(./)

# BVH, BIOVISION HIERARCHY

**Background of this article**:

We are exporting animation data into .bvh file in one of the application we are developing currently. There is no SDK for writing bvh files. I have written the code in C++ for exporting the head animation data into bvh file.

Lets start the article with brief introduction of bvh files and its format and the legends used in bvh files and how to write the code.

**BVH Introduction**:

BVH, Biovision Hierarchy, is an animation file format which is available in several 3D applications that do not support FBX. It stores the animation data in terms of ROOT, JOINTS and OFFSET.

**BVH Format**:
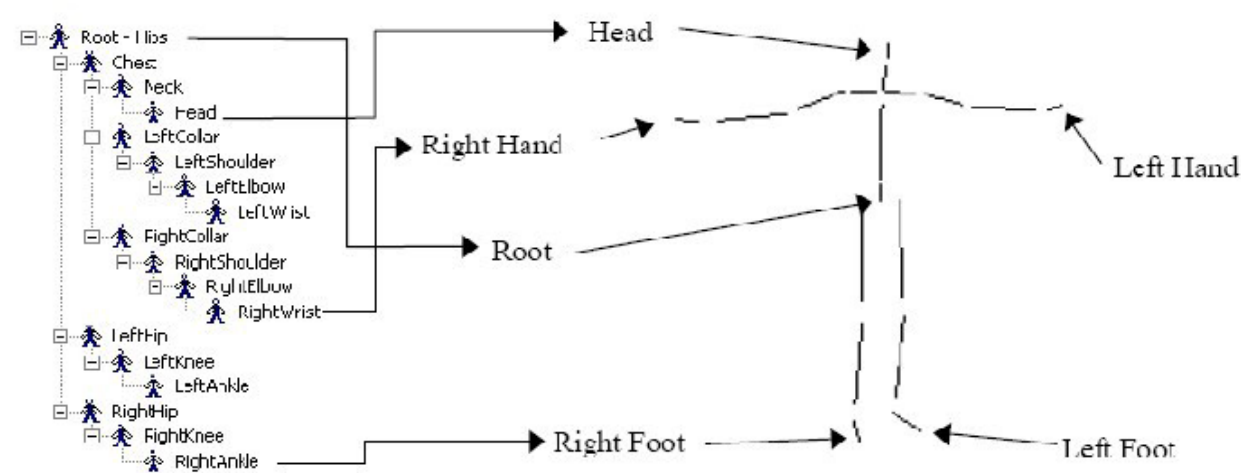


**BVH sample file**:

Given below is a sample bvh file content.

HIERARCHY

ROOT Neck

{

```
OFFSET 0.000000 0.000000 0.000000

CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation

JOINT Head

{

    OFFSET 0.200000 0.400000 0.500000

    CHANNELS 3 Zrotation Xrotation Yrotation

    End Site

    {


        OFFSET 0.600000 8.000000 2.000000

        }



    }


  }
MOTION

Frames: 1

Frame Time: 0.040000

0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

**Detailed description**:

There are two sections in a bvh file.

1. Hierarchy and

2. Motion

Hierarchy:

   - BVH file starts with HIERARCHY keyword. This section contains Root and Joints. And also its channels.

   - We observed some bvh files. All files are having ROOT and JOINTs. And that too ROOT has both channels Position and Rotation. where as JOINT has only Rotation channels.

Motion:

   - This section contains number of frames, frame rate and (more importantly) motion  data. Each line of the motion data represents a single and complete frame.

   -The values in a single line of a motion data = Root channels + Joint channels.

So, from the above sample bvh content, we can map the motion data as follows ...

| Neck | Neck | Neck | Neck | Neck | Neck | Head | Head | Head |
|------|------|------|------|------|------|------|------|------|
| X pos | Y pos | Z pos | Z rot | X rot | Y rot | Z rot | X rot | Y rot |
| 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |

**Legends used in bvh file:**

- **HIERARCHY**: The state of the header section begins with HIERARCHY

- **ROOT**: Root segment

- **JOINT**: Child segment

- **CHANNELS**: Number of channels (position and rotation) of a JOINT

or a ROOT

- **OFFSET**: Length and direction used for drawing the parent

segment

- **End Site**: End of a JOINT

- **MOTION**: The motion section begins with MOTION

- **FRAMES**: Number of frames

○

- **Frame Time**:Sampling rate of the data

If we map the bvh according to our requirement, then

ROOT = Neck

Head = Head or face

JOINT = 3D markers

**Source Code:**

**BVH.h file**

