final459

Generated by Doxygen 1.8.16

Chapter 1

readme

Main function is **BVTreeCollisionDetection.c** (p. 21) This function uses bouncing volumn tree to detection collision between triangle mesh and sphere

Yisen Wang 12/2019

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AABB_Def Struct Reference

`#include <bouncingBoxTree.h>`

Public Attributes

- float **min** [3]
- float **max** [3]

4.1.1 Detailed Description

Definition of axis aligned bouncing box
min point & max point

4.1.2 Member Data Documentation

4.1.2.1 max

`float AABB_Def::max[3]`

4.1.2.2 min

`float AABB_Def::min[3]`

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **bouncingBoxTree.h**

4.2 BVTreeNode Struct Reference

`#include <bouncingBoxTree.h>`

Public Attributes

- int **type**
- **AABB BV**
  node type
- int **numObject**
  bouncing box
- int * **object**
  number in node
- struct **BVTreeNode** * **left**
  index array to original data
- struct **BVTreeNode** * **right**

4.2.1 Detailed Description

Struct of node for BV tree
Node information:
type: NODE/ LEAF
BV: bouncin box for objects of the node
numObject: number of objects in this node
object: pointer to an array that stored the index of objects belongs to this node

4.2.2 Member Data Documentation

4.2.2.1 BV

AABB BVTreeNode::BV

node type

4.2.2.2 left

struct **BVTreeNode*** BVTreeNode::left

index array to original data

4.2.2.3 numObject

```
int BVTreeNode::numObject
```

bouncing box

4.2.2.4 object

```
int* BVTreeNode::object
```

number in node

4.2.2.5 right

```
struct  BVTreeNode* BVTreeNode::right
```

4.2.2.6 type

```
int BVTreeNode::type
```

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **bouncingBoxTree.h**

4.3 CollisionResult_def Struct Reference

```
#include <BVTreeTraversal.h>
```

Public Attributes

- int **tri**
- int **sph**
- struct **CollisionResult_def** * **next**

4.3.1 Detailed Description

Single linked list for storing collision result
tri,sph for storing the index of triangle and sphere

4.3.2 Member Data Documentation

4.3.2.1 next

```
struct  CollisionResult_def* CollisionResult_def::next
```

4.3.2.2 sph

```
int CollisionResult_def::sph
```

4.3.2.3 tri

```
int CollisionResult_def::tri
```

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **BVTreeTraversal.h**

4.4 Sphere_Def Struct Reference

```
#include <bouncingBoxTree.h>
```

Public Attributes

- Point **center**
- float **r**

4.4.1 Detailed Description

Definition of sphere data
data order is x y z r

4.4.2 Member Data Documentation

4.4.2.1 center

```
Point Sphere_Def::center
```

4.4.2.2 r

```
float Sphere_Def::r
```

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **bouncingBoxTree.h**

4.5 Stack_def Struct Reference

```
#include <BVTreeTraversal.h>
```

Public Attributes

- **Node** * **node1**
- **Node** * **node2**
- struct **Stack_def** * **next**
- struct **Stack_def** * **prev**

4.5.1 Detailed Description

Definition of Stack
node1 and node2 for storing two BV tree node

4.5.2 Member Data Documentation

4.5.2.1 next

```
struct  Stack_def* Stack_def::next
```

4.5.2.2 node1

```
 Node* Stack_def::node1
```

4.5.2.3 node2

```
Node* Stack_def::node2
```

4.5.2.4 prev

```
struct  Stack_def* Stack_def::prev
```

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **BVTreeTraversal.h**

4.6 Triangle_Def Struct Reference

```
#include <bouncingBoxTree.h>
```

Public Attributes

- float **p** [9]

4.6.1 Detailed Description

Definition of triangle data
data order is x1 y1 z1 x2 y2 z2 x3 y3 z3

4.6.2 Member Data Documentation

4.6.2.1 p

```
float Triangle_Def::p[9]
```

The documentation for this struct was generated from the following file:

