Man in The Mountain

AUTHOR
Version
Mon Jun 4 2018

Table of Contents

Table of contents

Namespace Index

Packages

Here are the packages with brief descriptions (if available):

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Namespace Documentation

CameraMovement Namespace Reference

Classes

- class **CameraMover**
  *This class deals with the movement of the camera in the unity engine. It uses the **RailSystem** which contains the references to the section of the Rail object that the in-engine camera will move.*
- class **RailSystem**
  *Rail System Script is how the **CameraMover** script interacts with the railNodes. The **CameraMover** will do the calculations to find out how far it should move the camera and where to do it here it give it where to go and how to get there.*

Enumerations

- enum **Mode** { **Mode.Linear**, **Mode.Catmull**, **Mode.Insta** }

Enumeration Type Documentation

enum CameraMovement.Mode[strong]

Enumerator:

Linear	
Catmull	
Insta	

Definition at line 9 of file RailSystem.cs.

TerrainGenData Namespace Reference

Class **`DataForTerrain`** Calculates min and max height values for terrain generation. AnimationCurve

Classes

- class **DataForTerrain**
- class **NoiseData**
- class **TextureData**
- class **UpdatableData**

Detailed Description

Class **`DataForTerrain`** Calculates min and max height values for terrain generation. AnimationCurve

Class **`UpdatableData`** handles automatic updates for terrain generation data.

Class **`TextureData`** handles data to generate and apply texture onto terrain. TextureFormat

Class **`NoiseData`** handles data from **NoiseGenerator.NormalizeMode**

Inherits from **UpdatableData**

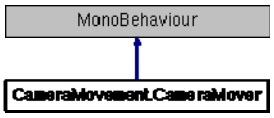
Inherits from ScriptableObject

Class Documentation

CameraMovement.CameraMover Class Reference

This class deals with the movement of the camera in the unity engine. It uses the **RailSystem** which contains the references to the section of the Rail object that the in-engine camera will move.

Inheritance diagram for CameraMovement.CameraMover:



Public Attributes

- **RailSystem rails**
- **Mode playMode**
- float **speed** = 2.5f
- bool **reversed**
- bool **looping**
- bool **pingpong**

Detailed Description

This class deals with the movement of the camera in the unity engine. It uses the **RailSystem** which contains the references to the section of the Rail object that the in-engine camera will move.

The Rail System must have a **RailSystem** Script to run the operations. The playMode which dictates how the camera will move when project is playing but one must be selected to run.

Definition at line 17 of file CameraMover.cs.

Member Data Documentation

bool CameraMovement.CameraMover.looping

Definition at line 25 of file CameraMover.cs.

bool CameraMovement.CameraMover.pingpong

Definition at line 26 of file CameraMover.cs.

Mode CameraMovement.CameraMover.playMode

Definition at line 21 of file CameraMover.cs.

RailSystem CameraMovement.CameraMover.rails

Definition at line 20 of file CameraMover.cs.

bool CameraMovement.CameraMover.reversed

Definition at line 24 of file CameraMover.cs.

float CameraMovement.CameraMover.speed = 2.5f

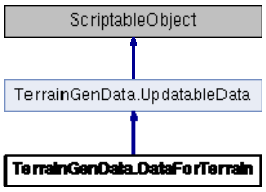
Definition at line 23 of file CameraMover.cs.

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

- **CameraMover.cs**

TerrainGenData.DataForTerrain Class Reference

Inheritance diagram for TerrainGenData.DataForTerrain:



Public Attributes

- float **uniformScale** = 2.5f
- float **meshHeightMultiplier**
- AnimationCurve **meshHeightCurve**

Properties

- float **minHeight** `[get]`
- float **maxHeight** `[get]`

Additional Inherited Members

Detailed Description

Definition at line 14 of file DataForTerrain.cs.

Member Data Documentation

AnimationCurve TerrainGenData.DataForTerrain.meshHeightCurve

Definition at line 19 of file DataForTerrain.cs.

float TerrainGenData.DataForTerrain.meshHeightMultiplier

Definition at line 18 of file DataForTerrain.cs.

float TerrainGenData.DataForTerrain.uniformScale = 2.5f

Definition at line 16 of file DataForTerrain.cs.

Property Documentation

float TerrainGenData.DataForTerrain.maxHeight`[get]`

Definition at line 29 of file DataForTerrain.cs.

float TerrainGenData.DataForTerrain.minHeight`[get]`

Definition at line 22 of file DataForTerrain.cs.

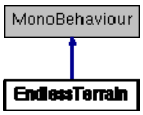
The documentation for this class was generated from the following file:

- **DataForTerrain.cs**

EndlessTerrain Class Reference

Handles the creation and update of all the meshes by delegating to a collection of **TerrainChunk**.

Inheritance diagram for EndlessTerrain:



Classes

- class **TerrainChunk**
  This class handles the dynamic creation and update of an individual mesh.

Public Attributes

- Material **mapMaterial**

Detailed Description

Handles the creation and update of all the meshes by delegating to a collection of **TerrainChunk**.

Code modified from Sebastian Lague's video, Procedural Landmass Generation. This class acts similar to a driver for the classes of the program that are involved in generating the terrain. It has references to and calls updates to other classes.

Definition at line 12 of file EndlessTerrain.cs.

Member Data Documentation

Material EndlessTerrain.mapMaterial

Definition at line 15 of file EndlessTerrain.cs.

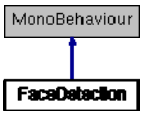
The documentation for this class was generated from the following file:

- **EndlessTerrain.cs**

FaceDetection Class Reference

This class is used to detect and outline faces in the webcam image. It also equalizes its histograms and (optionally) denoises the image.

Inheritance diagram for FaceDetection:



Public Member Functions

- void **UpdateFaceTexture** ()
  Detects and outlines faces in the WebcamTexture and stored as a Mat. Optionally equalizes and denoises WebcamTexture.

Public Attributes

- bool **EqualizeTexture**
- bool **DenoiseTexture**

Properties

- Texture2D **FaceTexture** `[get]`

Detailed Description

This class is used to detect and outline faces in the webcam image. It also equalizes its histograms and (optionally) denoises the image.

