

Blitzwerks Terrain Documentation

Version 2.02.1

Contents

1 Intro	2
1.1 Features	2
2 Commands	4
2.1 Creating commands	4
2.2 Building commands	13
2.3 Updating commands	15
2.4 Terrain modification commands	18
2.5 Getting commands	28
2.6 Miscellaneous Terrain commands	34
2.7 Miscellaneous commands	36
3 Examples	(WIP)
4 Internals	(WIP)
5 Extras	(WIP)
5.1 Blend Modes	(WIP)

1 Intro

1.1 Features

1.1.1 Basic features

Frustum Culling

This improves speed by preventing DirectX from rendering parts of the terrain which are off screen.

Smoothing

This removes jagged edges on in accurate heightmaps by averaging the height of each point with the points around it.

Level Of Detail

This reduces the detail of parts of the terrain which are far away from the camera so the detail of the terrain and objects close to the camera can be increased greatly improving graphics quality

Saving and Loading

You can save and load terrains with 2 very simple to use commands. This allows very fast loading times with large terrains.

Environment Mapping

You can apply environment maps to your terrains and get the point environment of any point you like. This allows you to set which parts are sand, grass, rock, etc and Blitzwerks Terrain will simply tell you which parts are which environment making footstep sounds much easier.

Custom rendering engine

Blitzwerks Terrain features its own custom rendering engine specifically designed to render terrains. It attaches itself to the DBPro/GDK rendering engine so terrains can be rendered at great speeds inside your games.

Quadtree Rendering

This is alot like BSP but each node has 4 children instead of 2. This allows Blitzwerks Terrain to reduce the amount of draw calls needed in far away areas by merging low detail sectors together and also improves speed in culling calculations.

Multiple camera support

Blitzwerks Terrain includes a set of functions to allow you to render to multiple cameras at very high speed. Usually everything will be recalculated and redrawn individually for each camera but Blitzwerks Terrain allows you to stop certain things from being recalculated such as LOD on 2 cameras which are in the same position.

Limits

Up to 512x512 heightmaps

1 Terrain

Up to 3 LOD Levels

1.1.2 Full version features

Exclusion

This allows you to remove parts of terrains which don't need to be drawn such as areas under buildings or under water. You can use the heightmap as an exclusion map too and set a height at which to exclude everything under which makes underwater exclusion a lot easier

Quad Rotation

This prevents triangles from sticking out the side of steep hills. It calculates which triangles will stick out and rotates them 90 degrees so they face down the hill and don't stick out. This improves graphics quality when using low detail heightmaps so you can turn down the detail of your heightmaps and improve render speed in your games.

Real time Terrain modification

This allows you to edit terrains in realtime. There is a range of functions which allow you to access and change vertexdata inside sectors or use brushes to create editors, craters after explosions, etc.

Quad Reduction

This reduces the detail of the terrain in areas which are flat. This allows you to add more detail into areas which need it which improves the graphics quality of your games.

Colour heightmapping

This allows you to use both the red and green channels of an image to make very accurate terrains. Greyscale heightmaps only allow for 255 steps of height in your terrains so if you want a big mountain with some shallow hills next to it, the hills will have a stepping effect. Colour heightmapping gives you 65000 steps of height so you can have very high mountains but the low areas will still look smooth.

Limits

Up to 2048x2048 heightmaps

Unlimited terrains

Unlimited LOD Levels

2 Commands

2.1 Creating commands

2.1.01 BT MAKETERRAIN

This command makes a terrain and returns its ID. After this command is called, you can set its settings and build it. The ID returned by this function must be used in all functions which are to do with this terrain. In GDK, this doesn't need to be called when using the terrain class as it is called automatically when the object is created.

Parameters and returns:

No Parameters

Returns: TerrainID(integer) - Use this ID in all commands you want to use on this terrain.

Syntax:

DBPro TerrainID=BT MakeTerrain()

GDK int TerrainID=BT_MakeTerrain();

GDK (OOP) BT_Terrain Terrain; // (Make terrain automatically called)

2.1.02 BT SETTERRAINHEIGHTMAP

This command takes a TerrainID and an ImageID and sets the image as the terrain's heightmap. A heightmap must be set before BuildTerrain is called or BuildTerrain will fail. Full version users can set colour heightmaps using this command for increased accuracy, Blitzwerks Terrain will automatically detect when a colour heightmap is used. The heightmaps image must exist until BuildTerrain is called.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Heightmap (integer) – The image ID of the heightmap

Returns: Nothing

Syntax:

DBPro BT SetTerrainHeightMap TerrainID,Heightmap

GDK BT_SetTerrainHeightMap(TerrainID,Heightmap);

GDK (OOP) Terrain->SetHeightmap(Heightmap);

2.1.03 BT SETTERRAINTEXTURE

This command takes a TerrainID and an ImageID and sets the image as the terrain's texture. This can be called at any time on the terrain and Blitzwerks Terrain will change the texture whenever it is called. Textures are applied to texture stage 0.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Texture (integer) – The image ID of the texture

Returns: Nothing

Syntax:

DBPro BT SetTerrainTexture TerrainID,Texture

GDK BT_SetTerrainTexture (TerrainID,Texture);

GDK (OOP) Terrain->SetTexture (Texture);

2.1.04 BT SETTERRAINDETAIL

This command takes a TerrainID and an ImageID and sets the image as the terrain's detailmap. Detailmaps are textures that get applied over the terrain's base texture to increase the detail. This can be called at any time on the terrain and Blitzwerks Terrain will change the detailmap whenever it is called. Detailmaps are applied to texture stage 1.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Detail (integer) – The image ID of the Detailmap

Returns: Nothing

Syntax:

DBPro BT SetTerrainDetail TerrainID, Detail

GDK BT_SetTerrainDetail (TerrainID,Detail);