- D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/ **bouncingBoxTree.h**

Chapter 5

File Documentation

5.1 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/bouncingBoxTree.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <assert.h>
#include <math.h>
#include "bouncingBoxTree.h"
```

Functions

- void **printAABB** (**AABB** a)

  Yisen Wang.
- void **setAABB** (**AABB** ∗a)
- void **printArr** (int ∗a, int n)
- double **maxIndex** (double x, double y, double z)

  BV Tree implementation: find axis that has largest length.
- double **max** (double x, double y, double z)
- double **min** (float x, float y, float z)
- int **TestAABB** (**AABB** a, **AABB** b)

  Bouncing Box implementation: chech whether two bouncing box intersect by checking the projections onto 3 axis.
- int **findNodeToPut** (float mean, float minCoordinate, float maxCoordinate, int num1, int num2)

  BV Tree implementation: find child with least number of object to put new object
  .
- void **updateBouncingBoxTri** (double ∗TriangleData, int index, **AABB** ∗pBV)

  BV Tree implementation(Triangle): update bouncing box with a new triangle.
- int **partitionObjectsTri** (double ∗TriangleData, int ∗objects, int ∗tobjects, int numObjects, **AABB** BV, **AABB** ∗pBVleft, **AABB** ∗pBVright)

  BV Tree implementation(Triangle): divide objects into 2 node by its position on the longest axis
  update the boucing box for two nodes at the same time

- void **buildBVTreeTri** (double ∗TriangleData, **Node** ∗∗tree, int ∗objects, int ∗tobjects, int numObjects, **AABB** BV, int ∗debug)

  BV Tree implementation(Triangle): main function for build BV tree

  .
- void **updateBouncingBoxSph** (double ∗SphereData, int index, **AABB** ∗pBV)

  BV Tree implementation(Sphere): update bouncing box with a new sphere

- int **partitionObjectsSph** (double ∗SphereData, int ∗objects, int ∗tobjects, int numObjects, **AABB** BV, **AABB** ∗pBVleft, **AABB** ∗pBVright)

  BV Tree implementation(Sphere): divide objects into 2 node by its position on the longest axis
  update the boucing box for two nodes at the same time

- void **buildBVTreeSph** (double ∗SphereData, **Node** ∗∗tree, int ∗objects, int ∗tobjects, int numObjects, **AABB** BV, int ∗debug)

  BV Tree implementation(Sphere): main function for build BV tree.
- void **freeTree** (**Node** ∗a)
- void **printTree** (**Node** ∗a)

5.1.1 Function Documentation

5.1.1.1 buildBVTreeSph()

```
void buildBVTreeSph (
            double * SphereData,
             Node ** tree,
            int * objects,
            int * tobjects,
            int numObjects,
             AABB BV,
            int * debug )
```

BV Tree implementation(Sphere): main function for build BV tree.

Parameters

[SphereData]	pointer to data of triangle mesh
[tree]	pointer to Node
[objects]	list of objects contained in the node
[tobjects]	extra list for dividing the list
[BV]	bouncing box that contain all objects in this node

5.1.1.2 buildBVTreeTri()

```
void buildBVTreeTri (
             double * TriangleData,
              Node ** tree,
             int * objects,
             int * tobjects,
             int numObjects,
              AABB BV,
             int * debug )
```

BV Tree implementation(Triangle): main function for build BV tree

.

Parameters

[TriangleData]	pointer to data of triangle mesh
[tree]	pointer to Node
[objects]	list of objects contained in the node
[tobjects]	extra list for dividing the node
[BV]	bouncing box that contain all objects in this node

5.1.1.3 findNodeToPut()

```
int findNodeToPut (
             float mean,
             float minCoordinate,
             float maxCoordinate,
             int num1,
             int num2 )
```

BV Tree implementation: find child with least number of object to put new object

.

Return values

0-left	node, 1-right node

5.1.1.4 freeTree()