This is the only part of the project that uses OpenCVForUnity. Code is modified from OpenCVForUnity example class FaceDetectionWebCamTextureExample.cs.

Definition at line 11 of file FaceDetection.cs.

Member Function Documentation

void FaceDetection.UpdateFaceTexture ()

Detects and outlines faces in the WebcamTexture and stored as a Mat. Optionally equalizes and denoises WebcamTexture.

Definition at line 82 of file FaceDetection.cs.

Member Data Documentation

bool FaceDetection.DenoiseTexture

Definition at line 42 of file FaceDetection.cs.

bool FaceDetection.EqualizeTexture

Definition at line 40 of file FaceDetection.cs.

Property Documentation

Texture2D FaceDetection.FaceTexture[get]

Definition at line 21 of file FaceDetection.cs.

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

- **FaceDetection.cs**

TerrainGenData.TextureData.Layer Class Reference

Public Attributes

- Texture2D **texture**
- Color **tint**
- float **tintStrength**
- float **startHeight**
- float **blendStrength**
- float **textureScale**

Detailed Description

Definition at line 75 of file TextureData.cs.

Member Data Documentation

float TerrainGenData.TextureData.Layer.blendStrength

Definition at line 84 of file TextureData.cs.

float TerrainGenData.TextureData.Layer.startHeight

Definition at line 82 of file TextureData.cs.

Texture2D TerrainGenData.TextureData.Layer.texture

Definition at line 77 of file TextureData.cs.

float TerrainGenData.TextureData.Layer.textureScale

Definition at line 85 of file TextureData.cs.

Color TerrainGenData.TextureData.Layer.tint

Definition at line 78 of file TextureData.cs.

float TerrainGenData.TextureData.Layer.tintStrength

Definition at line 80 of file TextureData.cs.

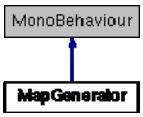
The documentation for this class was generated from the following file:

- **TextureData.cs**

MapGenerator Class Reference

This class is used to generate various "map" objects that are needed for processing images. Key features include the various processes used to validate mapData objects, the calls to the other generators for combining maps together and then the Start method which begins the various initializers for the different objects that combine to form a map. This map will then be passed to the **MeshGenerator** class to be converted into a 3D mesh

Inheritance diagram for MapGenerator:



Public Member Functions

- **MeshData RequestMeshData** (Vector2 chunkPosition)
  Requests the mesh data. Note that this sets the local heightMap object to a generated float[,] based on the current chunkPosition.
- void **OnTextureValuesUpdated** ()
  Called when [texture values updated].
- void **UpdateWaterHeight** ()
  Updates the height of the water.
- void **UpdateFullNoiseMap** ()
  Updates the full noise map.

Public Attributes

- **DataForTerrain terrainData**
  The terrain data
- **NoiseData noiseData**
  The noise data
- **TextureData textureData**
  The texture data
- Material **terrainMaterial**
  The terrain material
- GameObject **Water**
  The water
- float **minGreyValue**
  The minimum grey value used in assigning data to a given greyscale image This is possibly a legacy implementation but its use is to be assigned a float between 0 and 1 in which any given pixel in the greyscale image will be ignored if its calculated value for the height map is below that number. This allows for a "minimum" brightness level to be designated for generating a mesh with
- float **noiseWeight**
  The noise weight
- int **levelOfDetail**
  The level of detail

Static Public Attributes

- static **WebcamTextureController webcamController**
  The webcam controller

Properties

- int **MapChunkWidth** `[get]`

Gets the width of the map chunk.

- int **MapChunkHeight** `[get]`
  Gets the height of the map chunk.
- int **NumChunkWidth** `[get]`
  Gets the width of the number chunk.
- int **NumChunkHeight** `[get]`
  Gets the height of the number chunk.

Detailed Description

This class is used to generate various "map" objects that are needed for processing images. Key features include the various processes used to validate mapData objects, the calls to the other generators for combining maps together and then the Start method which begins the various initializers for the different objects that combine to form a map. This map will then be passed to the **MeshGenerator** class to be converted into a 3D mesh

Definition at line 12 of file MapGenerator.cs.

Member Function Documentation

void MapGenerator.OnTextureValuesUpdated ()

Called when [texture values updated].

Definition at line 288 of file MapGenerator.cs.

MeshData MapGenerator.RequestMeshData (Vector2 *chunkPosition*)

Requests the mesh data. Note that this sets the local heightMap object to a generated float[,] based on the current chunkPosition.

Parameters:

chunkPosition	The chunk position.

Returns:

MeshData object created via a combination of the heightmap, the meshHeightMultiplier, sample points from the meshHeightCurve and a specified level of detail

Definition at line 248 of file MapGenerator.cs.

void MapGenerator.UpdateFullNoiseMap ()

Updates the full noise map.

Definition at line 309 of file MapGenerator.cs.

void MapGenerator.UpdateWaterHeight ()

Updates the height of the water.

Definition at line 297 of file MapGenerator.cs.

Member Data Documentation

int MapGenerator.levelOfDetail

The level of detail

Definition at line 71 of file MapGenerator.cs.

float MapGenerator.minGreyValue

The minimum grey value used in assigning data to a given greyscale image This is possibly a legacy implementation but its use is to be assigned a float between 0 and 1 in which any given pixel in the greyscale image will be ignored if its calculated value for the height map is below that number. This allows for a "minimum" brightness level to be designated for generating a mesh with

Definition at line 60 of file MapGenerator.cs.

NoiseData MapGenerator.noiseData

The noise data

Definition at line 24 of file MapGenerator.cs.

float MapGenerator.noiseWeight

The noise weight

Definition at line 65 of file MapGenerator.cs.

DataForTerrain MapGenerator.terrainData

The terrain data

Definition at line 19 of file MapGenerator.cs.

Material MapGenerator.terrainMaterial

The terrain material

Definition at line 34 of file MapGenerator.cs.

TextureData MapGenerator.textureData

The texture data

Definition at line 29 of file MapGenerator.cs.

GameObject MapGenerator.Water

The water

Definition at line 49 of file MapGenerator.cs.