GDK (OOP) Terrain->SetDetail (Detail);

2.1.05 BT SETTERRAINDETAILTILE

This command sets the tiling of the detailmap. If tile is set to 1 (default), the detailmap will be stretched across each sector. If the tile is set to 2, It will squash the detailmap so it creates 2x2 tiles on each sector. If the tile is set to 0.5, each tile of the detailmap will take up 4 sectors.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Tile (float) – The amount the detailmap should be tiled

Returns: Nothing

Syntax:

DBPro BT SetTerrainDetailTile TerrainID,Tile

GDK BT_SetTerrainDetailTile(TerrainID,Tile);

GDK (OOP) Terrain->SetDetailTile(Tile);

2.1.06 BT SETTERRAINDETAILBLENDMODE

This command sets the way Blitzwerks Terrain blends the detailmap with the base texture. The mode is a number from the list of D3D Blend modes. You can view a list of blend modes in section 5.1.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

BlendMode (integer) – The blend mode (look at section 5.1 for possible values)

Returns: Nothing

Syntax:

DBPro BT SetTerrainDetailBlendMode TerrainID,BlendMode

GDK BT_SetTerrainDetailBlendMode(TerrainID,BlendMode);

GDK (OOP) Terrain->SetDetailBlendMode(BlendMode);

2.1.07 BT SETTERRAINSPPLIT

This command sets the amount of sectors Blitzwerks Terrain makes while building a terrain. Splitting is needed to split up the terrain into smaller chunks called sectors. These can be drawn separately allowing Blitzwerks Terrain to switch/hide parts of the terrain quickly so it can do LOD and culling. This is also needed to allow large terrains to run on old computers. If you set the split value to 2, this will split the terrain into a 2x2 grid of sectors. The split value must be a positive power of 2 number.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Split (integer) – The amount of rows and columns to split the sectors into.

Returns: Nothing

Syntax:

DBPro BT SetTerrainSplit TerrainID,Split

GDK BT_SetTerrainSplit(TerrainID,Split);

GDK (OOP) Terrain->SetSplit(Split);

2.1.08 BT SETTERRAINSCALE

This command sets the scale of the terrain on the X and Z axis. A scale of 0.5 will halve the size of the terrain, 2.0 will Double the size of the terrain and 1.0 will not effect the terrain. The final size of the terrain will be HeightmapSize(in pixels)*Scale.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Scale (float) – The scale value

Returns: Nothing

Syntax:

DBPro BT SetTerrainScale TerrainID,Scale

GDK BT_SetTerrainScale(TerrainID,Scale);

GDK (OOP) Terrain->SetScale(Scale);

2.1.09 BT SETTERRAINYSCALE

This command sets the scale of the terrain on the Y axis. A scale of 0.5 will halve the height of the terrain, 2.0 will Double the height of the terrain and 1.0 will not effect the terrain. The final height of the terrain will be YScale*255.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

YScale (float) – The scale value

Returns: Nothing

Syntax:

DBPro BT SetTerrainYScale TerrainID,YScale

GDK BT_SetTerrainYScale(TerrainID,YScale);

GDK (OOP) Terrain->SetYScale(YScale);

2.1.10 BT SETTERRAINLOD

This command sets the number of LODLevels the terrain has. LOD changes the detail of the terrain automatically in realtime so that far away areas of terrain have low detail and near areas have high detail. This allows you to focus all the detail in the area the camera is in which improves render speed and graphics quality.

You can use BT SetTerrainLODDistance (2.1.11) to control the distance which LOD Details change.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

LODLevels (integer) – The number of LODLevels to create

Returns: Nothing

Syntax:

DBPro BT SetTerrainLOD TerrainID,LODLevels

GDK BT_SetTerrainLOD(TerrainID,LODLevels);

GDK (OOP) Terrain->SetLOD(LODLevels);

2.1.11 BT SETTERRAINLODDISTANCE

This command sets the distance a LOD Level appears. For example, if you have 3 LOD levels, you will need to set LOD Distance 1 and LOD Distance 2. The highest LOD level appears between 0 and LOD Distance 1, The middle LOD level appears between LOD Distance 1 and LOD Distance 2 and the lowest LOD level appears between LOD Distance 2 and the cameras range. You do not need to call this function for LOD level 0.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

LODLevel (integer) – The LOD level to set the distance of

Value (float) – The Distance the LODLevel appears

Returns: Nothing

Syntax:

DBPro BT SetTerrainLODDistance TerrainID,LODLevel,Value

GDK BT_SetTerrainLODDistance(TerrainID,LODLevel,Value);

GDK (OOP) Terrain->SetLODDistance(LODLevel,Value);

2.1.12 BT SETTERRAINENVIRONMENT

This command sets the terrains environment map. Environment maps are maps with coloured areas. Each coloured area is an environment for example, Grass, sand, rock, etc. You can use the BT AddTerrainEnvironment(2.1.13) command to register different environments and the BT GetPointEnvironment(2.5.06) command to get the environment at a point.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

ImageID (integer) – The image ID of the Environment map

Returns: Nothing

Syntax:

DBPro BT SetTerrainEnvironment TerrainID,ImageID

GDK BT_SetTerrainEnvironment(TerrainID,ImageID);

GDK (OOP) Terrain->SetEnvironment(ImageID);

2.1.13 BT ADDTERRAINENVIRONMENT

This command registers an environment on a terrain and returns its ID. The colour parameter should be set to the colour of the areas on the environment map which represents this environment. This command returns an ID to the environment which should be checked against the value returned by BT GetPointEnvironment(2.5.06) command to find if this environment is at a point.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Colour (dword) – The colour of the environment on the environment map (use the rgb command to get this)

Returns: EnvironmentID(integer) – The ID for this environment

Syntax:

