerEdge in the controller's joint list.

d. nextBody(b2ControllerEdge \*):The pointer for the next ControllerEdge in the controller's joint list.

e. prevController(b2ControllerEdge \*):The pointer for the previous ControllerEdge in the body's joint list.

f. nextController(b2ControllerEdge \*):The pointer for the next ControllerEdge in the body's joint list.

1. Class:
2. b2ControllerDef:
3. Introduction:

This class is used to build controllers（该类用于构造controller）

B)Friends:

friend class: b2world(#include" b2World.h")

C)Members:

Member function(parameter table):

1. public:

virtual **~b2ControllerDef**():

Destructor.

1. private:

pure vrtual b2Controller \* **Create** (b2BlockAllocator\*):

Allocate memory for controllers. （create函数的作用是为controller分配内存空间）

B. b2Controller:

A)Introduction:

Base class for different controllers. Controllers are a convience for encapsulating common per-step functionality.

（所有不同作用controller的基类，controller的作用是将一个step（见b2TimeStep.h）中所具体实现的效果/功能封装在一个类中）

B)Friends:

friend class: b2world(#include" b2World.h")

C)Members:

a. Data members(data type):

a)protected:  
 m\_world(b2World\*): controller控制的粒子所在的世界

m\_bodyList(b2ControllerEdge\*): controller控制的粒子列表

m\_bodyCount(int32(signed int,defined in“b2setting.h”))：controller所控制的粒子数量

b)private:

m\_prev(b2controller \*):指向上一个controller的指针

m\_next(b2controller \*):指向下一个controller的指针

b. Member function(parameter table):

a)public:

virtual **~b2Controller**():

Destructor.

pure virtual void **Step**(const b2TimeStep&):

Controllers override this to implement per-step functionality.

（b2Controller派生类（不同的controller）中将重载Step函数来具体实现在这每个step中对粒子的操作）

virtual void **Draw**(b2DebugDraw \*):

Controllers override this to provide debug drawing.（用于debug的函数？没有读懂它的作用）

void **AddBody**(b2Body\*):

Adds a body to the controller’s list.

（从头（m\_bodylist）增添一个粒子到controller所控制的粒子列表中）

void **RemoveBody**(b2Body\*):

Removes a body from the controller’s list.

（在controller所控制的粒子列表中去除一个指定的粒子）

void **Clear**():

Removes all bodies from the controller list.

（清除controller控制列表中所有粒子）

b2Controller\* **GetNext**() 以及

const b2Controller\* **GetNext**() const: （？这里的重载有什么用？）

Get the next controller in the world's body list.

（返回当前b2Controller的m\_next指针表示下一个controller）

b2World\* **GetWorld**() 以及

const b2World\* **GetWorld**() const:

Get the parent world of this body.

（返回当前b2Controller的m\_world即所在的世界）

b2ControllerEdge\* **GetBodyList**() 以及

const b2ControllerEdge\* **GetBodyList**() const:

Get the attached body list.

（返回当前b2Controller的m\_bodylist即所控制的粒子的列表）

b)protected:

**b2Controller**(const b2ControllerDef \*):

Constructor.

pure virtual void **Destroy**(b2BlockAllocator \*):

Clear the memory space allocated.

（释放所有分配给该当前b2Controller的内存空间）

1. private:

Static void **Destroy**(b2Controller\*, b2BlockAllocator\*):

Clear the memory space allocated for the designative controller.

(释放所有分配给指定b2Controller的内存空间)

1. b2BuoyancyController.h（浮力控制器）:
2. headerfile:

“liquid\liquidfun\Contributions\Enhancements\Controller.h”

1. Class:
2. b2BuoyancyControllerDef(derived from b2ControllerDef):
3. Introduction:

This class is used to build buoyancy controllers.

1. Members:
2. Data members(data type):

a)public:

normal(b2Vec2(2D column vector,defined in“b2Math.h”)):

The outer surface normal.

offset(float32(float,defined in“b2Settings.h”)):

The height of the fluid surface along the normal.

density(float32):

The fluid density.

Velocity(b2Vec2):

The velocity, for drag calculations.

LinearDrag(float32):

Linear drag co-efficient(线性阻力系数).

angularDrag(float32):

angular drag co-efficient

useDensity(bool):

If false, bodies are assumed to be uniformly dense, otherwise use the shapes densities.

useWorldGravity(bool):

If true, gravity is taken from the world instead of the gravity parameter.

gravity(b2Vec2):

Gravity vector, if the world's gravity is not used

1. Member functions(parameter table):

a)public:

**b2BuoyancyControllerDef**():

Constructor.

b)

b2BuoyancyController\* **Create**(b2BlockAllocator\*) const:

Allocate memory for the buoyancycontrollers.

1. b2BuoyancyController(derived from b2Controller)
2. Introduction:

This class is used to calculates buoyancy forces for fluids in the form of a half plane.

1. Friends:

Friend class: b2BuoyancyControllerDef

1. Members:
2. Data members(data type):

The same as the class “b2BuoyancyControllerDef”

1. Member functions(parameter table):
2. public:

void **Step**(const b2TimeStep&):

Override this function in the base class b2Controller to implement per-step functionality.

void **Draw**(b2DebugDraw \*):

Override this function in the base class to provide debug drawing.