```
handling
//----------------------------
#define NEXT_LINE fwrite("\n", 1, strlen("\n"), m_filePtr)
#define TAB_SPACE fwrite("\t", 1, strlen("\t"), m_filePtr)
#define ONE_SPACE fwrite(" ", 1, strlen(" "), m_filePtr)
#define BEGIN_BRACKET fwrite("{", 1, strlen("{"), m_filePtr)
#define CLOSE_BRACKET fwrite("}", 1, strlen("}"), m_filePtr)
#define OFFSET fwrite("OFFSET", 1, strlen("OFFSET"), m_filePtr)
#define JOINT fwrite("JOINT", 1, strlen("JOINT"), m_filePtr)
#define CHANNELS fwrite("CHANNELS", 1, strlen("CHANNELS"),
m_filePtr)
```

```
#define END_SITE fwrite("End Site", 1, strlen("END Site"),

m_filePtr)

//---------------------------

// Hierarchy

//---------------------------

enum Hierarchy

{

root = 0,

joint,

childJoint

};

//---------------------------

// BVH class

//---------------------------

class BVH

{

public:

BVH();

~BVH();

int CreateBVHFile(char* rootName, bool hasRotChannel, bool

hasPosChannel, float* offsetValues);

int CloseBVHFile();

void SetFileName(CString& fileName);

void ExportToBvh();

protected:

//-- HIERARCHY

int AddHierarchy();

int CloseHierarchy();

int AddJoint(const char* jointName, bool hasRotChannel, bool

hasPosChannel, float* offsetValues);

int AddChildJoint(const char* childName, bool hasRotChannel,

bool hasPosChannel, float* offsetValues);

int AddJointEndSite(float* offsetValues);

int AddChildJointEndSite(float* offsetValues);

int CloseJoint();

int CloseChildJoint();

//-- MOTION
```

```cpp
    int AddMotionHeader(float frameTime);

    int AddMotionFrameData(int currFrame, char* floatValStr);

    int GetNumFrames();

    void SetNumFrames(int numFrames);

    private:

    int AddBody(const char* name, bool hasRotChannel, bool

    hasPosChannel, float* offsetValues, Hierarchy hierarchy);

    int AddEndSite(float* offsetValues, Hierarchy hierarchy);

    FILE* m_filePtr;

    CString m_rootName;

    CString m_fileName;

    bool m_hasRotChannel;

    bool m_hasPosChannel;

    float* m_offsetValues;

    int m_numFrames;

    };
```

**BVH.cpp file:**

```cpp
    #include "stdafx.h"

    #include "BVH.h"

    //--------------------------------------------------------------------

    //BVH constructor

    //--------------------------------------------------------------------

    BVH::BVH()

    {

    m_rootName = "";

    m_fileName = "";

    m_hasRotChannel = false;

    m_hasPosChannel = false;

    m_offsetValues = NULL;

    m_filePtr = NULL;

    m_numFrames = 0;

    }

    //----------------------------------------

    //OnBvhBnClickedOk:

    //----------------------------------------

    void BVH::ExportToBvh()

    {
```

```
//-- create bvh file

float rootOffsetValues[3] = {1.0f, 1.0f, 1.0f};

CreateBVHFile("Neck", true, true, rootOffsetValues);

//-- Add Hierarchy

AddHierarchy();

{

AddJoint("Head", true, false, rootOffsetValues);

AddJointEndSite(rootOffsetValues);

{

for (int i = 0; i < SOME_NUMBER_HERE; ++i)

{

AddChildJoint(SOME_NAME_HERE, false, true,

rootOffsetValues);

AddChildJointEndSite(rootOffsetValues);

CloseChildJoint();

}

}

CloseJoint();

}

CloseHierarchy();

//-- Add Motion data or frame data

SetNumFrames(theApp.m_lastTrackedFrame);

AddMotionHeader(0.040000f);

char motionData[2096] = {0};

for(int f = 1; f <= theApp.m_lastTrackedFrame; ++f)

{

motionData = one complete frame's motoin data here.

