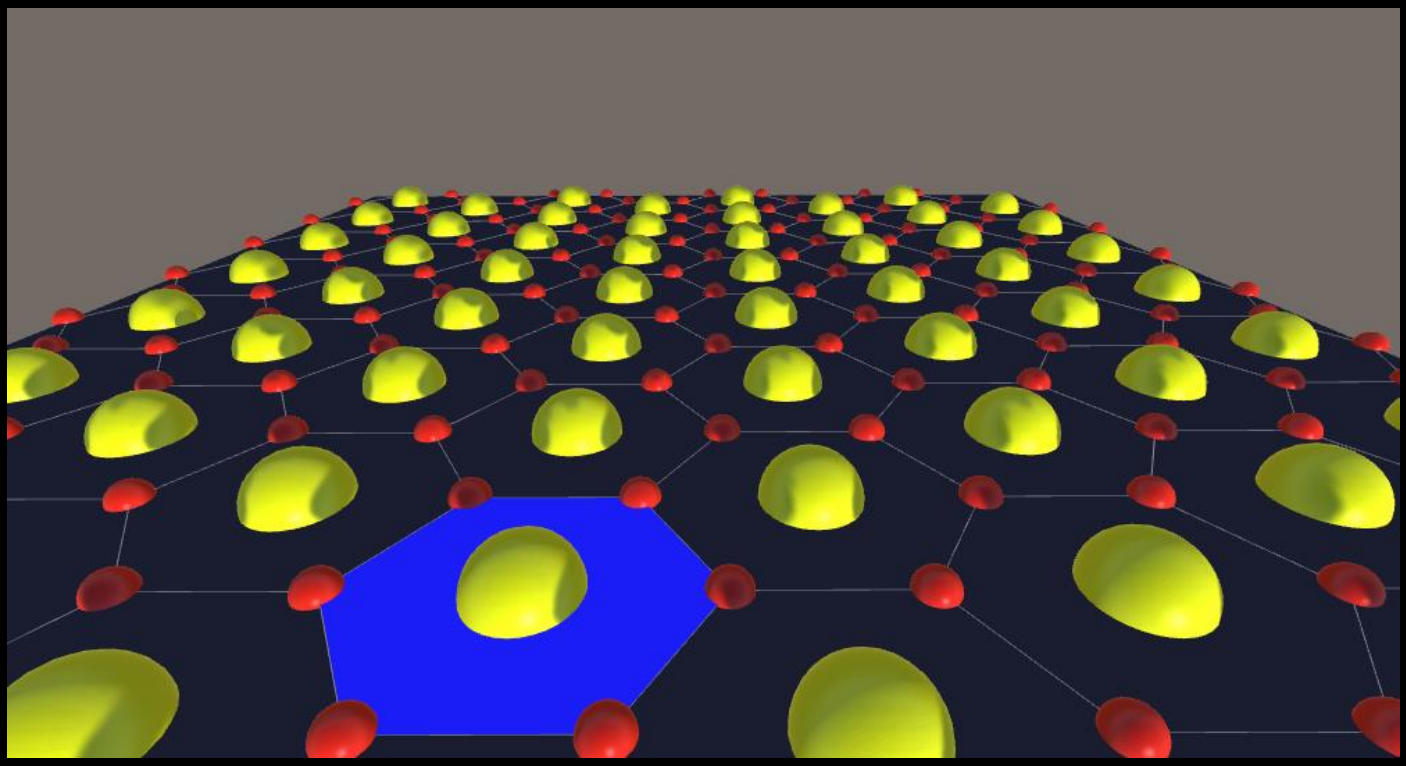
GRIDS 2D SYSTEM



RELEASE NOTES

Contents

Introduction

Thank you for purchasing!

Grids 2D System is a commercial asset for Unity 5.1.1 (or later) that allows to:

- Add configurable, optimized and flexible 2D grid to your scene
- Supports voronoi tessellation, boxed and hexagonal types.
- Two levels of regions: cells and territories.
- Powerful selectable and highlighting system for both cells and territories.
- Define cells visibility or territories using textures or the integrated grid configurator
- Coloring and fade support.
- A* Pathfinding support.
- Extensive API (C#) for controlling the grid.

QuickStart and Demo Scenes

1. Import the asset into your project or create an empty project.
2. Go to Demo folders and run them to learn about common use cases.
3. Examine the code behind the script attached to the Demo game object.

The Demo scenes contain a Grids2D 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#).