WebcamTextureController MapGenerator.webcamController[static]

The webcam controller

Definition at line 39 of file MapGenerator.cs.

Property Documentation

int MapGenerator.MapChunkHeight[get]

Gets the height of the map chunk.

The height of the map chunk.

Definition at line 126 of file MapGenerator.cs.

int MapGenerator.MapChunkWidth[get]

Gets the width of the map chunk.

The width of the map chunk.

Definition at line 112 of file MapGenerator.cs.

int MapGenerator.NumChunkHeight[get]

Gets the height of the number chunk.

The height of the number chunk.

Definition at line 154 of file MapGenerator.cs.

int MapGenerator.NumChunkWidth[get]

Gets the width of the number chunk.

The width of the number chunk.

Definition at line 140 of file MapGenerator.cs.

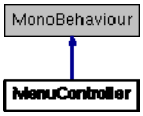
The documentation for this class was generated from the following file:

- **MapGenerator.cs**

MenuController Class Reference

Class `MenuConrtoller` provides structured layout for a user friendly interface.

Inheritance diagram for MenuController:



Public Member Functions

- void **OnPlayButtonClick** ()
  Method `OnPlayButtonClick` *calls Play() when play button is clicked.*
- void **OnPauseButtonClick** ()
  Method `OnPauseButtonClick` *calls Pause() when pause button is clicked.*
- void **OnChangeCameraButtonClick** ()
  Method `OnChangeCameraButtonClick` *calls ChangeWebcamTextureToNextAvailable() when change camera button is clicked.*
- void **SetUniformScale** (float vin)
  Method `SetUniformScale` *sets the uniform scale for terrain, and updates height.*
- void **SetHeightMultiplier** (float vin)
  Method `SetHeightMultiplier` *takes a float and sets the height mutliplier for terrain, then updates height.*
- void **SetNoiseBlending** (float vin)
  Method `SetNoiseBlending` *sets noise blending value.*
- void **SetNoiseScale** (float vin)
  Method `SetNoiseScale` *takes a value of type float and sets it as new noise scale.* ***noiseData***
- void **SetNoiseOctaves** (float vin)
  Method `SetNoiseOctaves` *takes a value of type float and sets it as the new octave value rounded to an int.* ***noiseData***
- void **SetNoisePersistance** (float vin)
  Method `SetNoisePersistance` *takes a value of type float and sets it as the new persistance value.* ***noiseData***
- void **SetNoiseLacunarity** (float vin)
  Method `SetNoiseLacunarity` *takes a value of type float and sets it as the new lacunarity value.* ***noiseData***
- void **SetTextureTint** (Color color, int index)
  Method `SetTextureTint` *takes a value of type float and sets it as the new texture tint value.*
- void **StringToTexture0** (String text)
  Method `StringToTexture0` *takes a value of type String and sets it as texture 0.*
- void **StringToTexture1** (String text)
  Method `StringToTexture1` *takes a value of type String and sets it as texture 1.*
- void **StringToTexture2** (String text)
  Method `StringToTexture2` *takes a value of type String and sets it as texture 2.*
- void **StringToTexture3** (String text)
  Method `StringToTexture3` *takes a value of type String and sets it as texture 3.*
- void **StringToTexture4** (String text)
  Method `StringToTexture4` *takes a value of type String and sets it as texture 4.*
- void **StringToTexture5** (String text)
  Method `StringToTexture5` *takes a value of type String and sets it as texture 5.*

- void **SetTextureHeight0** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 0.*
- void **SetTextureHeight1** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 1.*
- void **SetTextureHeight2** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 2*
- void **SetTextureHeight3** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 3.*
- void **SetTextureHeight4** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 4.*
- void **SetTextureHeight5** (float value)
  Method `SetTextureHeight0` *takes a value of type float and sets it as a height for texture 5.*
- void **SetCameraSpeed** (float vin)
  Method `SetCameraSpeed` *takes a value of type float and sets it as the new camera speed.*

Public Attributes

- **DataForTerrain terrainData**
- **NoiseData noiseData**
- **TextureData textureData**
- **MapGenerator mapGen**
- **CameraMover cameraMover**

Detailed Description

Class `MenuConrtoller` provides structured layout for a user friendly interface.

Definition at line 10 of file MenuController.cs.

Member Function Documentation

void MenuController.OnChangeCameraButtonClick ()

Method `OnChangeCameraButtonClick` calls ChangeWebcamTextureToNextAvailable() when change camera button is clicked.

Definition at line 48 of file MenuController.cs.

void MenuController.OnPauseButtonClick ()

Method `OnPauseButtonClick` calls Pause() when pause button is clicked.

Definition at line 40 of file MenuController.cs.

void MenuController.OnPlayButtonClick ()

Method `OnPlayButtonClick` calls Play() when play button is clicked.

Definition at line 32 of file MenuController.cs.

void MenuController.SetCameraSpeed (float *vin*)

Method `SetCameraSpeed` takes a value of type float and sets it as the new camera speed.

Definition at line 281 of file MenuController.cs.

void MenuController.SetHeightMultiplier (float *vin*)

Method `SetHeightMultiplier` takes a float and sets the height mutliplier for terrain, then updates height.

Parameters:

vin	new height multiplier value of type float

Definition at line 73 of file MenuController.cs.

void MenuController.SetNoiseBlending (float *vin*)

Method `SetNoiseBlending` sets noise blending value.

Parameters:

vin	

Definition at line 85 of file MenuController.cs.

void MenuController.SetNoiseLacunarity (float *vin*)

Method `SetNoiseLacunarity` takes a value of type float and sets it as the new lacunarity value. **noiseData**

Parameters:

vin	new lacunarity value of type float

Definition at line 130 of file MenuController.cs.

void MenuController.SetNoiseOctaves (float *vin*)

Method `SetNoiseOctaves` takes a value of type float and sets it as the new octave value rounded to an int. **noiseData**

Parameters:

vin	new octave value of type float

Definition at line 108 of file MenuController.cs.

void MenuController.SetNoisePersistance (float *vin*)

Method `SetNoisePersistance` takes a value of type float and sets it as the new persistance value. **noiseData**

Parameters:

vin	new persistance value of type float

Definition at line 119 of file MenuController.cs.

void MenuController.SetNoiseScale (float *vin*)

Method `SetNoiseScale` takes a value of type float and sets it as new noise scale. **noiseData**

Parameters:

vin	new noise scale value of type float

Definition at line 97 of file MenuController.cs.

void MenuController.SetTextureHeight0 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 0.