DBPro EnvironmentID=BT AddTerrainEnvironment TerrainID,Colour

GDK int EnvironmentID=BT_AddTerrainEnvironment(TerrainID,Colour);

GDK (OOP) int EnvironmentID=Terrain->AddEnvironment(Colour);

2.1.14 BT SETTERRAINQUADREDUCTION – FULL VERSION

This command enables quad reduction. Quad reduction reduces the amount of quads in each sector. It looks for areas which are flat and merges the quads together. This reduces the amount of polygons in view on a terrain.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Enabled (boolean) – Set this to 1 to enable QuadReduction

Returns: Nothing

Syntax:

DBPro BT SetTerrainQuadReduction TerrainID,Enabled

GDK BT_SetTerrainQuadReduction(TerrainID,Enabled);

GDK (OOP) Terrain->SetQuadReduction(Enabled);

2.1.15 BT SETTERRAINQUADROTATION

This command enables quad rotation. Quad rotation rotates the triangles in each quad in a way to prevent the triangles from sticking out causing jagged edges all over your terrain. It makes the terrain look a lot smoother.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Enabled (boolean) – Set this to 1 to enable QuadRotation

Returns: Nothing

Syntax:

DBPro BT SetTerrainQuadRotation TerrainID,Enabled

GDK BT_SetTerrainQuadRotation(TerrainID,Enabled);

GDK (OOP) Terrain->SetQuadRotation(Enabled);

2.1.16 BT SETTERRAINSMOOTHING

This command enables smoothing on the terrain. When enabled, smoothing averages each point with the points around it to smooth out sharp edges on terrains. It also helps reduce stepping caused by inaccurate heightmaps.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Enabled (boolean) – Set this to 1 to enable QuadRotation

Returns: Nothing

Syntax:

DBPro BT SetTerrainQuadRotation TerrainID,Enabled

GDK BT_SetTerrainQuadRotation(TerrainID,Enabled);

GDK (OOP) Terrain->SetQuadRotation(Enabled);

2.1.17 BT SETTERRAINEXCLUSION – FULL VERSION

This command sets the terrain's exclusion map. Exclusion maps are greyscale maps which cull quads at points where the brightness is less than the exclusion threshold. You can set the exclusion threshold with the BT SetTerrainExclusionThreshold(2.1.18) command. Exclusion is useful for culling quads which are under buildings or underwater.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

ImageID (integer) – The image ID of the Exclusion map

Returns: Nothing

Syntax:

DBPro BT SetTerrainExclusion TerrainID,ImageID

GDK BT_SetTerrainExclusion(TerrainID,ImageID);

GDK (OOP) Terrain->SetExclusion(ImageID);

2.1.18 BT SETTERRAINEXCLUSIONTHRESHOLD – FULL VERSION

This command sets the brightness level a point on the exclusion map needs to be less than in order to be culled. This is useful for setting the height of the water on your map and using the heightmap as the exclusion map.

Parameters and returns:

TerrainID (integer) – The ID of the terrain

Threshold (integer) – The threshold. This must be between 0 and 255,

Returns: Nothing

Syntax:

DBPro BT SetTerrainExclusionThreshold TerrainID,Threshold

GDK BT_SetTerrainExclusionThreshold(TerrainID,Threshold);

GDK (OOP) Terrain->SetExclusionThreshold(Threshold);

2.2 Building commands

2.2.01 BT SETBUILDSTEP

This command sets the number of sectors that get built in each call of BT ContinueBuild(2.2.04).

Parameters and returns:

BuildStep(integer) – The number of sectors to build.

Returns: Nothing

Syntax:

DBPro BT SetBuildStep BuildStep

GDK BT_SetBuildStep(BuildStep);

GDK (OOP) Blitzwerks Terrain->SetBuildStep(BuildStep);

2.2.02 BT BUILDTERRAIN

This command builds a terrain. The ObjectID must not be 0 and must be unused. This is the ID Blitzwerks Terrain will create its object into. You can use DBPro functions on this ID to allow you to use all the object functions on the terrain. If build fully is set to 1, The terrain will be built inside this call and the terrain will be ready for use without a call to ContinueBuild(2.2.04).

Parameters and returns:

TerrainID(integer) – The ID of the terrain

ObjectID(integer) – An unused object id for the terrains object

BuildFully(boolean) – optional – Set this to 1 to make the terrain build completely inside this call.

Returns: Nothing

Syntax:

DBPro BT BuildTerrain TerrainID,ObjectID
BT BuildTerrain TerrainID,ObjectID,BuildFully

GDK BT_BuildTerrain(TerrainID,ObjectID);
BT_BuildTerrain(TerrainID,ObjectID,BuildFully);

GDK (OOP) Terrain->Build(ObjectID);
Terrain->Build(ObjectID,BuildFully);

2.2.03 BT LOADTERRAIN

This command loads a terrain from a file. This will do everything between BT MakeTerrain(2.1.01) and BT BuildTerrain(2.2.02). You will have to apply your texture and detailmap again as these are not saved inside the terrain file.

Parameters and returns:

Filename(string) – The filename of the terrain

ObjectID(integer) – An unused object id for the terrains object

BuildFully(boolean) – optional – Set this to 1 to make the terrain build completely inside this call.

Returns: TerrainID(integer) - Use this ID in all commands you want to use on this terrain.

Syntax:

DBPro	BT LoadTerrain FileName,ObjectID BT LoadTerrain FileName,ObjectID,BuildFully
GDK	BT_LoadTerrain(FileName,ObjectID); BT_LoadTerrain(FileName,ObjectID,BuildFully);
GDK (OOP)	Blitzwerks Terrain->LoadTerrain(FileName,ObjectID); Blitzwerks Terrain->LoadTerrain(FileName,BuildFully);

2.2.04 BT CONTINUEBUILD

This command generates sectors for the terrain currently being built. You can change the number of sectors it builds each call by using the BT SetBuildStep(2.2.01) command. This command is useful for creating loading bars for your program. It returns the current percentage of how much terrain has been built. If it returns -1 then the terrain is built and can be used.