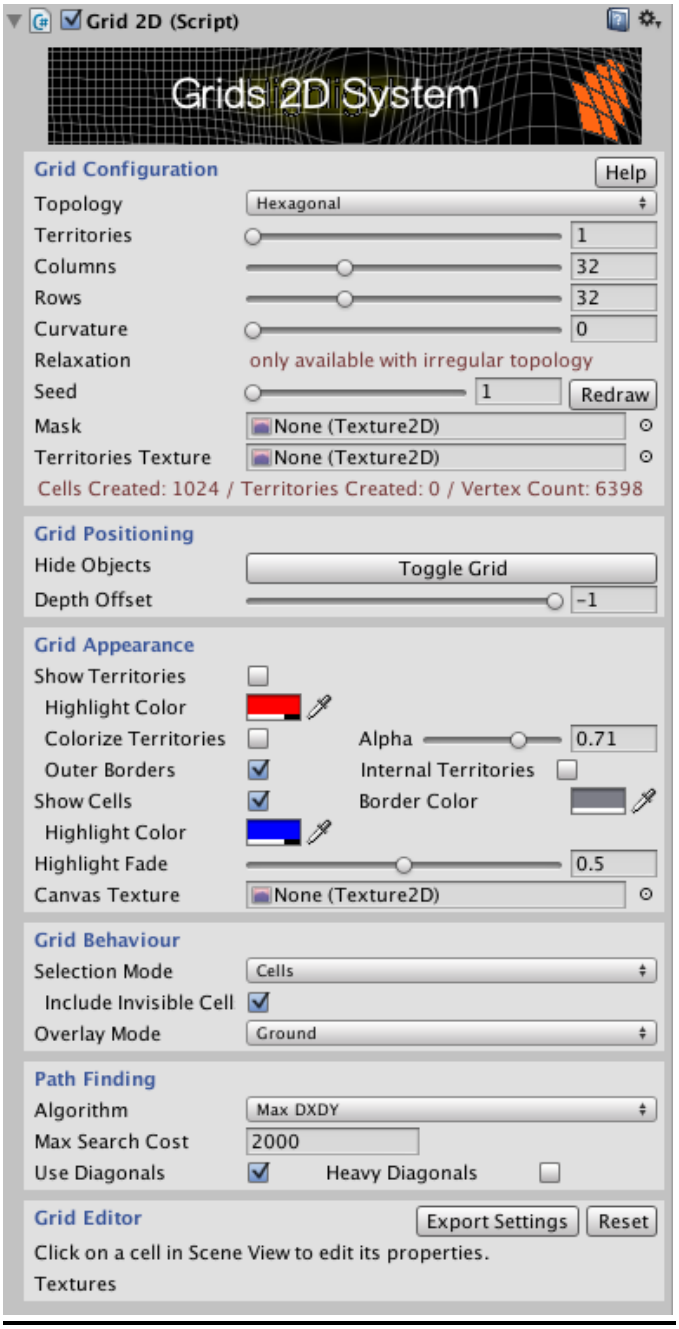
Support & Contact info

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 “Grids2D” from “Resources/Prefabs” folder to your scene.
Select the GameObject created to show custom properties and customize it to your preferences.



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 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.

Columns and Rows: specify the number of cellsin the grid. These two parameters are not available in the irregular topology and instead a single number of cells is shown.

- **Curvature**: if your grid has 100 or less cells, you can apply an optional curvature. Note that using this parameter will multiply by 3 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.

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

- **Mask:** assign a texture in this slot to define cells visibility (alpha channel is used to determine visibility, where a=0 means that cell is invisible).

- **Territories Texture:** assign a color texture to define territories shape or cell ownership. One territory for each different color will be created. Ensure you use solid colors (no smooth or gradients) in the texture. If you use Gimp, you can switch to Indexed Color and set a maximum number of colors, then switch back to RGBA and export the texture in PNG format. Optionally specify which color will be used as neutral, meaning those cell won't belong to any territory.

  When using a territories texture, check the Import settings of the texture:
  Texture type: Advanced
  Read/Write: enabled
  Filter: disabled (disable any bilinear or other kind of filtering)
  Compression: Automatic Truecolor
  Max size: the max width or height of your texture.

Grid Positioning

- **Depth Offset**: this parameter control how much offset is applied to the shaders used by Grids 2D System. This value affects the Z-Buffer position of the pixels of both highlight surfaces and grid mesh.

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.

If you want to make the grid background transparent, you need to change the material of Grids2D to "CanvasBackgroundClear" (material is found inside Grids2D/Resources/Materials folder) and set your camera's Clear Flags setting to Solid Color.