Parameters:

value	new textuure height value of type float

Definition at line 218 of file MenuController.cs.

void MenuController.SetTextureHeight1 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 1.

Parameters:

value	new textuure height value of type float

Definition at line 228 of file MenuController.cs.

void MenuController.SetTextureHeight2 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 2

Parameters:

value	new textuure height value of type float

Definition at line 238 of file MenuController.cs.

void MenuController.SetTextureHeight3 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 3.

Parameters:

value	new textuure height value of type float

Definition at line 248 of file MenuController.cs.

void MenuController.SetTextureHeight4 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 4.

Parameters:

value	new textuure height value of type float

Definition at line 258 of file MenuController.cs.

void MenuController.SetTextureHeight5 (float *value*)

Method `SetTextureHeight0` takes a value of type float and sets it as a height for texture 5.

Parameters:

value	new textuure height value of type float

Definition at line 268 of file MenuController.cs.

void MenuController.SetTextureTint (Color *color*, int *index*)

Method `SetTextureTint` takes a value of type float and sets it as the new texture tint value.

Parameters:

color	
index	

Definition at line 143 of file MenuController.cs.

void MenuController.SetUniformScale (float *vin*)

Method `SetUniformScale` sets the uniform scale for terrain, and updates height.

Parameters:

vin	new uniform scale value of type float

Definition at line 62 of file MenuController.cs.

void MenuController.StringToTexture0 (String *text*)

Method `StringToTexture0` takes a value of type String and sets it as texture 0.

Parameters:

text	String value to be parsed into texture

Definition at line 152 of file MenuController.cs.

void MenuController.StringToTexture1 (String *text*)

Method `StringToTexture1` takes a value of type String and sets it as texture 1.

Parameters:

text	String value to be parsed into texture

Definition at line 163 of file MenuController.cs.

void MenuController.StringToTexture2 (String *text*)

Method `StringToTexture2` takes a value of type String and sets it as texture 2.

Parameters:

text	String value to be parsed into texture

Definition at line 174 of file MenuController.cs.

void MenuController.StringToTexture3 (String *text*)

Method `StringToTexture3` takes a value of type String and sets it as texture 3.

Parameters:

text	String value to be parsed into texture

Definition at line 185 of file MenuController.cs.

void MenuController.StringToTexture4 (String *text*)

Method `StringToTexture4` takes a value of type String and sets it as texture 4.

Parameters:

text	String value to be parsed into texture

Definition at line 196 of file MenuController.cs.

void MenuController.StringToTexture5 (String *text*)

Method `StringToTexture5` takes a value of type String and sets it as texture 5.

Parameters:

text	String value to be parsed into texture

Definition at line 207 of file MenuController.cs.

Member Data Documentation

CameraMover MenuController.cameraMover

Definition at line 17 of file MenuController.cs.

MapGenerator MenuController.mapGen

Definition at line 16 of file MenuController.cs.

NoiseData MenuController.noiseData

Definition at line 14 of file MenuController.cs.

DataForTerrain MenuController.terrainData

Definition at line 13 of file MenuController.cs.

TextureData MenuController.textureData

Definition at line 15 of file MenuController.cs.

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

- **MenuController.cs**

MeshData Class Reference

MeshData is the means by which individual meshes are constructed and then stored. Each **MeshData** class contains three key data arrays: vertices, triangles and uvs. Vertices is a collection of Vector3 objects that represent points in 3D space within the Unity engine Triangles is a collection of "triangles" that are formed via the connection of the points stored within the vertices array with points closest to one another connecting to form triangles. Note, the overall mesh is contiguous but it is built via the connection and "sewing" together of triangles UVs is the uv data for each triangle used to map textures onto the mesh through the triangles stored within the **MeshData**

Public Member Functions

- **MeshData** (int meshWidth, int meshHeight)
  *Initializes a new instance of the **MeshData** class.*
- void **AddTriangle** (int a, int b, int c)
  Adds the triangle.
- Mesh **CreateMesh** ()
  Creates the mesh.
- void **UpdateMesh** (Mesh mesh)
  Updates the mesh.

Public Attributes

- Vector3 [] **vertices**
  The vertices
- int [] **triangles**
  The triangles
- Vector2 [] **uvs**
  The uvs

Detailed Description

MeshData is the means by which individual meshes are constructed and then stored. Each **MeshData** class contains three key data arrays: vertices, triangles and uvs. Vertices is a collection of Vector3 objects that represent points in 3D space within the Unity engine Triangles is a collection of "triangles" that are formed via the connection of the points stored within the vertices array with points closest to one another connecting to form triangles. Note, the overall mesh is contiguous but it is built via the connection and "sewing" together of triangles UVs is the uv data for each triangle used to map textures onto the mesh through the triangles stored within the **MeshData**

Definition at line 64 of file MeshGenerator.cs.

Constructor & Destructor Documentation

MeshData.MeshData (int *meshWidth*, int *meshHeight*)

Initializes a new instance of the **MeshData** class.

Parameters:

meshWidth	Width of the mesh.
meshHeight	Height of the mesh.

Definition at line 91 of file MeshGenerator.cs.

Member Function Documentation

void MeshData.AddTriangle (int *a*, int *b*, int *c*)

Adds the triangle.

Parameters:

a	side a.
b	side b.
c	side c.

Definition at line 104 of file MeshGenerator.cs.

Mesh MeshData.CreateMesh ()

Creates the mesh.

Returns:

Definition at line 119 of file MeshGenerator.cs.

void MeshData.UpdateMesh (Mesh *mesh*)

Updates the mesh.

Parameters:

mesh	The mesh to be updated.

Definition at line 133 of file MeshGenerator.cs.

Member Data Documentation

int [] MeshData.triangles

The triangles

Definition at line 74 of file MeshGenerator.cs.

Vector2 [] MeshData.uvs

The uvs

Definition at line 79 of file MeshGenerator.cs.