Parameters and returns:

No Parameters

Returns: Progress(integer) – The percentage of terrain built, or -1 if the terrain is built.

Syntax:

DBPro	Progress=BT ContinueBuild()
GDK	int Progress=BT_ContinueBuild();
GDK (OOP)	int Progress=Terrain->ContinueBuild();

2.3 Updating commands

2.3.01 BT SETCURRENTCAMERA

This command sets the camera which should be used for culling terrains and updating LOD. This is useful when multiple cameras are used, Blitzwerks Terrain can update and render terrains individually for each camera instead of displaying everything that every camera can see to every camera. This greatly improves speed. It also allows for Blitzwerks Terrain to update LOD individually for cameras which are in different locations.

Parameters and returns:

CameraID(integer) – The DBPro camera to make Blitzwerks Terrain use.

Returns: Nothing

Syntax:

DBPro BT SetCurrentCamera CameraID

GDK BT_SetCurrentCamera(CameraID);

GDK (OOP) Blitzwerks Terrain->SetCurrentCamera(CameraID);

2.3.02 BT UPDATETERRAINCULL

This command updates the culling for a terrain. Culling checks which parts of the terrain are in the cameras view. The parts which are not get removed temporarily to improve render speed. The cull area is taken from the current camera (set through BT SetCurrentCamera(2.3.01)).

Parameters and returns:

TerrainID(integer) – The ID of the terrain you want to update the cull for

Returns: Nothing

Syntax:

DBPro BT UpdateTerrainCull TerrainID

GDK BT_UpdateTerrainCull(TerrainID);

GDK (OOP) Terrain->UpdateCull();

2.3.03 BT UPDATETERRAINLOD

This command updates the LOD for a terrain. This doesn't need to be called when there is only 1 LOD level. LOD reduces the detail in far away areas improving render speed which allows you to increase the detail/size of your terrains with very little impact on frame rate. The LOD position is taken from the current cameras position (set through BT SetCurrentCamera(2.3.01)).

Parameters and returns:

TerrainID(integer) – The ID of the terrain you want to update the LOD for
Returns: Nothing

Syntax:

DBPro	BT UpdateTerrainLOD TerrainID
GDK	BT_UpdateTerrainLOD(TerrainID);
GDK (OOP)	Terrain->UpdateLOD();

2.3.04 BT UPDATETERRAIN

This function is BT UpdateTerrainCull(2.3.02) and BT UpdateTerrainLOD(2.3.03) put together. It does not need to be called if you use these 2 functions instead.

Parameters and returns:

TerrainID(integer) – The ID of the terrain you want to update
Returns: Nothing

Syntax:

DBPro	BT UpdateTerrain TerrainID
GDK	BT_UpdateTerrain(TerrainID);
GDK (OOP)	Terrain->Update();

2.3.05 BT RENDER TERRAIN

This command draws the terrain to the current camera. This must be called every loop before sync is called.

Parameters and returns:

TerrainID(integer) – The ID of the terrain you want to render
Returns: Nothing

Syntax:

DBPro	BT RenderTerrain TerrainID
GDK	BT_RenderTerrain(TerrainID);
GDK (OOP)	Terrain->Render();

2.4 Terrain modification commands

2.4.01 BT LOCKVERTEXDATAFORSECTOR – FULL VERSION

This command locks the vertexdata of a sector. It allows you to use the other vertexdata commands on this sector. You can only lock one sector at a time and sectors are automatically unlocked in the sync command.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on
LODLevel(integer) – The LODLevel of the terrain the sector is on
Sector(integer) – The ID of the sector
Returns: Nothing

Syntax:

DBPro	BT LockVertexDataForSector TerrainID,LODLevel,Sector
GDK	BT_LockVertexDataForSector(TerrainID,LODLevel,Sector);
GDK (OOP)	N/A

2.4.02 BT LOCKEDASECTOR – FULL VERSION

This command returns 1 if a sector is locked and 0 if there are no locked sectors.

Parameters and returns:

No parameters
Returns: Sector locked(boolean) - 1 if there is a locked sector, otherwise 0

Syntax:

DBPro	SectorLocked=BT LockedASector()
GDK	int SectorLocked=BT_LockedASector();
GDK (OOP)	N/A

2.4.03 BT GETLOCKEDTERRAIN – FULL VERSION

If a sector has been locked, this returns the ID of the terrain which the locked sector belongs to

Parameters and returns:

No parameters

Returns: TerrainID(integer) - The ID of the terrain the locked sector is in

Syntax:

DBPro TerrainID=BT GetLockedTerrain()

GDK int TerrainID=BT_GetLockedTerrain();

GDK (OOP) N/A

2.4.04 BT GETLOCKEDLODLEVEL – FULL VERSION

If a sector has been locked, this returns the ID of the LOD level on the terrain which the locked sector belongs to

Parameters and returns:

No parameters

Returns: LODLevel(integer) - The ID of the LOD level the locked sector is in

Syntax:

DBPro LODLevel=BT GetLockedLODLevel()

GDK int LODLevel=BT_GetLockedLODLevel();

GDK (OOP) N/A

2.4.05 BT GETLOCKEDSECTOR – FULL VERSION

If a sector has been locked, this returns the ID of the locked sector on the terrain which the locked sector belongs to.

Parameters and returns:

No parameters

Returns: SectorID(integer) - The ID of the locked sector

Syntax:

DBPro SectorID=BT GetLockedSector()

GDK int SectorID=BT_GetLockedSector();

GDK (OOP) N/A

2.4.06 BT UNLOCKVERTEXDATA – FULL VERSION

This command unlocks the locked sector. All edited vertices will automatically change on the terrain.