```
void freeTree (
              Node * a )
```

BV Tree implementation: Free tree recursively

5.1.1.5 max()

```
double max (
            double x,
            double y,
            double z )
```

5.1.1.6 maxIndex()

```
double maxIndex (
            double x,
            double y,
            double z )
```

BV Tree implementation: find axis that has largest length.

Parameters

in	*number*	is the data you want to print.

Return values

index	of the axis: 0 - x axis 1 - y axis 2 - z axis

5.1.1.7 min()

```
double min (
            float x,
            float y,
            float z )
```

5.1.1.8 partitionObjectsSph()

```
int partitionObjectsSph (
            double * SphereData,
            int * objects,
            int * tobjects,
            int numObjects,
             AABB BV,
             AABB * pBVleft,
             AABB * pBVright )
```

BV Tree implementation(Sphere): divide objects into 2 node by its position on the longest axis
update the boucing box for two nodes at the same time

5.1.1.9 partitionObjectsTri()

```
int partitionObjectsTri (
             double * TriangleData,
             int * objects,
             int * tobjects,
             int numObjects,
              AABB BV,
              AABB * pBVleft,
              AABB * pBVright )
```

BV Tree implementation(Triangle): divide objects into 2 node by its position on the longest axis
update the boucing box for two nodes at the same time

5.1.1.10 printAABB()

```
void printAABB (
              AABB a )
```

Yisen Wang.

Debug implementation: print bouncing box

5.1.1.11 printArr()

```
void printArr (
             int * a,
             int n )
```

Debug implementation: print objects list that contain all objects in this node

5.1.1.12 printTree()

```
void printTree (
              Node * a )
```

5.1.1.13 setAABB()

```
void setAABB (
              AABB * a )
```

Bouncing Box implementation: set BV to zero

5.1.1.14 TestAABB()

```
int TestAABB (
              AABB a,
              AABB b )
```

Bouncing Box implementation: chech whether two bouncing box intersect by checking the projections onto 3 axis.

Return values

if	two AABB are intersected: 0-interescted

5.1.1.15 updateBouncingBoxSph()

```
void updateBouncingBoxSph (
             double * SphereData,
             int index,
              AABB * pBV )
```

BV Tree implementation(Sphere): update bouncing box with a new sphere

5.1.1.16 updateBouncingBoxTri()

```
void updateBouncingBoxTri (
             double * TriangleData,
             int index,
              AABB * pBV )
```

BV Tree implementation(Triangle): update bouncing box with a new triangle.

5.2 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_↩ tree/bouncingBoxTree.h File Reference

```
#include <stdlib.h>
#include <stddef.h>
```

Classes

- struct **AABB_Def**
- struct **Triangle_Def**
- struct **Sphere_Def**
- struct **BVTreeNode**

Macros

- #define **LEAF** 0
- #define **NODE** 11
- #define **LEFT** 0
- #define **RIGHT** 1
- #define **MIN_OBJECTS_PER_LEAF** 1
- #define **MIN_SIZE** 0.00001

Typedefs

- typedef struct **AABB_Def** **AABB**
- typedef struct **Triangle_Def** **Triangle**
- typedef struct **Sphere_Def** **Sphere**
- typedef struct **BVTreeNode** **Node**

5.2.1 Macro Definition Documentation

5.2.1.1 LEAF

```
#define LEAF 0
```

5.2.1.2 LEFT

```
#define LEFT 0
```

5.2.1.3 MIN_OBJECTS_PER_LEAF

```
#define MIN_OBJECTS_PER_LEAF 1
```

5.2.1.4 MIN_SIZE

```
#define MIN_SIZE 0.00001
```

5.2.1.5 NODE

```
#define NODE 11
```

5.2.1.6 RIGHT

```
#define RIGHT 1
```

5.2.2 Typedef Documentation

5.2.2.1 AABB

```
typedef struct  AABB_Def  AABB
```

Definition of axis aligned bouncing box
min point & max point

5.2.2.2 Node