Vector3 [] MeshData.vertices

The vertices

Definition at line 69 of file MeshGenerator.cs.

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

- **MeshGenerator.cs**

MeshGenerator Class Reference

MeshGenerator is the class responsible for handling the calculations required to create the 3D object from a calculated heightmap. The key method is GenerateTerrainMesh in which a heightMap representing a greyscale image converted into a float[,] is iterated through and sampled at regular intervals in order to find a value for which to create a point in 3D space. These points are then combined into triangles within a **MeshData** object and these **MeshData** objects are then combined to form the overall mesh

Static Public Member Functions

- static **MeshData GenerateTerrainMesh** (float[,] heightMap, float heightMultiplier, AnimationCurve heightCurve, int levelOfDetail)

  Generates the terrain mesh. The key process of this method is the iteration through the heightMap and the regular sampling that takes place. The heightMap is iterated for every combination point x,y at an interval equal to the meshSimplificationIncremement. This means that as the level of detail decreases, the granularity and resolution of the sampling will decrease and the generated mesh will contain less detail/vertices.

Detailed Description

MeshGenerator is the class responsible for handling the calculations required to create the 3D object from a calculated heightmap. The key method is GenerateTerrainMesh in which a heightMap representing a greyscale image converted into a float[,] is iterated through and sampled at regular intervals in order to find a value for which to create a point in 3D space. These points are then combined into triangles within a **MeshData** object and these **MeshData** objects are then combined to form the overall mesh

Definition at line 9 of file MeshGenerator.cs.

Member Function Documentation

static MeshData MeshGenerator.GenerateTerrainMesh (float *heightMap*[,], float *heightMultiplier*, AnimationCurve *heightCurve*, int *levelOfDetail*) [static]

Generates the terrain mesh. The key process of this method is the iteration through the heightMap and the regular sampling that takes place. The heightMap is iterated for every combination point x,y at an interval equal to the meshSimplificationIncremement. This means that as the level of detail decreases, the granularity and resolution of the sampling will decrease and the generated mesh will contain less detail/vertices.

Parameters:

heightMap	The height map value.
heightMultiplier	The height multiplier value.
heightCurve	The height curve value.
levelOfDetail	The level of detail value.

Returns:

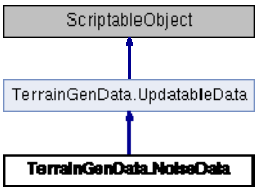
Definition at line 21 of file MeshGenerator.cs.

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

- **MeshGenerator.cs**

TerrainGenData.NoiseData Class Reference

Inheritance diagram for TerrainGenData.NoiseData:



Public Attributes

- **NoiseGenerator.NormalizeMode normalizeMode**
- float **noiseScale**
- int **octaves**
  Octave: One of the coherent-noise functions in a series of coherent-noise functions that are added together to form Perlin noise.
- float **persistance**
  Persistance: A multiplier that determines how quickly the amplitudes diminish for each successive octave in a Perlin-noise function.
- float **lacunarity**
  Lacunarity: A multiplier that determines how quickly the frequency increases for each successive octave in a Perlin-noise function.
- int **seed**
- Vector2 **offset**

Protected Member Functions

- override void **OnValidate** ()
  *Method `OnValidate` checks values of lacunarity and octaves, then updates then and calls super **OnValidate()**.*

Properties

- bool **Updated** `[get, set]`

Additional Inherited Members

Detailed Description

Definition at line 11 of file NoiseData.cs.

Member Function Documentation

override void TerrainGenData.NoiseData.OnValidate ()[protected], [virtual]

Method `OnValidate` checks values of lacunarity and octaves, then updates then and calls super **OnValidate()**.

Reimplemented from **TerrainGenData.UpdatableData** (*p.44*).

Definition at line 41 of file NoiseData.cs.

Member Data Documentation

float TerrainGenData.NoiseData.lacunarity

Lacunarity: A multiplier that determines how quickly the frequency increases for each successive octave in a Perlin-noise function.

Definition at line 31 of file NoiseData.cs.

float TerrainGenData.NoiseData.noiseScale

Definition at line 15 of file NoiseData.cs.

NoiseGenerator.NormalizeMode TerrainGenData.NoiseData.normalizeMode

Definition at line 13 of file NoiseData.cs.

int TerrainGenData.NoiseData.octaves

Octave: One of the coherent-noise functions in a series of coherent-noise functions that are added together to form Perlin noise.

Definition at line 21 of file NoiseData.cs.

Vector2 TerrainGenData.NoiseData.offset

Definition at line 34 of file NoiseData.cs.

float TerrainGenData.NoiseData.persistance

Persistance: A multiplier that determines how quickly the amplitudes diminish for each successive octave in a Perlin-noise function.

Definition at line 27 of file NoiseData.cs.

int TerrainGenData.NoiseData.seed

Definition at line 33 of file NoiseData.cs.

Property Documentation

bool TerrainGenData.NoiseData.Updated[get], [set]

Definition at line 36 of file NoiseData.cs.

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

- **NoiseData.cs**

NoiseGenerator Class Reference

This class is responsible for the calculations needed to generate a float[,] representing a perlin noise map. It is additionally responsible for blending a `Texture2D` object (that in most cases represents a webcam image or some still image) with a noiseMap before then returning it as a float[,]. In both cases, the float[,] represents a heightMap containing values at each point that represent the height of a given x,y coordinate. This is then used to determine how "tall" that point should be in the 3D Unity worldspace as part of the mesh.