Parameters and returns:

No parameters

Returns Nothing

Syntax:

DBPro BT UnlockVertexdata()

GDK BT_UnlockVertexdata();

GDK (OOP) N/A

2.4.07 BT GETVERTEXCOUNT – FULL VERSION

This command returns the number of vertices in the current locked sector.

Parameters and returns:

No parameters

Returns: VertexCount(integer) - The number of vertices in the locked sector

Syntax:

DBPro VertexCount=BT GetVertexCount()

GDK int VertexCount=BT_GetVertexCount();

GDK (OOP) N/A

2.4.08 BT GETINDEXCOUNT – FULL VERSION

This command returns the number of indices in the current locked sector.

Parameters and returns:

No parameters

Returns: IndexCount(integer) - The number of indices in the locked sector

Syntax:

DBPro IndexCount=BT GetIndexCount()

GDK int IndexCount=BT_GetIndexCount();

GDK (OOP) N/A

2.4.09 BT GETINDEX – FULL VERSION

This command returns the value of the specified index on the current locked sector. There are 3 indices per triangle. Each index is a vertex number. For example, the 3 vertices for the first triangle in the sector can be found by doing, BT GetIndex(0), BT GetIndex(1) and BT GetIndex(2). You can use the numbers returned by these in the vertex commands to change each vertex on this triangle.

Parameters and returns:

IndexID(integer) – The ID of the index

Returns: VertexID(integer) – The vertex at this index

Syntax:

DBPro VertexID=BT GetIndex(IndexID)

GDK int VertexID=BT_GetIndex(IndexID);

GDK (OOP) N/A

2.4.10 BT GETVERTEXPOSITIONX – FULL VERSION

This command returns the x coordinate of the specified vertex on the current locked sector.

Parameters and returns:

VertexID(integer) – The ID of the vertex

Returns: XCoordinate(float) – The vertex X position

Syntax:

DBPro XCoordinate=BT GetVertexPositionX(VertexID)

GDK float XCoordinate=BT_GetVertexPositionX(VertexID);

GDK (OOP) N/A

2.4.11 BT GETVERTEXPOSITIONY – FULL VERSION

This command returns the y coordinate of the specified vertex on the current locked sector.

Parameters and returns:

VertexID(integer) – The ID of the vertex

Returns: YCoordinate(float) – The vertex Y position

Syntax:

DBPro YCoordinate=BT GetVertexPositionY(VertexID)

GDK float YCoordinate=BT_GetVertexPositionY(VertexID);

GDK (OOP) N/A

2.4.12 BT GETVERTEXPOSITIONZ – FULL VERSION

This command returns the z coordinate of the specified vertex on the current locked sector.

Parameters and returns:

VertexID(integer) – The ID of the vertex

Returns: ZCoordinate(float) – The vertex Z position

Syntax:

DBPro ZCoordinate=BT GetVertexPositionZ(VertexID)

GDK float ZCoordinate=BT_GetVertexPositionZ(VertexID);

GDK (OOP) N/A

2.4.13 BT GETVERTEXPOSITIONX – FULL VERSION

This command does the same as BT GetVertexPositionX(2.4.10) but this addresses each vertex by its row and column.

Parameters and returns:

Row(integer) – The row of the vertex

Column(integer) – The column of the vertex

Returns: XCoordinate(float) – The vertex X position

Syntax:

DBPro XCoordinate=BT GetVertexPositionX(Row,Column)

GDK float XCoordinate=BT_GetVertexPositionX(Row,Column);

GDK (OOP) N/A

2.4.14 BT GETVERTEXPOSITIONY – FULL VERSION

This command does the same as BT GetVertexPositionY(2.4.11) but this addresses each vertex by its row and column.

Parameters and returns:

Row(integer) – The row of the vertex

Column(integer) – The column of the vertex

Returns: YCoordinate(float) – The vertex Y position

Syntax:

DBPro YCoordinate=BT GetVertexPositionY(Row,Column)

GDK float YCoordinate=BT_GetVertexPositionY(Row,Column);

GDK (OOP) N/A

2.4.15 BT GETVERTEXPOSITIONZ – FULL VERSION

This command does the same as BT GetVertexPositionZ(2.4.12) but this addresses each vertex by its row and column.

Parameters and returns:

Row(integer) – The row of the vertex

Column(integer) – The column of the vertex

Returns: ZCoordinate(float) – The vertex Z position

Syntax:

DBPro ZCoordinate=BT GetVertexPositionZ(Row,Column)

GDK float ZCoordinate=BT_GetVertexPositionZ(Row,Column);

GDK (OOP) N/A

2.4.16 BT SETVERTEXHEIGHT – FULL VERSION

This command sets the height of a vertex (Y coordinate). You can use this to modify the sector in realtime.

Parameters and returns:

VertexID(integer) – The ID of the vertex

Height(float) – The new height of the vertex

Returns: Nothing

Syntax:

DBPro BT SetVertexHeight VertexID,Height

GDK BT_SetVertexHeight(VertexID,Height);

GDK (OOP) N/A

2.4.17 BT SETVERTEXHEIGHT – FULL VERSION

This command does the same as BT SetVertexHeight(2.4.16) but this addresses each vertex by its row and column.

Parameters and returns:

Row(integer) – The row of the vertex
Column(integer) – The column of the vertex
Height(float) – The new height of the vertex
Returns: Nothing

Syntax:

DBPro	BT SetVertexHeight Row,Column,Height
GDK	BT_SetVertexHeight(Row,Column,Height);
GDK (OOP)	N/A

2.4.18 BT SETPOINTHEIGHT – FULL VERSION

This command sets the height of a point on the terrain. You do not need to lock any sectors to use this function. It will set the height on multiple sectors if the point is on the edge of a sector, and it will also set the height through all LOD levels. This is the recommended method of setting the height of points but is a bit slower than doing it using the above functions.