AddMotionFrameData(f,motionData);

}

//-- close bvh file

CloseBVHFile();

}

//-------------------------------------------------------------------

//Create: creates the bvh file

//-------------------------------------------------------------------

int BVH::CreateBVHFile(char* rootName, bool hasRotChannel, bool

hasPosChannel, float* offsetValues)
```

```
{

m_rootName.SetString(rootName);

m_hasRotChannel = hasRotChannel;

m_hasPosChannel = hasPosChannel;

m_offsetValues = offsetValues;

m_filePtr = fopen(m_fileName.GetString(), FILE_FLAGS);

if (m_filePtr)

return 1;

return 0;

}

//----------------------------------------------------------------------

//Create: creates the bvh file

//----------------------------------------------------------------------

int BVH::CloseBVHFile()

{

if (m_filePtr)

fclose(m_filePtr);

m_filePtr = NULL;

return 1;

}

//----------------------------------------------------------------------

//BVH estructor

//----------------------------------------------------------------------

BVH::~BVH

{

if (m_filePtr)

fclose(m_filePtr);

m_filePtr = NULL;

}

//----------------------------------------------------------------------

//AddHierarchy: Adds the default hierarchy data into .bvh file

//----------------------------------------------------------------------

int BVH::AddHierarchy()

{

if (!m_filePtr)

return 0;

//Header data
```

```
fwrite("HIERARCHY", 1, strlen("HIERARCHY"), m_filePtr);

NEXT_LINE;

fwrite("ROOT", 1, strlen("ROOT"), m_filePtr);

//body

return AddBody(m_rootName.GetBuffer(), m_hasRotChannel,

m_hasPosChannel, m_offsetValues, root);

}
//----------------------------------------------------------------------

//CloseHierarchy: closes hierarchy
//----------------------------------------------------------------------

int BVH::CloseHierarchy()

{

if (!m_filePtr)

return 0;

NEXT_LINE;

CLOSE_BRACKET;

NEXT_LINE;

return 1;

}
//----------------------------------------------------------------------

//AddJoint: Adds the joint into the .bvh file
//----------------------------------------------------------------------

int BVH::AddJoint(const char* jointName, bool hasRotChannel, bool

hasPosChannel, float* offsetValues)

{

if (!m_filePtr)

return 0;

//only one tab space

TAB_SPACE;

fwrite("JOINT", 1, strlen("JOINT"), m_filePtr);

return AddBody(jointName, hasRotChannel, hasPosChannel,

offsetValues, joint);

}
//----------------------------------------------------------------------

//CloseJoint: closes the joint
//----------------------------------------------------------------------

int BVH::CloseJoint()
```

```
{

if (!m_filePtr)

return 0;

TAB_SPACE;

CLOSE_BRACKET;

return 1;

}

//----------------------------------------------------------------------

//AddJoint: Adds the child joint into the .bvh file

//----------------------------------------------------------------------

int BVH::AddChildJoint(const char* childName, bool hasRotChannel, bool

hasPosChannel, float* offsetValues)

{

if (!m_filePtr)

return 0;

//two tabs

TAB_SPACE;

TAB_SPACE;

fwrite("JOINT", 1, strlen("JOINT"), m_filePtr);

return AddBody(childName, hasRotChannel, hasPosChannel,

offsetValues, childJoint);

}

//----------------------------------------------------------------------

//CloseChildJoint: closes the child joint

//----------------------------------------------------------------------

int BVH::CloseChildJoint()

{

if (!m_filePtr)

return 0;

TAB_SPACE;

TAB_SPACE;

CLOSE_BRACKET;

NEXT_LINE;

return 1;

}

//----------------------------------------------------------------------

//AddJointEndSite: Adds the "End Site" of the joint into .bvh file
```

```
//---------------------------------------------------------------

int BVH::AddJointEndSite(float* offsetValues)