Public Types

- enum **NormalizeMode** { **NormalizeMode.Local**, **NormalizeMode.Global** }

Static Public Member Functions

- static float [,] **GenerateNoiseMap** (int mapWidth, int mapHeight, int seed, float scale, int octaves, float persistance, float lacunarity, Vector2 offset, **NormalizeMode** normalizeMode)

  Generates the noise map based upon perlin noise. This creates a smooth transition between minimum and maximum value boundaries and allows for a more realistic overall look.
- static float [,] **LerpNoiseMapWithTextureToNoiseChunk** (Texture2D texture, float[,] noiseMap, float noiseWeight, float minGreyValue, int chunkWidth, int chunkHeight, Vector2 offset)

  Lerps the noise map with texture to noise chunk. This method is used to blend a texture with a generated noiseMap linearly in order to provide a controllable and uniform semi-randomization to the texture. This method combines a noiseMap with a Texture2D by sampling each x,y point and taking the linearly interpolated value generated based upon a weight and then places that new value into a new float[,] which is then returned once all points have been sampled

---

Detailed Description

This class is responsible for the calculations needed to generate a float[,] representing a perlin noise map. It is additionally responsible for blending a `Texture2D` object (that in most cases represents a webcam image or some still image) with a noiseMap before then returning it as a float[,]. In both cases, the float[,] represents a heightMap containing values at each point that represent the height of a given x,y coordinate. This is then used to determine how "tall" that point should be in the 3D Unity worldspace as part of the mesh.

Definition at line 13 of file NoiseGenerator.cs.

Member Enumeration Documentation

enum NoiseGenerator.NormalizeMode[strong]

Enumerator:

Local	
Global	

Definition at line 18 of file NoiseGenerator.cs.

Member Function Documentation

static float [,] NoiseGenerator.GenerateNoiseMap (int *mapWidth*, int *mapHeight*, int *seed*, float *scale*, int *octaves*, float *persistance*, float *lacunarity*, Vector2 *offset*, NormalizeMode *normalizeMode*)[static]

Generates the noise map based upon perlin noise. This creates a smooth transition between minimum and maximum value boundaries and allows for a more realistic overall look.

Parameters:

mapWidth	Width of the map.
mapHeight	Height of the map. This is the "y" component of a 2D map, not the actual height of how "tall" something on the 3D mesh would be
seed	The seed used for the pseudo-random number generator
scale	The scale. This can be thought of "zooming" in and out of sections of the noise map
octaves	The octaves. This controls the large troughs and peaks on the noise map. This is akin to mountains and valleys
persistance	The persistance. This controls the medium level details on the noiseMap and can be thought of as ridges, hills or crevices
lacunarity	The lacunarity. This controls the small level details on the noiseMap and can be thought of as individual rocks/boulders or other small details
offset	The offset.
normalizeMode	The normalize mode.

Returns:

A float[,] representing a NoiseMap generated via the method. In particular, values within the float[,] will range from 0 to a value controlled by the maxPossibleHeight variable

Definition at line 34 of file NoiseGenerator.cs.

static float [,] NoiseGenerator.LerpNoiseMapWithTextureToNoiseChunk (Texture2D *texture*, float *noiseMap*[,], float *noiseWeight*, float *minGreyValue*, int *chunkWidth*, int *chunkHeight*, Vector2 *offset*)[static]

Lerps the noise map with texture to noise chunk. This method is used to blend a texture with a generated noiseMap linearly in order to provide a controllable and uniform semi-randomization to the texture. This method combines a noiseMap with a Texture2D by sampling each x,y point and taking the linearly interpolated value generated based upon a weight and then places that new value into a new float[,] which is then returned once all points have been sampled

Parameters:

texture	The texture.
noiseMap	The noise map.
noiseWeight	The noise weight.
minGreyValue	The minimum grey value.
chunkWidth	Width of the chunk.
chunkHeight	Height of the chunk.
offset	The offset.

Returns:

A float[,] representing a heightMap that has been generated via the combination of the Texture2D and the noiseMap after the two have been blended together

Definition at line 131 of file NoiseGenerator.cs.

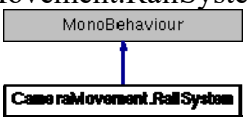
The documentation for this class was generated from the following file:

- **NoiseGenerator.cs**

CameraMovement.RailSystem Class Reference

Rail System Script is how the **CameraMover** script interacts with the railNodes. The **CameraMover** will do the calculations to find out how far it should move the camera and where to do it here it give it where to go and how to get there.

Inheritance diagram for CameraMovement.RailSystem:



Public Member Functions

- void **setNodes** ()
  *Will get the the current list of positions from GameObjects used for **RailSystem**.*
- Vector3 **PositionOnRailSystem** (int seg, float ratio, **Mode** playMode)
  Dictates which type of movement the camera will do. Once determined it will pass the ratio and current segment.
- Quaternion **Orientation** (int seg, float ratio)
  This class deals with the orientation of the camera as it moves to the next point.
- Transform [] **getRailNodes** ()
  This is to get access of the List of railNode positions.

Detailed Description

Rail System Script is how the **CameraMover** script interacts with the railNodes. The **CameraMover** will do the calculations to find out how far it should move the camera and where to do it here it give it where to go and how to get there.

This Script must be attacted to the GameObjects to be used as a Rail System. railNodes: List of positions from GameObject, which is known as Rails in the editor.

Definition at line 25 of file RailSystem.cs.

Member Function Documentation

Transform [] CameraMovement.RailSystem.getRailNodes ()

This is to get access of the List of railNode positions.

Returns:

List of railNode positions

Definition at line 99 of file RailSystem.cs.

Quaternion CameraMovement.RailSystem.Orientation (int *seg*, float *ratio*)

This class deals with the orientation of the camera as it moves to the next point.

Parameters:

seg	Current segment the camera is coming from

ratio	Current ratio it is to the next segment

Returns:

The orientation of the camera according to its next position

Definition at line 88 of file RailSystem.cs.

Vector3 CameraMovement.RailSystem.PositionOnRailSystem (int *seg*, float *ratio*, Mode *playMode*)

Dictates which type of movement the camera will do. Once determined it will pass the ratio and current segment.

Parameters:

seg	Is the list segment the currently on in the railNode list
ratio	
playMode	Tells what Type of movement the camera will do

Returns:

This returns the position the camera from the given playMode

Definition at line 58 of file RailSystem.cs.

void CameraMovement.RailSystem.setNodes ()

Will get the the current list of positions from GameObjects used for **RailSystem**.

Definition at line 41 of file RailSystem.cs.

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

- **RailSystem.cs**

EndlessTerrain.TerrainChunk Class Reference

This class handles the dynamic creation and update of an individual mesh.