Parameters and returns:

TerrainID(integer) – The terrain which the vertex is on
Row(integer) – The row of the vertex
Column(integer) – The column of the vertex
Height(float) – The new height of the vertex
Returns: Nothing

Syntax:

DBPro	BT SetPointHeight TerrainID,Row,Column,Height
GDK	BT_SetPointHeight(TerrainID,Row,Column,Height);
GDK (OOP)	N/A

2.4.19 BT RAISETERRAIN – FULL VERSION

This command raises a circular piece of terrain at the point specified. It will raise the middle point the most and the points near the radius the least.

This command uses the following function on each point in the radius:

$$\text{NewHeight} = \text{CurrentHeight} + ((\cos(\text{float}((\text{DistanceFromCentre} * 1.57079633) / (\text{Radius} / 2.0f))) + 1.0f) / 2.0f) * \text{Amount}$$

You can turn this function into a lower terrain function by using a negative amount.

Parameters and returns:

TerrainID(integer) – The terrain which the vertex is on

X(float) – The X position of the middle of the brush

Z(float) – The Z position of the middle of the brush

Radius(float) – The radius of the brush

Amount(float) – The amount the middle point should move

Returns: Nothing

Syntax:

DBPro BT RaiseTerrain TerrainID,X,Z,Radius,Amount

GDK BT_RaiseTerrain(TerrainID,X,Z,Radius,Amount);

GDK (OOP) N/A

2.4.21 BT FLATTENTERRAIN – FULL VERSION

This function flattens a circular area of terrain. It sets the height of all the vertices in the radius to the height of the middle point.

Parameters and returns:

TerrainID(integer) – The terrain which the vertex is on

X(float) – The X position of the middle of the brush

Z(float) – The Z position of the middle of the brush

Radius(float) – The radius of the brush

Returns: Nothing

Syntax:

DBPro BT FlattenTerrain TerrainID,X,Z,Radius

GDK BT_FlattenTerrain(TerrainID,X,Z,Radius);

GDK (OOP) N/A

2.4.22 BT PAINTTERRAIN – FULL VERSION

This function sets the colour of an area of terrain. You can either set the whole area to a fixed colour or to random colours in a range.

Parameters and returns:

TerrainID(integer) – The terrain which to paint
X(float) – The X position of the middle of the brush
Z(float) – The Z position of the middle of the brush
Radius(float) – The radius of the brush
Colour(dword) – The colour to set to this area
MinColour(dword) – The minimum colour of the range to set to this area
MaxColour(dword) – The maximum colour of the range to set to this area
Returns: Nothing

Syntax:

DBPro	BT PaintTerrain TerrainID,X,Z,Radius,Colour BT PaintTerrain TerrainID,X,Z,Radius,MinColour,MaxColour
GDK	BT_PaintTerrain(TerrainID,X,Z,Radius,Colour); BT_PaintTerrain(TerrainID,X,Z,Radius,MinColour,MaxColour);
GDK (OOP)	N/A

2.4.23 BT SETPOINTCOLOUR – FULL VERSION

This function sets the colour of a point on a terrain.

Parameters and returns:

TerrainID(integer) – The terrain which the point is on
X(float) – The X position of the middle of the brush
Z(float) – The Z position of the middle of the brush
Colour(dword) – The colour to set this point to
Returns: Nothing

Syntax:

DBPro	BT SetPointColour TerrainID,X,Z,Colour
GDK	BT_SetPointColour(TerrainID,X,Z,Colour);
GDK (OOP)	N/A

2.4.24 BT UPDATETERRAINTEXTURE – FULL VERSION

This function applies the changes to the texture to all mip levels of the texture. You only need to call this once between the last texture modification and the sync command. You do not need to call this when there are no other mip levels.

Parameters and returns:

TerrainID(integer) – The terrain with the texture that needs updating

Returns: Nothing

Syntax:

DBPro	BT UpdateTerrainTexture TerrainID
GDK	BT_UpdateTerrainTexture(TerrainID);
GDK (OOP)	N/A

2.5 Getting commands

2.5.01 BT GETGROUNDHEIGHT

This function gets the height of a specified point. You can use this for positioning objects on a terrain.

Parameters and returns:

TerrainID(integer) – The terrain which the point is on

X(float) – The points X coordinate

Z(float) – The points Z coordinate

Returns: Height(float) – The Height of the point

Syntax:

DBPro Height=BT GetGroundHeight(TerrainID,X,Z)

GDK float Height=BT_GetGroundHeight(TerrainID,X,Z);

GDK (OOP) float Height=Terrain->GetGroundHeight(X,Z);

2.5.02 BT GETPOINTEXCLUDED – FULL VERSION

If an exclusion map has been applied. This function checks if a point is excluded. This can be used to make sure objects aren't positioned in areas which don't exist.

Parameters and returns:

TerrainID(integer) – The terrain which the point is on

X(float) – The points X coordinate

Z(float) – The points Z coordinate

Returns: Excluded(boolean) – returns 1 if the point is excluded, otherwise returns 0

Syntax:

DBPro Excluded =BT GetPointExcluded(TerrainID,X,Z)

GDK bool Excluded=BT_GetPointExcluded(TerrainID,X,Z);

GDK (OOP) bool Excluded=Terrain->GetPointExcluded(X,Z);

2.5.03 BT GETSECTORCOUNT

This function gets the number of sectors on the specified LOD level of the specified terrain.
This would usually be: $(Split/2^{LODLevel})^2$

Parameters and returns:

TerrainID(integer) – The terrain ID

LODLevel(integer) – The LOD level to get the sector count of. Starting from 0 (highest)

Returns: SectorCount(integer) – The number of sectors on this LOD level

Syntax:

DBPro SectorCount =BT GetSectorCount(TerrainID,LODLevel)

GDK int SectorCount=BT_GetSectorCount(TerrainID,LODLevel);

GDK (OOP) int SectorCount=Terrain->GetSectorCount(LODLevel);

2.5.04 BT TERRAINEXIST

This function finds if the specified terrain has been made with BT MakeTerrain(2.1.01) but not deleted with BT DeleteTerrain(2.6.01).