{

TAB_SPACE;

TAB_SPACE;

return AddEndSite(offsetValues, joint);

}

//---------------------------------------------------------------

//AddChildJointEndSite: Adds the "End Site" of the child joint into

.bvh file

//---------------------------------------------------------------

int BVH::AddChildJointEndSite(float* offsetValues)

{

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

return AddEndSite(offsetValues, childJoint);

}

//---------------------------------------------------------------

//AddEndSite: Adds the "End Site" of the joint or child joint

//---------------------------------------------------------------

int BVH::AddEndSite(float* offsetValues, Hierarchy hierarchy)

{

char floatVal[32] = {0};

END_SITE;

NEXT_LINE;

if (hierarchy == joint)

{

TAB_SPACE;

TAB_SPACE;

}

else if (hierarchy == childJoint)

{

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

}
```

```
BEGIN_BRACKET;

NEXT_LINE;

if (hierarchy == joint)

{

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

}

else if (hierarchy == childJoint)

{

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

}

OFFSET;

TAB_SPACE;

sprintf(floatVal,"%f",offsetValues[0]);

fwrite(floatVal, 1, strlen(floatVal), m_filePtr);

ONE_SPACE;

sprintf(floatVal,"%f",offsetValues[0]);

fwrite(floatVal, 1, strlen(floatVal), m_filePtr);

ONE_SPACE;

sprintf(floatVal,"%f",offsetValues[0]);

fwrite(floatVal, 1, strlen(floatVal), m_filePtr);

NEXT_LINE;

if (hierarchy == joint)

{

TAB_SPACE;

TAB_SPACE;

}

else if (hierarchy == childJoint)

{

TAB_SPACE;

TAB_SPACE;

TAB_SPACE;

}
```

```
CLOSE_BRACKET;

NEXT_LINE;

return 1;

}

//---------------------------------------------------------------------

//AddBody: Adds the body of the Root, Joint and child joint

//---------------------------------------------------------------------

int BVH::AddBody(const char* name, bool hasRotChannel, bool

hasPosChannel, float* offsetValues, Hierarchy hierarchy)

{

char floatVal[32] = {0};

if (!m_filePtr)

return 0;

ONE_SPACE;

fwrite(name, 1, strlen(name), m_filePtr);

NEXT_LINE;

if (hierarchy == joint)

{

TAB_SPACE;

}

else if (hierarchy == childJoint)

{

TAB_SPACE;

TAB_SPACE;

}

BEGIN_BRACKET;

NEXT_LINE;

if (hierarchy == root)

{

TAB_SPACE;

}

else if (hierarchy == joint)

{

TAB_SPACE;

TAB_SPACE;

}

else if (hierarchy == childJoint)
```

```
{
TAB_SPACE;
TAB_SPACE;
TAB_SPACE;
}
OFFSET;
TAB_SPACE;
sprintf(floatVal,"%f",offsetValues[0]);
fwrite(floatVal, 1, strlen(floatVal), m_filePtr);
sprintf(floatVal," %f",offsetValues[1]);
fwrite(floatVal, 1, strlen(floatVal), m_filePtr);
sprintf(floatVal," %f",offsetValues[2]);
fwrite(floatVal, 1, strlen(floatVal), m_filePtr);
NEXT_LINE;
if (hierarchy == root)
{
TAB_SPACE;
}
else if (hierarchy == joint)
{
TAB_SPACE;
TAB_SPACE;
}
else if (hierarchy == childJoint)
{
TAB_SPACE;
TAB_SPACE;
TAB_SPACE;
}
//channel data
CHANNELS;
ONE_SPACE;
if (hasRotChannel & hasPosChannel)
{
fwrite("6 Xposition Yposition Zposition Zrotation
Xrotation Yrotation",
1, strlen("6 Xposition Yposition Zposition
```

```
        Zrotation Xrotation Yrotation"), m_filePtr);

        }

        else if(hasRotChannel)

        {

        fwrite("3 Zrotation Xrotation Yrotation",

        1, strlen("3 Zrotation Xrotation Yrotation"),

        m_filePtr);

