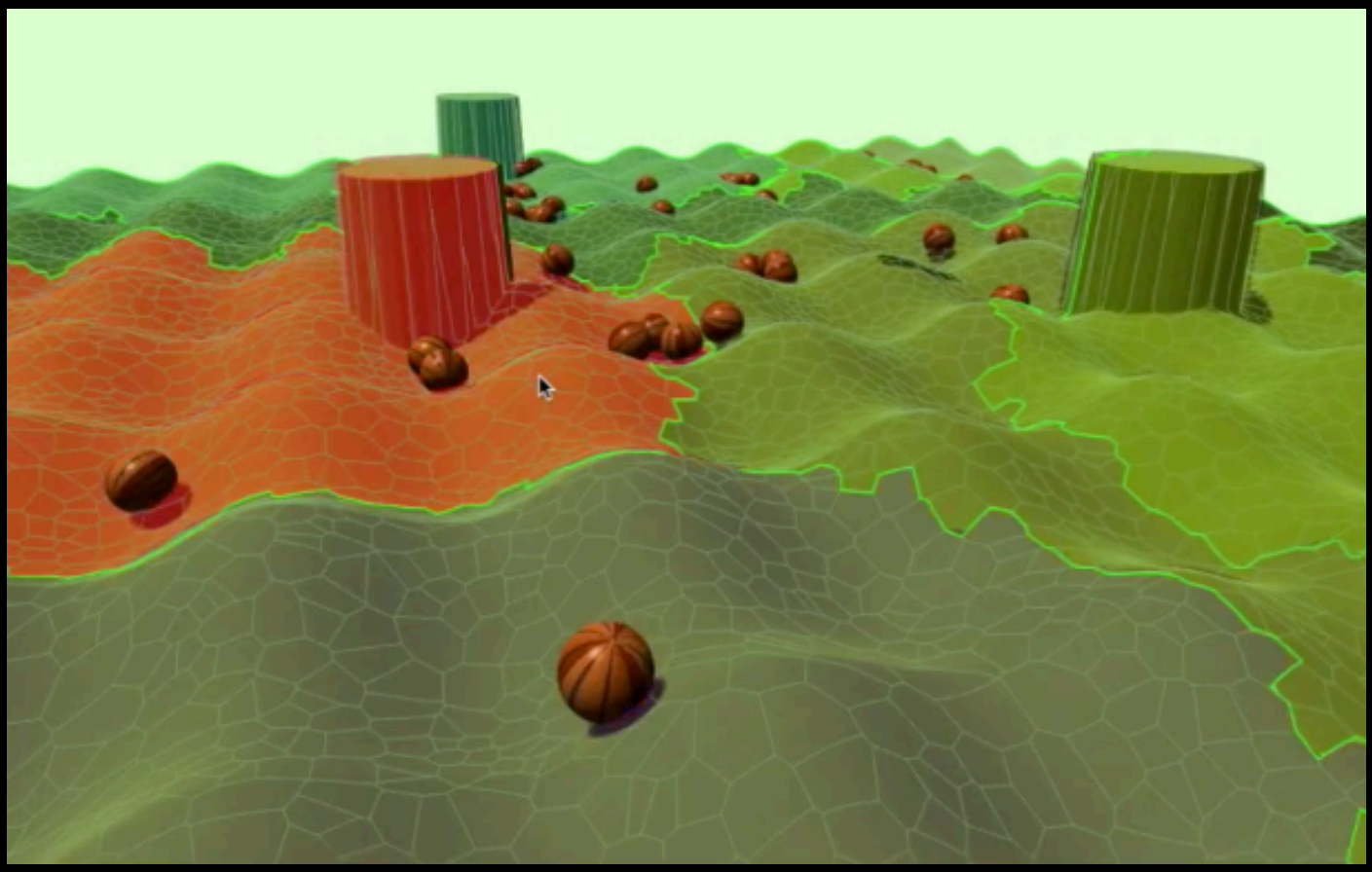
TERRAIN GRID SYSTEM



RELEASE NOTES

V.1.0 / 7.10.2015

Introduction

Thank you for purchasing!

Terrain Grid System is a commercial asset for Unity 5.1.1 (or later) that allows to:

- Add a configurable and flexible grid to Unity terrain
- Supports voronoi tessellation, boxed, rectangular and hexagonal types.
- Two levels of regions: cells and territories.
- Powerful selectable and highlighting system for both cells and territories.
- Coloring support.
- Different positioning options for best mesh adaptation to the terrain.
- Extensive API (C#) for controlling the grid.
- Can work alone or with Unity's Terrain object

You can use this asset for:

- Organizing content or areas of interest over a terrain
- Provide a generic selectable grid zone without using a terrain
- Allow the user to highlight and select a zone of the map
- Display with different colors regions of the map, either territories or cells
- Manage the grow of different regions, by combining cells into greater cells or other territories.