Parameters and returns:

TerrainID(integer) – The terrain ID

Returns: Exists(boolean) – returns 1 if the terrain exists, otherwise returns 0

Syntax:

DBPro Exists =BT TerrainExist(TerrainID)

GDK bool Exists=BT_TerrainExist(TerrainID);

GDK (OOP) N/A

2.5.05 BT GETTERRAINOBJECTID

This function returns the ID of the terrains object which was set in BT BuildTerrain(2.2.02).

Parameters and returns:

TerrainID(integer) – The terrain ID

Returns: ObjectID(integer) – The DBPro object for the terrain

Syntax:

DBPro ObjectID=BT GetTerrainObjectID(TerrainID)

GDK int ObjectID=BT_GetTerrainObjectID(TerrainID);

GDK (OOP) int ObjectID=Terrain->GetObjectID();

2.5.06 BT GETPOINTENVIRONMENT

If an environment map has been applied, this function will return the ID of the environment at a point. If the Environment at this point has not been added with BT AddTerrainEnvironment(2.1.13), this will return 0.

Parameters and returns:

TerrainID(integer) – The terrain which the point is on

X(float) – The points X coordinate

Z(float) – The points Z coordinate

Returns: Environment(integer) – The environment ID at this point

Syntax:

DBPro Environment=BT GetPointEnvironment(TerrainID,X,Z)

GDK int Environment=BT_GetPointEnvironment(TerrainID,X,Z);

GDK (OOP) N/A

2.5.07 BT GETTERRAINSIZE

This function gets the total size of a terrain in DBPro units. You can use this to make sure objects don't go off the edge of the terrain

Parameters and returns:

TerrainID(integer) – The terrain to get the size of

Returns: Size(float) – The total size of the terrain in DBPro units

Syntax:

DBPro Size=BT GetTerrainSize(TerrainID)

GDK float Size=BT_GetTerrainSize(TerrainID);

GDK (OOP) float Size=Terrain->GetSize();

2.5.08 BT GETSECTORPOSITIONX

This gets the X position in DBPro units of the specified sector.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on

LODLevel(integer) – The LODLevel of the terrain the sector is on

Sector(integer) – The ID of the sector

Returns: X Position(float) – The X position in DBPro units of the sector.

Syntax:

DBPro XPosition=BT GetSectorPositionX(TerrainID,LODLevel,Sector)

GDK float XPosition=BT_GetSectorPositionX(TerrainID,LODLevel,Sector)

GDK (OOP) float XPosition=Terrain->GetSectorPositionX(LODLevel,Sector)

2.5.09 BT GETSECTORPOSITIONY

This gets the Y position in DBPro units of the specified sector.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on

LODLevel(integer) – The LODLevel of the terrain the sector is on

Sector(integer) – The ID of the sector

Returns: Y Position(float) – The Y position in DBPro units of the sector.

Syntax:

DBPro YPosition=BT GetSectorPositionY(TerrainID,LODLevel,Sector)

GDK float YPosition=BT_GetSectorPositionY(TerrainID,LODLevel,Sector)

GDK (OOP) float Yposition=Terrain->GetSectorPositionY(LODLevel,Sector)

2.5.10 BT GETSECTORPOSITIONZ

This gets the Z position in DBPro units of the specified sector.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on

LODLevel(integer) – The LODLevel of the terrain the sector is on

Sector(integer) – The ID of the sector

Returns: Z Position(float) – The Z position in DBPro units of the sector.

Syntax:

DBPro ZPosition=BT GetSectorPositionZ(TerrainID,LODLevel,Sector)

GDK float ZPosition=BT_GetSectorPositionZ(TerrainID,LODLevel,Sector)

GDK (OOP) float ZPosition=Terrain->GetSectorPositionZ(LODLevel,Sector)

2.5.11 BT GETSECTOREXCLUDED – FULL VERSION

This command checks if the specified sector is excluded. Sectors can be excluded if every single quad on the sector is excluded with an exclusion map. You can use this function to make sure the sector exists before trying to use one of the other sector commands on it.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on

LODLevel(integer) – The LODLevel of the terrain the sector is on

Sector(integer) – The ID of the sector

Returns: Excluded(boolean) – 1 if the sector is excluded, 0 if its not excluded.

Syntax:

DBPro Excluded=BT GetSectorExcluded(TerrainID,LODLevel,Sector)

GDK bool Excluded=BT_GetSectorExcluded(TerrainID,LODLevel,Sector)

GDK (OOP) bool Excluded =Terrain->GetSectorExcluded(LODLevel,Sector)

2.6 Miscellaneous Terrain commands

2.6.01 BT DELETETERRAIN

This command deletes a terrain. The terrain must be past the BT MakeTerrain(2.1.01) stage in order to be deleted. When terrains are deleted their ID will be re used if a new terrain is created afterwards.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on
Returns: Nothing

Syntax:

DBPro	BT DeleteTerrain(TerrainID)
GDK	BT_DeleteTerrain(TerrainID);
GDK (OOP)	Terrain->Delete(); or delete Terrain;

2.6.02 BT MAKESECTOROBJECT

This command creates a DBPro object from the vertex data of the specified sector. These objects should only be used for getting vertex data from a terrain and setting up collision. They should not be used for visuals.