```
typedef struct  BVTreeNode  Node
```

Struct of node for BV tree
Node information:
type: NODE/ LEAF
BV: bouncin box for objects of the node
numObject: number of objects in this node
object: pointer to an array that stored the index of objects belongs to this node

5.2.2.3 Sphere

```
typedef struct  Sphere_Def  Sphere
```

Definition of sphere data
data order is x y z r

5.2.2.4 Triangle

```
typedef struct  Triangle_Def  Triangle
```

Definition of triangle data
data order is x1 y1 z1 x2 y2 z2 x3 y3 z3

5.3 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/BVTreeCollisionDetection.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "inputData.c"
#include "bouncingBoxTree.h"
#include "BVTreeTraversal.h"
#include "bouncingBoxTree.c"
#include "BVTreeTraversal.c"
#include "getCond_new.c"
```

Functions

- int **main** (int argc, char *argv[])
  Yisen Wang.

5.3.1 Function Documentation

5.3.1.1 main()

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

Yisen Wang.

Update bouncing box for root

generate BV tree for triangle mesh

generate BV tree for sphere

treversal two BV tree to get collision results

5.4 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/BVTreeTraversal.c File Reference

```
#include <stddef.h>
#include "bouncingBoxTree.h"
#include "BVTreeTraversal.h"
```

Functions

- void **Push** (**Stack** ∗∗s, **Node** ∗a, **Node** ∗b)

  Yisen Wang.
- void **Pop** (**Stack** ∗∗s, **Node** ∗∗a, **Node** ∗∗b)

  Stack implementation: Pop stack element into a and b

  .
- int **isEmpty** (**Stack** ∗s)

  Stack implementation: Check is Stack empty

  .
- int **isLeaf** (**Node** ∗a)
- int **compareAABB** (**AABB** ∗box1, **AABB** ∗box2)
- int **descendA** (**Node** ∗a, **Node** ∗b)
- void **BVTreeTraversal** (**Node** ∗Tri, **Node** ∗Sph, double ∗TriangleData, double ∗SphereData, **CollisionResult** ∗r, int ∗lenResults)
- void **DirectTraversal** (double ∗TriangleData, int tri_num, double ∗SphereData, int sph_num, int ∗lenResults)

  Traversal implementation: Brute force traversal.
- void **getBox** (**Node** ∗treeTri, int deep, double ∗boxes, int ∗index)

  Debug purpose: Print bouncing Box for first k level bouncing tree.
- void **printList** (FILE ∗f3, **CollisionResult** ∗r, int len, double time)

  Print results of collision detection into file f3.

5.4.1 Function Documentation

5.4.1.1 BVTreeTraversal()

```
void BVTreeTraversal (
              Node * Tri,
              Node * Sph,
             double * TriangleData,
             double * SphereData,
              CollisionResult * r,
             int * lenResults )
```

Traversal implementation: Traversal two bouncing volumn trees

Parameters

[Tri]	bouncing volumn tree for triangle mesh
[Sph]	bouncing volumn tree for Sphere
[STriangleData,SphereData]	pointer to original data
[r]	List for storing result

Note

- Be sure you have called Dev_Init function before call this fuction.
- Remember to check return value.

5.4.1.2 compareAABB()

```
int compareAABB (
              AABB * box1,
              AABB * box2 )
```

AABB implementation: Compare the volumn of two bouncing box
return 1 if box1>box2

5.4.1.3 descendA()

```
int descendA (
              Node * a,
              Node * b )
```

Traversal implementation: Decide which child to descend into
return 1 if descend into Node a first

5.4.1.4 DirectTraversal()

```
void DirectTraversal (
             double * TriangleData,
             int tri_num,
             double * SphereData,
             int sph_num,
             int * lenResults )
```

Traversal implementation: Brute force traversal.

