BlueTiger
Application Programming Interface

Revision 2.00
June 10, 2013

Prepared for

BlueTiger LLC
650 South 79th Street
Chandler, Arizona 85226

by

MMI Development Group, Inc.
1388 West Courtney Lane
Tempe, Arizona 85284
info@mmidev.com

1 Introduction

1.1 Scope

This document describes the Application Programming Interface (API) for the BlueTiger Motion Simulator. It is intended for use by any software developer desiring to add motion simulation to their product.

1.2 Revision History

0.1	18 January 08	S. Johnson	Initial internal release
0.2	21 January 08	S. Johnson	Incorporated changes after review session
0.3	4 February 08	S. Johnson	Added BTAPIVersion and BTFirmwareVersion
0.4	4 March 08	S. Johnson	Modified motion control functions to work with simulators with greater degrees of freedom
0.5	24 March 08	S. Johnson	Modified motion control functions based on review session feedback
0.6	8 May 08	S. Johnson	Modified many functions based on beta test feedback
0.7	13 May 08	S. Johnson	Reverted back to individual parameters rather than bundling parameters into vectors for motion control functions
0.8	9 June 08	S. Johnson	Simplified handling of effects.
0.9	8 December 08	S. Johnson	Deprecated functions that are no longer needed because they are now handled by the new BlueTiger Interface program.
1.00	13 April 09	S. Johnson	Removed description of deprecated functions that are no longer needed because they are now handled by the new BlueTiger Interface program.
2.00	10 June 13	Childress	Added appendix with usage agreement

1.3 Summary

This API provides a very simple interface to the BlueTiger Motion Simulator. One of the goals of this API is to enable a software developer to quickly and easily add motion simulation to a new or existing product. To accomplish this, there are relatively few function calls to learn, and there are several different functions available to move the simulator. These functions take different inputs to perform a move, so the developer can choose the function that best fits the data representation in their existing software.

Following this introduction, Section 2 describes the API functions that initialize and configure the Motion Simulator. Section 3 defines the API functions that move the Motion Simulator, and Section 4 discusses the API functions related to motion effects. Section 5 contains the functions used for running diagnostics, and Section 6 deals with stopping and starting the motion. Sections 7 and 8 contain the status and error codes returned by some of the API functions. Finally, Section 9 defines the structures used by some of the functions and Section 10 gives some examples of how to use these functions in a real application.

1.4 Definitions

World x-axis The x-axis of a right-handed Cartesian coordinate system that is stationary with respect to the simulated world.

World y-axis The y-axis of a right-handed Cartesian coordinate system that is stationary with respect to the simulated world.

World z-axis The z-axis of a right-handed Cartesian coordinate system that is stationary with respect to the simulated world.

Local x-axis The x-axis of a right-handed Cartesian coordinate system that is local to the movable object being simulated. A positive value indicates a position to the right of the object, and a negative value indicates a position to the left of the object.

Local y-axis The y-axis of a right-handed Cartesian coordinate system that is local to the movable object being simulated. A positive value indicates a position above the object, and a negative value indicates a position below the object.

Local z-axis The z-axis of a right-handed Cartesian coordinate system that is local to the movable object being simulated. A positive value indicates a position behind the object, and a negative value indicates a position in front of the object.

Rotation vector A unit vector that remains unchanged by the rotation. This vector defines the axis of rotation of the object being simulated.

Rotation angle The angle of the object being simulated's rotation around the rotation vector. This angle is right-handed.

Forward vector A vector that defines the local negative z-axis in world coordinates.

Right vector A vector that defines the local positive x-axis in world coordinates.

Up vector A vector that defines the local positive y-axis in world coordinates.

2 Initialization

2.1 BTInit

2.1.1 Description

Initializes communication with BlueTiger platform, commands platform to center position and allows platform to accept movement data. This function should be called once before any of the other functions in this API are used.

2.1.2 Declaration

```
unsigned short BTInit (
    char*   company,
    char*   product,
    char*   version
);
```

2.1.3 Parameters

company
A null-terminated character string of up to 40 characters that contains the name of the software development company.

product
A null-terminated character string of up to 40 characters that contains the name of the software program.

version
A null-terminated character string of up to 40 characters that contains the version number of the software program.