        }

        else if(hasPosChannel)

        {

        fwrite("3 Xposition Yposition Zposition",

        1, strlen("3 Xposition Yposition Zposition"),

        m_filePtr);

        }

        NEXT_LINE;

        return 1;

        }

        //------------------------------------------------------------------------

        //SetNumFrames: Sets the number frames into the .bvh file

        //------------------------------------------------------------------------

        void BVH::SetFileName(CString& fileName)

        {

        m_fileName = fileName;

        }

        //------------------------------------------------------------------------

        //SetNumFrames: Sets the number frames into the .bvh file

        //------------------------------------------------------------------------

        void BVH::SetNumFrames(int numFrames)

        {

        m_numFrames = numFrames;

        }

        //------------------------------------------------------------------------

        //GetNumFrames: returns the number of frames in a .bvh file

        //------------------------------------------------------------------------

        int BVH::GetNumFrames()

        {

        return m_numFrames;
```

```
    }
    //---------------------------------------------------------------
    //AddMotionHeader: Adds the minimum default motion data into .bvh

    file
    //---------------------------------------------------------------
    int BVH::AddMotionHeader(float frameTime)

    {
    char frameStr[32] = {0};

    fwrite("MOTION", 1, strlen("MOTION"), m_filePtr);

    NEXT_LINE;

    fwrite("Frames:", 1, strlen("Frames:"), m_filePtr);

    TAB_SPACE;

    sprintf(frameStr, "%d", m_numFrames);

    fwrite(frameStr, 1, strlen(frameStr), m_filePtr);

    NEXT_LINE;

    fwrite("Frame Time:", 1, strlen("Frame Time:"), m_filePtr);

    ONE_SPACE;

    sprintf(frameStr, "%f", frameTime);

    fwrite(frameStr, 1, strlen(frameStr), m_filePtr);

    NEXT_LINE;

    return 1;

    }
    //---------------------------------------------------------------
    //AddMotionFrameData: Adds the motion data or frame data into .bvh

    file
    //---------------------------------------------------------------
    int BVH::AddMotionFrameData(int currFrame, char* floatValStr)

    {
    if (currFrame > m_numFrames)

    return 0;

    fwrite(floatValStr, 1, strlen(floatValStr), m_filePtr);

    NEXT_LINE;

    return 1;

    }
```

**Testcase:**

```
#include "stdafx.h"

#include "BVH.h"

int main(int argc, _TCHAR* argv[])

{

if (argc < 2)

return 0;

BVH bvh;

bvh.SetFileName(argv[1]);
```

### Reference:

http://www.cs.cityu.edu.hk/~howard/Teaching/CS4185-5185-2007-SemA/Group12/BVH.html
(http://www.cs.cityu.edu.hk/%7Ehoward/Teaching/CS4185-5185-2007-SemA/Group12/BVH.html)

http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html
(http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html)

By,

Manjunath

Like 51K     G+1 0     Back to Top

## CONTACT US

* Name

* Email

Phone

Enter your message here.

7-1=        Contact Us

## WHY CHOOSE US

Wide Portfolio of Services

Domain Intensive Value Proposition

Technological Excellence across platforms

Avail competitive pricing (Reduce 70% cost, ensure 100% quality)

Get 4 weeks complimentary Risk-Free Trial*

We Build Innovative Applications: Browse through our Intellectual Property (http://www.mindfiresolutions.com/ip.htm)

## CLIENT TESTIMONIAL

"Mindfire has enabled us to reduce costs and increase productivity. The team we have provides fast and reliable development support. We are very happy with the level of service from Mindfire." - *Ken Morency, IceFire Studios Corp.*

More Client Testimonials (client-testimonials.htm)

## NEWS & UPDATES

### January 17 (news.htm#January17)

DDM IT Quiz, Inspired by Mindfire, is Spreading its Wings!

### December 16 (news.htm#December16)

Santa in a different Avatar- International vacations gifted to All Mindfireans!

Learn More (news.htm)

## LET US CONNECT!

Newsletter

*privacy

Awards and Achievements