5.4.1.5 getBox()

```
void getBox (
              Node * treeTri,
             int deep,
             double * boxes,
             int * index )
```

Debug purpose: Print bouncing Box for first k level bouncing tree.

5.4.1.6 isEmpty()

```
int isEmpty (
              Stack * s )
```

Stack implementation: Check is Stack empty
.

Parameters

in	*number*	is the data you want to print.

Return values

the	number of print information, in bytes. return zero indicate print error !.

Note

- Be sure you have called Dev_Init function before call this fuction.
- Remember to check return value.

5.4.1.7 isLeaf()

```
int isLeaf (
              Node * a )
```

Traversal implementation: Check type of Node a, return 1 if a is LEAF

5.4.1.8 Pop()

```
void Pop (
              Stack ** s,
              Node ** a,
              Node ** b )
```

Stack implementation: Pop stack element into a and b
.

5.4.1.9 printList()

```
void printList (
             FILE * f3,
              CollisionResult * r,
             int len,
             double time )
```

Print results of collision detection into file f3.

5.4.1.10 Push()

```
void Push (
              Stack ** s,
              Node * a,
              Node * b )
```

Yisen Wang.

Stack implementation: push Node a and Node b into Stack c/

Parameters

in	*number*	is the data you want to print.

Return values

the	number of print information, in bytes. return zero indicate print error !.

Note

- Be sure you have called Dev_Init function before call this fuction.
- Remember to check return value.

5.5 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/BVTreeTraversal.h File Reference

```
#include "bouncingBoxTree.h"
```

Classes

- struct **Stack_def**
- struct **CollisionResult_def**

Typedefs

- typedef struct **Stack_def Stack**
- typedef struct **CollisionResult_def CollisionResult**

5.5.1 Typedef Documentation

5.5.1.1 CollisionResult

`typedef struct` **`CollisionResult_def CollisionResult`**

Single linked list for storing collision result
tri,sph for storing the index of triangle and sphere

5.5.1.2 Stack

`typedef struct` **`Stack_def Stack`**

Definition of Stack
node1 and node2 for storing two BV tree node

5.6 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_↩ tree/getCond_new.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Functions

- double * **GetplaneT** (double triangle_data[9])
- double * **project** (double triangle_data[9], double sphere_data[4])
- double * **CrossMatrix** (double *vect)
- int **colli** (double triangle_data[9], double sphere_data[4])

5.6.1 Function Documentation

5.6.1.1 colli()

```
int colli (
             double triangle_data[9],
             double sphere_data[4] )
```

5.6.1.2 CrossMatrix()

```
double* CrossMatrix (
             double * vect )
```

5.6.1.3 GetplaneT()

```
double* GetplaneT (
             double triangle_data[9] )
```

5.6.1.4 project()

```
double* project (
             double triangle_data[9],
             double sphere_data[4] )
```

5.7 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/inputData.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>
```

Functions

- void **writeStrToFile** (FILE *f, char *str)
- double * **ReadCSV1** (FILE *f1, int *n_rows, int n_cols, int kk)
- double * **ReadCSV2** (FILE *f1, double *r, int *n_rows, int n_cols)
- void **WriteCSV** (FILE *f2, double *data, int rows, int cols)

5.7.1 Function Documentation

5.7.1.1 ReadCSV1()

```
double* ReadCSV1 (
            FILE * f1,
            int * n_rows,
            int n_cols,
            int kk )
```

5.7.1.2 ReadCSV2()

```
double* ReadCSV2 (
            FILE * f1,
            double * r,
            int * n_rows,
            int n_cols )
```

5.7.1.3 WriteCSV()

```
void WriteCSV (
            FILE * f2,
            double * data,
            int rows,
            int cols )
```

5.7.1.4 writeStrToFile()

```
void writeStrToFile (
            FILE * f,
            char * str )
```

5.8 D:/Courses/ME459/finalProject/src/Xinyi/Xinyi/final/method2_tree/readme.md File Reference

Index