2.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

3 Motion Control

3.1 BTPitchRollData

3.1.1 Description

This function moves the platform based on the given pitch and roll of the object being simulated. These parameters are adjusted by the by the current configuration data set by the BlueTiger Interface Program.

3.1.2 Declaration

```
unsigned short BTPitchRollData (
    float   pitch,
    float   roll
);
```

3.1.3 Parameters

pitch
The current pitch of the object being simulated in radians. Positive pitch will raise the front of the simulator, and negative pitch will lower the front of the simulator.

roll
The current roll of the object being simulated in radians. Positive roll will tip the simulator to the left (counter-clockwise when viewed from the rear), and negative roll will tip the simulator to the right (clockwise when viewed from the rear).

3.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

3.2 BTRotationVectorData

3.2.1 Description

This function moves the platform based on the given rotation vector and rotation angle of the object being simulated. These parameters are adjusted by the by the current configuration data set by the BlueTiger Interface Program.

3.2.2 Declaration

```
unsigned short BTRotationVectorData (
    float         xRotation,
    float         yRotation,
    float         zRotation,
    float         aRotation
);
```

3.2.3 Parameters

xRotation
The x-axis portion of the rotation vector of the object being simulated.

yRotation
The y-axis portion of the rotation vector of the object being simulated.

zRotation
The z-axis portion of the rotation vector of the object being simulated.

aRotation
The angle of rotation around the rotation vector in radians. The direction of rotation is right-handed.

3.2.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

3.3 BTAccelerationData

3.3.1 Description

This function takes as its parameters the raw acceleration and current rotational orientation of the object being simulated and calculates the next position of the motion simulator based on this information and the current configuration data set by the BlueTiger Interface Program.

3.3.2 Declaration

```
unsigned short
BTAccelerationData (
    float       xAccel,
    float       yAccel,
    float       zAccel,
    float       xRotAccel,
    float       yRotAccel,
    float       zRotAccel,
    float       xForward,
    float       yForward,
    float       zForward,
    float       xRight,
    float       yRight,
    float       zRight
);
```

3.3.3 Parameters

xAccel
The local x-axis portion of the current acceleration vector in meters/second^2.

yAccel
The local y-axis portion of the current acceleration vector in meters/second^2.

zAccel
The local z-axis portion of the current acceleration vector in meters/second^2.

xRotAccel
The local x-axis portion of the current rotational acceleration vector in radians/second^2.

yRotAccel
The local y-axis portion of the current rotational acceleration vector in radians/second^2.

zRotAccel
The local z-axis portion of the current rotational acceleration vector in radians/second^2.

xForward
The x-axis portion of the forward vector.

yForward
The y-axis portion of the forward vector.

zForward
The z-axis portion of the forward vector.

xRight
The x-axis portion of the right vector.

yRight
The y-axis portion of the right vector.

zRight
The z-axis portion of the right vector.

3.3.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

4 Motion Effects

4.1 BTDefineEffect

4.1.1 Description

This function defines an effect that can be played back later by calling BTPlayEffect or BTStartEffect with the same effectIndex. An effect is intended to be a short sequence of fairly limited movement that creates a short burst of movement not already accounted for in the values provided to the platform through the motion control functions.

4.1.2 Declaration

```
unsigned short BTDefineEffect (
    unsigned char   effectIndex,
    tBTEffect*      effectScript,
    unsigned char   numEntries
);
```

4.1.3 Parameters

effectIndex
The index number of the effect in the effect table.

effectScript
A pointer to an array of tBTEffect entries that define the effect script.

numEntries
The number of entries in the effect script.

4.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

4.2 BTPlayEffect

4.2.1 Description

This function plays the effect defined by BTDefineEffect the given number of times.

4.2.2 Declaration

```
unsigned short BTPlayEffect (
    unsigned char   effectIndex,
    unsigned char   amplitude,
    unsigned char   repetitions
);
```

4.2.3 Parameters

effectIndex
The index number of the effect in the effect table.

amplitude
The amplitude of the effect is multiplied by the percentage given by this parameter. A value of 0 will cause the effect to be ignored. A value of 50 will cause the amplitude of the effect to be reduced by half, and a value of 200 will cause the amplitude of the effect to be doubled. Values greater than 200 cause this command to be ignored.

repetitions
The number of times the effect will be played.