Public Member Functions

- **TerrainChunk** (Vector2 coord, int width, int height, Transform parent, Material material)
- void **UpdateTerrainChunk** ()
  *Gets updated **MeshData** and delegates update of mesh to **MeshData** class.*

Detailed Description

This class handles the dynamic creation and update of an individual mesh.

Definition at line 81 of file EndlessTerrain.cs.

Constructor & Destructor Documentation

EndlessTerrain.TerrainChunk.TerrainChunk (Vector2 *coord*, int *width*, int *height*, Transform *parent*, Material *material*)

Parameters:

coord	Used to determine where chunk should be generated. Uses positive (x,y) coordinates.
width	Width of chunk.
height	Height of chunk.
parent	Transform of class creating **TerrainChunk**.
material	Used by MeshRenderer.

Definition at line 97 of file EndlessTerrain.cs.

Member Function Documentation

void EndlessTerrain.TerrainChunk.UpdateTerrainChunk ()

Gets updated **MeshData** and delegates update of mesh to **MeshData** class.

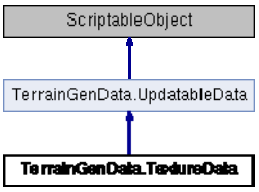
Definition at line 123 of file EndlessTerrain.cs.

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

- **EndlessTerrain.cs**

TerrainGenData.TextureData Class Reference

Inheritance diagram for TerrainGenData.TextureData:



Classes

- class **Layer**

Public Member Functions

- void **ApplyToMaterial** (Material material)
  Method `ApplyToMaterial` *takes a material and applies texture to it.*
- void **UpdateMeshHeights** (Material material, float minHeight, float maxHeight)
  Method `UpdateMeshHeights` *updates the min and max height for mesh.*

Public Attributes

- **Layer [] layers**

Additional Inherited Members

Detailed Description

Definition at line 15 of file TextureData.cs.

Member Function Documentation

void TerrainGenData.TextureData.ApplyToMaterial (Material *material*)

Method `ApplyToMaterial` takes a material and applies texture to it.

Parameters:

material	the material of type Material that we apply texture to

Definition at line 29 of file TextureData.cs.

void TerrainGenData.TextureData.UpdateMeshHeights (Material *material*, float *minHeight*, float *maxHeight*)

Method `UpdateMeshHeights` updates the min and max height for mesh.

Parameters:

material	the material of type Material that we update its height data
minHeight	the min value for height of type float
maxHeight	the max value for height of type float

Definition at line 49 of file TextureData.cs.

Member Data Documentation

Layer [] TerrainGenData.TextureData.layers

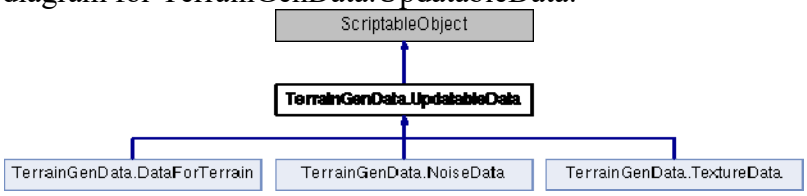
Definition at line 20 of file TextureData.cs.

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

- **TextureData.cs**

TerrainGenData.UpdatableData Class Reference

Inheritance diagram for TerrainGenData.UpdatableData:



Public Member Functions

- void **NotifyOfUpdatedValues** ()
  Method `NotifyOfUpdatedValues` *checks if values updated, calls* ***OnValuesUpdated***

Public Attributes

- bool **autoUpdate**

Protected Member Functions

- virtual void **OnValidate** ()
  Method `OnValidate` *checks auto updates*

Events

- System.Action **OnValuesUpdated**

Detailed Description

Definition at line 12 of file UpdatableData.cs.

Member Function Documentation

void TerrainGenData.UpdatableData.NotifyOfUpdatedValues ()

Method `NotifyOfUpdatedValues` checks if values updated, calls **OnValuesUpdated**

Definition at line 33 of file UpdatableData.cs.

virtual void TerrainGenData.UpdatableData.OnValidate ()[protected], [virtual]

Method `OnValidate` checks auto updates

Reimplemented in **TerrainGenData.NoiseData** (*p.33*).

Definition at line 20 of file UpdatableData.cs.

Member Data Documentation

bool TerrainGenData.UpdatableData.autoUpdate

Definition at line 15 of file UpdatableData.cs.

Event Documentation

System.Action TerrainGenData.UpdatableData.OnValuesUpdated

Definition at line 14 of file UpdatableData.cs.

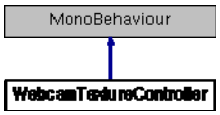
The documentation for this class was generated from the following file:

- **UpdatableData.cs**

WebcamTextureController Class Reference

Handles and stores the WebcamTexture. Also converts WebCamTexture to OpenCV mat.

Inheritance diagram for WebcamTextureController:



Public Member Functions

- void **Initialize** ()
  Should be called before using this class to initialize class properties and start webcam. Class only needs to be initialized once.
- void **ChangeWebcamTextureToNextAvailable** ()
  Will initialize a new WebcamTexture using the next available Webcam device.
- bool **DidUpdateThisFrame** ()
  Checks if the video buffer updated this frame

Public Attributes

- string **requestedDeviceName**
- int **webcamRequestedWidth**
- int **webcamRequestedHeight**

Properties

- int **WebcamHeight** `[get]`
  Height of the texture in pixels.
- int **WebcamWidth** `[get]`
  Width of the texture in pixels.
- WebCamTexture **WebcamTex** `[get]`
  WebcamTexture this class handles.
- Mat **WebcamMat** `[get]`
  OpenCV Mat of WebcamTexture.
- Color32 [] **Colors** `[get]`
  Can be used for OpenCV functions that require Color32[] to reduce garbage collection.

Detailed Description

Handles and stores the WebcamTexture. Also converts WebCamTexture to OpenCV mat.

Class needs to be initialized by another class before it is ready to be used. Use this class to retrieve the WebcamTexture that will be used to help create the program's height map.

Definition at line 14 of file WebcamTextureController.cs.

Member Function Documentation

void WebcamTextureController.ChangeWebcamTextureToNextAvailable ()

Will initialize a new WebcamTexture using the next available Webcam device.