QuickStart and Demo Scenes

1. Import the asset into your project or create an empty project.
2. Open scene “Demo1” or “Demo2” in Demo folder.
3. Run and experiment with the demonstration.

The Demo scenes contain a TerrainGridSystem instance (the prefab) and a Demo gameobject which has a Demo script attached which you can browse to understand how to use some of the properties of the asset from code (C#).

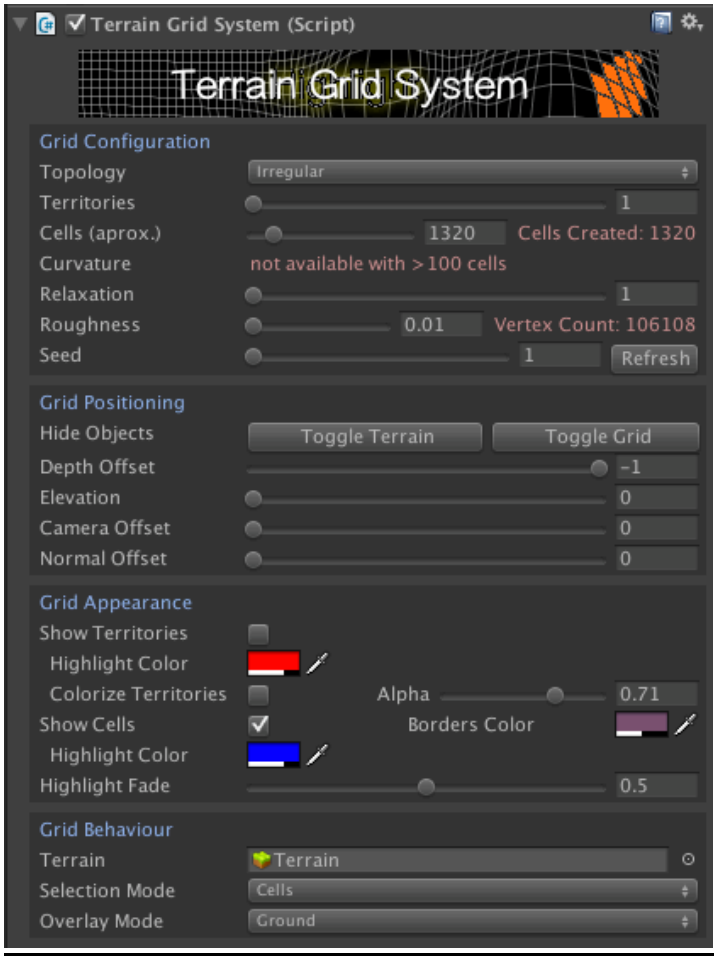
Support & Contact info

We hope you find the asset easy and fun to use. Feel free to contact us for any enquiry.
Visit our Support Forum for usage tips and access to the latest beta releases.

Kronnect Games
Email: contact@kronnect.me
Kronnect Support Forum: http://www.kronnect.me

How to use the asset in your project

Drag the prefab “TerrainGridSystem” from “Resources/Prefabs” folder to your scene.
Select the GameObject created to show custom properties and assign the terrain gameobject reference.



Custom Editor Properties

Grid Configuration: this section allows you to configure the grid irrespective of its appearance or usage. Basically these properties define the number of cells, territories and their shapes.

Topology: this parameters defines the grid overall shape. You can choose between irregular (which uses a highly optimized voronoi tesselation algorithm), boxed, rectangular or hexagonal type.

Territories and Cells: each grid has a number of cells and OPTIONALLY a number of territories. A territories contains one or more cells, so you will always have more cells than territories (or same number).

Note that adding territories will make the system slower with a high number of cells. To prevent lag during runtime while creating the neccesary meshes you can pre-warm the geometry using the API. More on that later.

- **Curvature**: if your grid has 100 or less cells, you can apply an optional curvature. Note that using this parameter will x3 the number of segments of the grid.

- **Relaxation:** this option will make irregular shapes (Voronoi) more homogeneous. Basically it iterates over the voronoi calculation algorithm weighing and modulating the separation between cell centers. Each iteration will redo the voronoi calculation and even it’s been optimized that means that mesh generation times will go up with each relaxation level.