4.2.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

4.3 BTStartEffect

4.3.1 Description

This function starts the playback of the effect defined by BTDefineEffect. The effect will be continuously played until BTStopEffect is called with the same effectIndex.

4.3.2 Declaration

```
unsigned short BTStartEffect (
    unsigned char   effectIndex,
    unsigned char   amplitude
);
```

4.3.3 Parameters

effectIndex
The index number of the effect in the effect table.

amplitude
The amplitude of the effect is multiplied by the percentage given by this parameter. A value of 0 will cause the effect to be ignored. A value of 50 will cause the amplitude of the effect to be reduced by half, and a value of 200 will cause the amplitude of the effect to be doubled. Values greater than 200 cause this command to be ignored.

4.3.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

4.4 BTStopEffect

4.4.1 Description

This function stops the playback of the effect defined by BTDefineEffect.

4.4.2 Declaration

```
unsigned short
BTStopEffect (
    unsigned char   effectIndex
);
```

4.4.3 Parameters

effectIndex
The index number of the effect in the effect table.

4.4.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

5 Diagnostics

5.1 BTAbsoluteMovementData

5.1.1 Description

This function takes as its parameters the desired simulator position.

5.1.2 Declaration

```
unsigned short
BTAbsoluteMovementData (
    float       xPosition,
    float       yPosition,
    float       zPosition,
    float       xRotation,
    float       yRotation,
    float       zRotation,
    float       aRotation
);
```

5.1.3 Parameters

xPosition
The x-axis portion of the simulator position vector in meters.

yAccel
The y-axis portion of the simulator position vector in meters.

zAccel
The z-axis portion of the simulator position vector in meters.

xRotation
The x-axis portion of the simulator rotation vector.

yRotation
The y-axis portion of the simulator rotation vector.

zRotation
The z-axis portion of the simulator rotation vector.

aRotation
The angle of rotation around the rotation vector in radians. The direction of rotation is right-handed.

5.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

5.2 BTRelativeMovementData

5.2.1 Description

This function takes as its parameters the desired offset from the current simulator position.

5.2.2 Declaration

```
unsigned short
BTRelativeMovementData (
    float       xPosition,
    float       yPosition,
    float       zPosition,
    float       xRotation,
    float       yRotation,
    float       zRotation,
    float       aRotation
);
```

5.2.3 Parameters

xPosition
The x-axis portion of the simulator position vector in meters.

yAccel
The y-axis portion of the simulator position vector in meters.

zAccel
The z-axis portion of the simulator position vector in meters.

xRotation
The x-axis portion of the simulator rotation vector.

yRotation
The y-axis portion of the simulator rotation vector.

zRotation
The z-axis portion of the simulator rotation vector.

aRotation
The angle of rotation around the rotation vector in radians. The direction of rotation is right-handed.

5.2.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

6 Termination

6.1 BTPause

6.1.1 Description

This function returns the platform to the center position and stops all further motion until BTResume or BTInit is called.

6.1.2 Declaration

```
unsigned short BTPause (void);
```

6.1.3 Parameters

None

6.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

6.2 BTResume

6.2.1 Description

This function releases the platform from its paused position and allows the various motion control functions to move the platform.

6.2.2 Declaration

```
unsigned short BTResume (void);
```

6.2.3 Parameters

None

6.2.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

6.3 BTShutdown

6.3.1 Description

This function returns the platform to the center position and stops all further motion until BTInit is called. This function should be called when the user exits the motion enabled portion of the application.

6.3.2 Declaration

```
unsigned short BTShutdown (void);
```

6.3.3 Parameters

None

6.3.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

7 Status

7.1 BTStatus

7.1.1 Description

This function returns the current error status.

7.1.2 Declaration

```
unsigned short BTStatus (void);
```

7.1.3 Parameters

None

7.1.4 Return Value

Returns BT_ERROR_NONE if successful and an error code otherwise.

7.2 BTAPIVersion

7.2.1 Description

This function returns the revision number of the BlueTiger API DLL.