Definition at line 152 of file WebcamTextureController.cs.

bool WebcamTextureController.DidUpdateThisFrame ()

Checks if the video buffer updated this frame

Returns:

True if the video buffer updated this frame; false if not.

Definition at line 206 of file WebcamTextureController.cs.

void WebcamTextureController.Initialize ()

Should be called before using this class to initialize class properties and start webcam. Class only needs to be initialized once.

Definition at line 78 of file WebcamTextureController.cs.

Member Data Documentation

string WebcamTextureController.requestedDeviceName

Definition at line 17 of file WebcamTextureController.cs.

int WebcamTextureController.webcamRequestedHeight

Definition at line 25 of file WebcamTextureController.cs.

int WebcamTextureController.webcamRequestedWidth

Definition at line 23 of file WebcamTextureController.cs.

Property Documentation

Color32 [] WebcamTextureController.Colors[get]

Can be used for OpenCV functions that require Color32[] to reduce garbage collection.

Definition at line 72 of file WebcamTextureController.cs.

int WebcamTextureController.WebcamHeight[get]

Height of the texture in pixels.

Definition at line 28 of file WebcamTextureController.cs.

Mat WebcamTextureController.WebcamMat[get]

OpenCV Mat of WebcamTexture.
Definition at line 61 of file WebcamTextureController.cs.

WebCamTexture WebcamTextureController.WebcamTex[get]

WebcamTexture this class handles.
Definition at line 56 of file WebcamTextureController.cs.

int WebcamTextureController.WebcamWidth[get]

Width of the texture in pixels.
Definition at line 36 of file WebcamTextureController.cs.

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

- **WebcamTextureController.cs**

File Documentation

CameraMover.cs File Reference

Classes

- class **CameraMovement.CameraMover**
  *This class deals with the movement of the camera in the unity engine. It uses the **RailSystem** which contains the references to the section of the Rail object that the in-engine camera will move.*

Namespaces

- namespace **CameraMovement**

DataForTerrain.cs File Reference

Classes

- class **TerrainGenData.DataForTerrain**

Namespaces

- namespace **TerrainGenData**

Class ***DataForTerrain*** *Calculates min and max height values for terrain generation.*
AnimationCurve

EndlessTerrain.cs File Reference

Classes

- class **EndlessTerrain**
  *Handles the creation and update of all the meshes by delegating to a collection of **TerrainChunk**.*
- class **EndlessTerrain.TerrainChunk**
  This class handles the dynamic creation and update of an individual mesh.

FaceDetection.cs File Reference

Classes

- class **FaceDetection**
  This class is used to detect and outline faces in the webcam image. It also equalizes its histograms and (optionally) denoises the image.

MapGenerator.cs File Reference

Classes

- class **MapGenerator**
  *This class is used to generate various "map" objects that are needed for processing images. Key features include the various processes used to validate mapData objects, the calls to the other generators for combining maps together and then the Start method which begins the various initializers for the different objects that combine to form a map. This map will then be passed to the **MeshGenerator** class to be converted into a 3D mesh*

MenuController.cs File Reference

Classes

- class **MenuController**
  Class `MenuConrtoller` *provides structured layout for a user friendly interface.*

MeshGenerator.cs File Reference

Classes

- class **MeshGenerator**
  MeshGenerator *is the class responsible for handling the calculations required to create the 3D object from a calculated heightmap. The key method is GenerateTerrainMesh in which a heightMap representing a greyscale image converted into a float[,] is iterated through and sampled at regular intervals in order to find a value for which to create a point in 3D space. These points are then combined into triangles within a* ***MeshData*** *object and these* ***MeshData*** *objects are then combined to form the overall mesh*
- class **MeshData**
  MeshData *is the means by which individual meshes are constructed and then stored. Each* ***MeshData*** *class contains three key data arrays: vertices, triangles and uvs. Vertices is a collection of Vector3 objects that represent points in 3D space within the Unity engine Triangles is a collection of "triangles" that are formed via the connection of the points stored within the vertices array with points closest to one another connecting to form triangles. Note, the overall mesh is contiguous but it is built via the connection and "sewing" together of triangles UVs is the uv data for each triangle used to map textures onto the mesh through the triangles stored within the* ***MeshData***

NoiseData.cs File Reference

Classes

- class **TerrainGenData.NoiseData**

Namespaces

- namespace **TerrainGenData**

Class ***`DataForTerrain`*** *Calculates min and max height values for terrain generation.*
AnimationCurve

NoiseGenerator.cs File Reference

Classes

- class **NoiseGenerator**
  This class is responsible for the calculations needed to generate a float[,] representing a perlin noise map. It is additionally responsible for blending a `Texture2D` *object (that in most cases represents a webcam image or some still image) with a noiseMap before then returning it as a float[,]. In both cases, the float[,] represents a heightMap containing values at each point that represent the height of a given x,y coordinate. This is then used to determine how "tall" that point should be in the 3D Unity worldspace as part of the mesh.*

RailSystem.cs File Reference

Classes

- class **CameraMovement.RailSystem**
  *Rail System Script is how the **CameraMover** script interacts with the railNodes. The **CameraMover** will do the calculations to find out how far it should move the camera and where to do it here it give it where to go and how to get there.*

Namespaces

- namespace **CameraMovement**

Enumerations

- enum **CameraMovement.Mode** { **CameraMovement.Mode.Linear**, **CameraMovement.Mode.Catmull**, **CameraMovement.Mode.Insta** }

TextureData.cs File Reference

Classes

- class **TerrainGenData.TextureData**
- class **TerrainGenData.TextureData.Layer**

Namespaces

- namespace **TerrainGenData**

Class ***`DataForTerrain`*** *Calculates min and max height values for terrain generation.*
AnimationCurve

UpdatableData.cs File Reference

Classes

- class **TerrainGenData.UpdatableData**

Namespaces

- namespace **TerrainGenData**

Class ***DataForTerrain*** *Calculates min and max height values for terrain generation.*
AnimationCurve

WebcamTextureController.cs File Reference

Classes

- class **WebcamTextureController**
  Handles and stores the WebcamTexture. Also converts WebCamTexture to OpenCV mat.

Index

INDEX