- **Roughness:** this option will determine the level of gripping of the grid to the terrain. The greater the value, more abrupt the grid will appear with respect to the terrain. You usually will want to leave this value to the minimum possible unless your terrain is quite flat (in that case, increasing roughness will reduce vertex count).

- **Seed:** this parameter will allow you to recreate the grid with same disposition of cells. This option basically affects the random functionality.

Grid Positioning

As the grid is a phisically generated mesh, it needs to cover the terrain subtly and effectively. This means that there should be an adequate offset between the terrain geometry and that of the grid. But at the same time that offset should be the minimum possible so the grid stays below any content put over the terrain.
To achieve the best possible effect, Terrain Grid System includes several parameters that allows you to control precisely how the grid is positioned with respect to the terrain mesh.

- **Depth Offset**: this parameter control how much offset is applied to the shaders used by Terrain Grid System. This value affects the Z-Buffer position of the pixels of both highlight surfaces and grid mesh.

- **Elevation**: amount of displacement along the Y-axis.

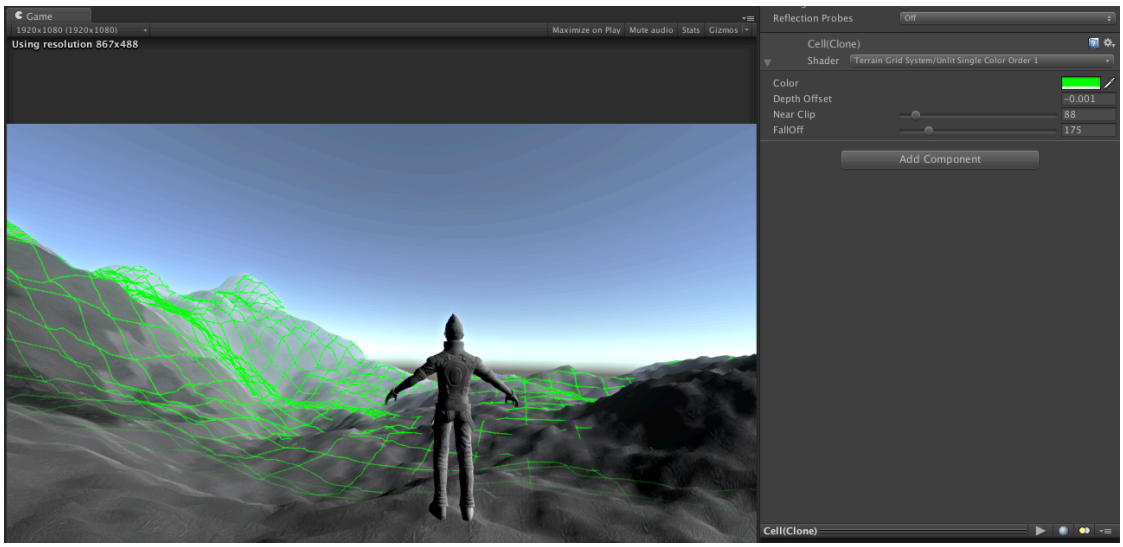
- **Camera Offset**: apply a displacemenet of the grid towards the camera position. This displacement is dynamic, meaning that it will move the grid automatically whenever the grid or the camera changes position.

- **Normal Offset**: this option will apply a displacement per vertex along the normal of the terrain below its position. It will make the grid "grow" around the terrain in all directions. Note that this is an expensive operation. You usually don't need to use this parameter.

Grid Appearance

This section controls the look and feel of the grid system and it should be self-explanatory. However it's important to know that activating either "Show Territories" or "Colorize Territories" (or set the highlight mode to Territory as well) will enable the territory mesh generation. So, if you don't use the territory functionality, make sure all options regarding territories are unchecked to save time and memory.

Grid lines are drawn using custom shaders that allow you to specify a near clip and offset. These properties should be useful for those situations where the camera is really near the grid (like in FPS):



To change the default clipping and falloff values, look for the materials in Resource/Materials folder (Cell and Territory materials).

Grid Behaviour

This section controls what the grid can do.

The first and most important field is the Terrain reference, which you should set to make the grid system start working with your terrain.

In addition to that, you may choose:

- **Selection Mode**: choose between None, Cells or Territories.
- **Overlay Mode**: choose between Ground or Overlay modes. The overlay mode will make the highlight surfaces to be displayed on top of everything. Not very fancy but it's the fastest and prevents any visual artifact with complex terrains. The default option, Ground, will make the highlight surfaces to appear below any content put over the terrain, effectively seeming that it's being painted over the terrain.