7.2.2 Declaration

```
unsigned short BTAPIVersion (void);
```

7.2.3 Parameters

None

7.2.4 Return Value

The revision number of the BlueTiger API DLL. The last two decimal digits represent the minor revision number, and the rest of the decimal digits represent the major revision number. For example, the decimal number 532 should be interpreted as Version 5.32. This function will return zero on error. Call BTStatus to determine the specific error code.

8 Error Codes

0	BT_ERROR_NONE
1	BT_ERROR_COMM_FAIL
9	BT_ERROR_PARAM_OUT_OF_RANGE
10	BT_ERROR_NOT_SUPPORTED
11	BT_ERROR_NO_CONNECT

9 Variable Type Definitions

9.1 tBTVector3

```
typedef struct {
    float   x,                  /* The x-axis portion of the vector */
    float   y,                  /* The y-axis portion of the vector */
    float   z                   /* The z-axis portion of the vector */
} tBTVector3;
```

9.2 tBTEffect

```
typedef struct
{
    unsigned short  duration;           /* Length of movement in ms */
    tBTVector3      positionVector;     /* Offset vector in meters */
    tBTVector3      rotationVector;     /* Rotation vector */
    float           rotationAngle       /* Rotation angle in radians */
} tBTEffect;
```

10 Example Code

Although all of the functions in this API could be used in a single application, there are only a few that are required to move the platform. BTInit must be called before any other API functions are used, and BTShutdown must be called when the user exits the motion enabled portion of the application. To actually perform the movement one of the movement functions (BTPitchRollData, BTRotationVectorData, or BTAccelerationData) is required to be called frequently during the period when motion is being simulated. One call per rendered frame should work well in most applications. BTAccelerationData is the recommended movement function as it provides the most comprehensive information to the platform. BTAbsoluteMovementData and BTRelativeMovementData are intended only for platform diagnostics.

10.1 Initialization

```
BTInit("Company", "Product", "1.0");
```

10.2 Motion Control

```
BTAccelerationData(localAccel.x, localAccel.y, localAccel.z,
                   localRotAccel.x, localRotAccel.y, localRotAccel.z,
                   forwardVector.x, forwardVector.y, forwardVector.z,
                   rightVector.x, rightVector.y, rightVector.z);
```

10.3 Termination

```
BTShutdown();
```

API USAGE AGREEMENT

Use of API. BlueTiger, LLC (BlueTiger) developed the BlueTiger Application Programming Interface (API) that includes a DLL, other software, and documentation to provide to game developers to motion enable games to operate a BlueTiger Motion Simulator. BlueTiger is making the API available to game companies and motion simulator companies without charge to encourage use of the API as an industry standard for motion enabling games.

Agreement. By installing or using the API, the game company or other user (Game Company) agrees to the terms of the API Usage Agreement.

BlueTiger's Use of Game. BlueTiger is authorized to use any game motion enabled by Game Company to demonstrate and market BlueTiger's motion simulator. Such authorized uses include referring to the Game as being motion simulator enabled and using descriptions of the game, pictures of the game's box, screen shots from the game, and video or pictures of the motion simulator using the game in BlueTiger's marketing, promotional, and instructional materials. Game Company shall not be entitled to any compensation from BlueTiger for the uses that are authorized in this section. BlueTiger agrees to discontinue all such uses of the game if Game Company provides BlueTiger with written notice at 650 South 79th St, Chandler, Arizona 85226 requesting that BlueTiger discontinue such uses.

Providing motion enabled game to BlueTiger. Game Company is encouraged to provide a copy of its motion enabled games to BlueTiger so that Blue Tiger may test the game. If BlueTiger finds the game suitable for the BlueTiger Motion Simulator, BlueTiger will include the game in communications with its customers as one of the games suitable for use with the BlueTiger Motion Simulator.

Disclaimer. Some games are not suitable for use with motion simulators, and if not appropriate could cause injury. Game Company agrees that BlueTiger is not responsible for any injury or property damage that occurs from games that were motion enabled using the API. BlueTiger makes no warranties or representations to Game Company concerning the API except that it was developed by BlueTiger.