1. protected:

void **Destroy**(b2BlockAllocator\*):

clear the memory allocated for the b2BuoyancyController.

1. private:

**b2BuoyancyController**(const b2BuoyancyControllerDef\*):

Constructor.

1. b2GravityController.h（重力控制器）:
2. headerfile:

“liquid\liquidfun\Contributions\Enhancements\Controller.h”

1. Class:
2. b2GravityControllerDef(derived from b2ControllerDef):
3. Introduction:

This class is used to build gravity controllers

1. Members:
2. Data members(data type):

public:

G(float32):

Specifies the strength of the gravitiation force

invSqr(bool):

If true, gravity is proportional to r^-2, otherwise r^-1

1. Member function(parameter table):

private:

b2GravityController\* **Create**(b2BlockAllocator\*) const:

Allocate memory for the gravitycontrollers.

1. b2GravityController(derived from b2controller):
2. Introduction:

This class is used to apply simplified gravity between every pair of bodies

1. Friends:

friend class: b2Controller

1. Members:
2. Data members(data type):

The same as the class“b2GravityControllerDef”

1. Member functions(parameter table):
2. public:

void **Step**(const b2TimeStep&):

Override this function in the base class b2Controller to implement per-step functionality.

1. protected:

void **Destroy**(b2BlockAllocator\* allocator):

clear the memory allocated for the b2GravityController.

1. private:

**b2GravityController**(const b2GravityControllerDef\*):

Constructor.

1. b2TensorDampingController.h(张量阻尼控制器)
2. headerfile:  
   “liquid\liquidfun\Contributions\Enhancements\Controller.h”
3. Class:
4. b2TensorDampingControllerDef(derived from b2ControllerDef):
5. Introduction:

This class is used to build tensor damping controllers.

1. Members:
2. Data members(data type):

public:

T(b2Mat22):  
 Tensor to use in damping model.

maxTimestep(float32):  
Set this to a positive number to clamp the maximum amount of damping done.

1. Member functions(parameter table):
2. public:

void **SetAxisAligned**(float32,float32):

Sets damping independantly along the x and y axes.

1. private:

b2TensorDampingController\* **Create**(b2BlockAllocator\* ) const:

Allocate memory for the tensordampingcontrollers.

1. b2TensorDampingController(derived from b2Controller):
2. Introduction:

The class is used to apply top down linear damping to the controlled bodies.The damping is calculated by multiplying velocity by a matrix in local co-ordinates.

1. Friends:  
   Friend class:

b2TensorDampingControllerDef

1. Members:
2. Data members(data type):

The same as the class“b2TensorDampingControllerDef”.

1. Member functions:
2. public:

void **Step**(const b2TimeStep&):

Override this function in the base class b2Controller to implement per-step functionality.

1. protected:

void **Destroy**(b2BlockAllocator\*):

clear the memory allocated for the b2TensorDampingController.

1. private:

**b2TensorDampingController**(const b2TensorDampingControllerDef \*):

Constructor.

1. b2ConstantAccelController.h（恒加速度控制器）:
2. headerfile:

“liquid\liquidfun\Contributions\Enhancements\Controller.h”

1. Class:
2. b2ConstantAccelControllerDef(derived from b2ControllerDef):
3. Introduction:

This class is used to build constant acceleration controllers.

1. Members:
2. Data members(data type):

public:

A(b2Vec2):

The acceleration to apply.

1. Member functions(parameter table):

private:

b2ConstantAccelController\* **Create**(b2BlockAllocator\*) const:

Allocate memory for the constantaccelcontrollers.

1. b2ConstantAccelController(derived from b2Controller)
2. Introduction:

This class is used to apply a acceleration every frame.

1. Friends:

Friend class: b2ConstantAccelControllerDef

1. Members:
2. Data members(data type):

The same as the class b2ConstantAccelControllerDef.

1. Member functions(parameter table):
2. public:

void **Step**(const b2TimeStep&):

Override this function in the base class b2Controller to implement per-step functionality.

1. protected:

void **Destroy**(b2BlockAllocator\* allocator):

clear the memory allocated for the b2ConstantAccelController.

1. Private:

**b2ConstantAccelController**(const b2ConstantAccelControllerDef\*):

Constructor.

1. b2ConstantForceController.h（恒力控制器）:
2. headerfile:

“liquid\liquidfun\Contributions\Enhancements\Controller.h”

1. Class:
2. b2ConstantForceControllerDef(derived from b2ControllerDef):
3. Introduction:

This class is used to build constant force controllers.

1. Members:
2. Data members(data type):

public:

F(b2Vec2):

The force to apply.

1. Member functions(parameter table):

Private:

b2ConstantForceController\* **Create**(b2BlockAllocator\*) const:

Allocate memory for the constantforcecontrollers.

1. b2ConstantForceController(derived from b2Controller):
2. Introduction:

The class is used to apply a force every frame.

1. Friends:

Friend class: b2ConstantForceControllerDef

1. Members:
2. Data members:

The same as the class b2ConstantForceControllerDef.

1. Member functions:
2. public:

void **Step**(const b2TimeStep&):

Override this function in the base class b2Controller to implement per-step functionality.

1. protected:

void Destroy(b2BlockAllocator\*):

clear the memory allocated for the b2ConstantForceController.

1. private:

b2ConstantForceController(const b2ConstantForceControllerDef\*):

Constructor.