Programming Guide (API)

You can instantiate the prefab "TerrainGridSystem" or add it to the scene from the Editor. Once the prefab is in the scene, you can access the functionality from code through the static instance property:

```
TerrainGridSystem tgs;

void Start () {
   tgs = TerrainGridSystem.instance;
         ...
}
```

Public Properties

Cell related

tgs.numCells: desired number of cells. The actual generated cells will vary depending on the topology.

tgs.cells: return a List<Cell> of all cells/states records.

tgs.showCells: whether to show or not the cells borders.

tgs.cellHighlighted: returns the cell/state object in the cells list under the mouse cursor (or null if none).

tgs.cellHighlightedIndex: returns the cell/state index in the cells list under the mouse cursor (or -1 if none).

tgs.cellLastClickedIndex: returns the index of last cell clicked.

tgs.cellHighlightColor: color for the highlight of cells.

tgs.cellBorderColor color for all cells borders.

tgs.cellBorderAlpha: helper method to change only the alpha of cell borders.

Territory related

tgs.numTerritoriess: desired number of territories (1-number of cells).

tgs.territories: return a List<Territory> of all territories.

tgs.showTerritories: whether to show or not the territories borders.

tgs.territoryHighlighted: returns the Territory object for the territory under the mouse cursor (or null).

tgs.territoryHighlightedIndex: returns the index of territory under the mouse cursor (or -1 if none).

tgs.territoryLastClickedIndex: returns the index of last territory clicked.

tgs.territoryHighlightColor: color for the highlight of territories.

tgs.territoryFrontiersColor: color for all territory frontiers.

tgs.territoryFrontiersAlpha: helper method to change only the alpha of territory frontiers.

tgs.colorizeTerritories: whether to fill or not the territories.

tgs.colorizeTderritoriesAlpha: transparency level for colorized territories.

User interaction

tgs.highlightMode: choose between None, Territories or Cells. Defaults to Cells.

tgs.highlightFadeAmount: alpha range for the fading effect of highlighted cells/territories (0-1).

tgs.overlayMode: decide whether the highlight is shown always on top or takes into account z-buffer which will make it more fancy and part of the scene.

Grid

tgs.seed: seed constant for the random generator.

tgs.gridTopology: type of cells (Irregular, Boxed, Rectangular, Hexagonal, …)

tgs.gridRelaxation: relaxation factor for the irregular topology (defaults 1). It creates more homogeneous irregular grids but has a performance hit during the creation of the grid (not afterwards).

tgs.gridCurvature: will bend cell edges for all topologies. It will add more vertex count to the mesh so it's only available on grids with less than 100 cells.

tgs.gridElevation: vertical offset of the grid with respect to the terrain.

tgs.gridCameraOffset: dynamic camera-oriented displacement. This will move the mesh towards the camera.

tgs.gridNormalOffset: general offset of the mesh from the terrain. This will make the grid "grow" in all directions. This will impact performance during the mesh creation.

tgs.gridDepthOffset: Z-Buffer offset for the grid system shaders. Very cheap method to prevent z-fighting which can be combined with above methods. You should use this property first to adjust your grid.

tgs.gridRoughness: gripping factor for the mesh. The lower the value the better fit of the mesh to the terrain.

Other

tgs.terrain: reference to the Unity Terrain object (assign your terrain with this field).

tgs.terrainCenter: world position of the center of the terrain.

tgs.instance: reference to the actual Terrain Grid System component on the scene.

Public Methods

tgs.GetCellIndex(Cell): returns the index of the provided Cell object in the Cells collection.

tgs.GetCellPosition(cellIndex): returns the center of a cell in the world space.

tgs.GetCellNeighbours(cellIndex): returns a list of neighbour cells to a specific cell.

tgs.GetTerritoryNeighbours(territoryIndex): returns a list of neighbour territories to a specific territory.

tgs.CellMerge(Cell 1, Cell 2): merges two cells into the first one.

tgs.ToggleCellRegionSurface(cellIndex, visible, color, refreshGeometry): colorize one cell's surface with specified color. Use refreshGeometry to ignore cached surfaces (usually you won't do this).

tgs.HideCellRegionSurface(cellIndex): uncolorize/clears one cell's surface.

tgs.ToggleTerritoryRegionSurface(territoryIndex, visible, color, refreshGeometry): colorize one cell's surface with specified color. Use refreshGeometry to ignore cached surfaces (usually you won't do this).

tgs.HideTerritoryRegionSurface(territoryIndex): uncolorize/clears one cell's surface.