Parameters and returns:

TerrainID(integer) – The ID of the terrain the sector is on
LODLevel(integer) – The LODLevel of the terrain the sector is on
Sector(integer) – The ID of the sector
ObjectID(integer) – The DBPro object ID to create the sector object in.
Returns: Nothing

Syntax:

DBPro	BT MakeSectorObject(TerrainID,LODLevel,Sector,ObjectID)
GDK	BT_MakeSectorObject(TerrainID,LODLevel,Sector,ObjectID);
GDK (OOP)	Terrain->MakeSectorObject(LODLevel,Sector,ObjectID);

2.6.03 BT SAVETERRAIN

This command saves a terrain into Blitzwerks Terrains native format. The file extension can be anything you want. The files produced can only be loaded with BT LoadTerrain. If you have the full version of Blitzwerks Terrain and save a terrain which uses full version features. It cannot be loaded into the free version of Blitzwerks Terrain.

Parameters and returns:

TerrainID(integer) – The ID of the terrain to save

FileName(string) – The filename to save the terrain as

Returns: Nothing

Syntax:

DBPro BT SaveTerrain(TerrainID,FileName)

GDK BT_SaveTerrain(TerrainID,FileName);

GDK (OOP) Terrain->Save(FileName);

2.6.04 BT MAKETERRAINOBJECT

This command creates a DBPro object from the vertex data of the specified terrain. These objects should only be used for getting vertex data from a terrain and setting up collision. They should not be used for visuals.

Parameters and returns:

TerrainID(integer) – The ID of the terrain

LODLevel(integer) – The LODLevel to use for the object

ObjectID(integer) – The DBPro object ID to create the terrain object in.

Returns: Nothing

Syntax:

DBPro BT MakeTerrainObject(TerrainID,LODLevel,ObjectID)

GDK BT_MakeTerrainObject(TerrainID,LODLevel,ObjectID);

GDK (OOP) Terrain->MakeObject(LODLevel,ObjectID);

2.6.05 BT UPDATETERRAINOBJECTS

This command repositions the DBPro objects of a terrain (created with BT MakeSectorObject and BT MakeTerrainObject) to the same position as the terrain. This makes repositioning terrains with collision objects a lot easier.

Parameters and returns:

TerrainID(integer) – The ID of the terrain

Returns: Nothing

Syntax:

DBPro BT UpdateTerrainObjects TerrainID

GDK BT_UpdateTerrainObjects(TerrainID);

GDK (OOP) Terrain->UpdateObjects();

2.7 Miscellaneous commands

2.7.01 BT SETATMODE

This command makes Blitzwerks Terrain emulate Advanced Terrains method of loading heightmaps and applying textures. This should be used if you are using a heightmap designed to work with Advanced Terrain. Advanced Terrain rotates and mirrors the terrain but Blitzwerks Terrain doesn't. This command solves this issue.

Parameters and returns:

Enabled(boolean) – Set this to 1 to enable AT mode. Set this to 0 to disable it.
Returns: Nothing

Syntax:

DBPro	BT SetATMode(Enabled)
GDK	BT_SetATMode(Enabled);
GDK (OOP)	Blitzwerks Terrain->SetATMode(Enabled);

2.7.02 BT GETVERSION

This command returns a string with the version of Blitzwerks Terrain that is currently being used.

Parameters and returns:

No Parameters
Returns: Version(string) – A string with the version info. (char* in C++)

Syntax:

DBPro	Version\$=BT GetVersion()
GDK	char* Version=BT_GetVersion();
GDK (OOP)	char* Version=Blitzwerks Terrain->GetVersion();

2.7.03 BT GETSTATISTIC

This command returns a statistic. The only parameter is the statistic code which sets which statistic to return. The values returned is the amount of work Blitzwerks Terrain has done so far in its render loop. To get statistics of all the terrains put together, I recommend that you call this after rendering all the terrains and before sync is called.

Code	Returns
1	Polygon count
2	Draw calls
3	Cull checks

Example:

DBPro:

```
PolygonCount=BT GetStatistic(1)
```

```
DrawCalls=BT GetStatistic(2)
```

```
CullChecks=BT GetStatistic(2)
```

GDK:

```
int PolygonCount=BT_GetStatistic(1);
```

```
int DrawCalls=BT_GetStatistic(2);
```

```
int CullChecks=BT_GetStatistic(3);
```

GDK (OOP):

```
Use BT_Main::GetStatistic(int Code);
```


2.7.04 BT ENABLEAUTORENDER

This command enables automatic rendering. When this command is called, you do not have to call any updating or rendering functions inside your loop.

Parameters and returns:

Enabled(bool) – Set this to 1 to enable auto render, 0 to disable auto render.
Returns: Nothing

Syntax:

DBPro	BT EnableAutoRender 1
GDK	BT_EnableAutoRender(1);
GDK (OOP)	Blitzwerks Terrain->EnableAutoRender(1);

3 Examples
(WIP)

4 Internals
(WIP)

5 Extras

5.1 Blend modes

Name	Number	Name	Number
DISABLE	1	BLENDFACTORALPHA	14
SELECTARG1	2	BLENDTEXTUREALPHAPM	15
SELECTARG2	3	BLENDCURRENTALPHA	16
MODULATE	4	PREMODULATE	17
MODULATE2X	5	MODULATEALPHA_ADDCOLOR	18
MODULATE4X	6	MODULATECOLOR_ADDALPHA	19
ADD	7	MODULATEINVALPHA_ADDCOLOR	20
ADDSIGNED	8	MODULATEINVCOLOR_ADDALPHA	21
ADDSIGNED2X	9	BUMPENVMAP	22
SUBTRACT	10	BUMPENVMAPLUMINANCE	23
ADDSMOOTH	11	DOTPRODUCT3	24
BLENDDIFFUSEALPHA	12	MULTIPLYADD	25
BLENDTEXTUREALPHA	13	LERP	26