Grid Behaviour

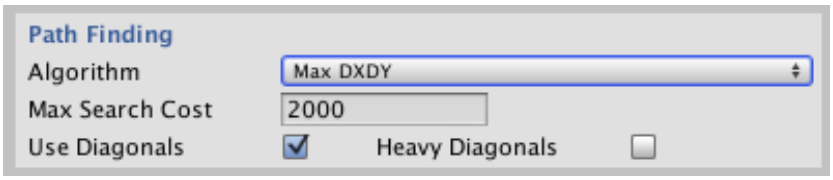
- **Selection Mode**: choose between None, Cells or Territories.

- **Overlay Mode**: choose between Ground or Overlay modes. The overlay mode will make the highlight surfaces to be displayed on top of everything. The default option, Ground, will make the highlight surfaces to appear below any content put over the grid.

- **Respect Other UI**: when enabled, interaction with grid will be disabled when pointer is over an UI element. Note that only uGUI elements are supported (those under a Canvas root element, no immeditate GUI created in OnGUI).

Path Finding

This section controls the path finding behaviour.



- **Algorithm**: choose among available algorithms:
  - **MaxDXDY**: use estimations based on horizontal/vertical distance to target. Produces more natural paths althouth not optimal.
  - **Manhattan**: produces jaggy paths.
  - **Diagonal Shortcut**: favours diagonals in path.
  - **Euclidean**: estimation based on the straight distance to target. Produces the optimal path.
  - **Euclidean Non SQR**: similar but don't uses sqrt which makes it faster.
  - **Custom1**: experimental method.

- **Max Search Cost**: the maximum cost for the returned path.
- **Max Steps**: the maximum number of steps for any path.
- **Use Diagonals**: if diagonals between cells can be used.
- **Heavy DIagonals**: add an extra cost for diagonals, making them less frequent.

The Path Finding algorithm is very fast but TGS also includes an optimized variant of the algorithm for grids that have column count power of 2 (4, 8, 16, 32, 64, 256, …).
Check demo scene 12 for example on using Path Finding with Terrain Grid System. Also read the available APIs for using path finding feature below.

Grids2D provides you with 4 options to customize paths:

- **CellSetCanCross**: calling this method you can simply specify if a cell blocks any path or not.
- **CellSetGroup**: use this method to assign a cell to a custom group. You can later specify which groups are allowed when computing a path calling FindPath method.
- **CellSetCrossCost**: defines a custom crossing cost when entering a cell.
- **CellSetSideCrossCost**: defines a custom crossing cost for a givel cell and a specific edge with option to consider moving direction (ie. entering or exiting the cell through that edge).

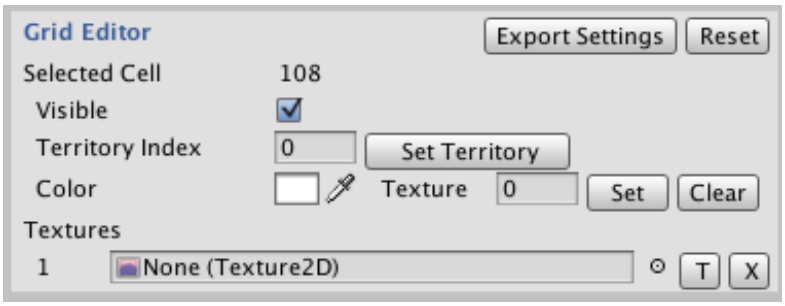
Cell Grouping

By default all cells belong to the Universal group (which has a code number of -1). You can assign a different group to any number of cells using **CellSetGroup** (or wonder which group a cell belongs to calling **CellGetGroup**).

The cell group can be used as a parameter when calling to **FindPath** function in path-finding or **CellGetLineOfSight**. Groups are useful to ignore certain type of cells which might be occupied by some characters that should not be considered either in the route or in the line of sight.

Grid Editor

This section allows you to customize the cells in Editor time.



With the Scene View selected (not Game View), you can use the mouse to select any cell and change their properties.

Export Settings will create add a “Grid2DConfig” script which serves as a container for all the settings and will load them when activated.

Reset will clear all cell settings and reverts them to default values.

Textures array is a convenient way to load several textures and use them quickly to paint several cells. Just click the “T” small buton to enable the “paint mode”. Once enabled, you can click several cells to assign the chosen texture.

You can have several “Grid2DConfig” scripts added. To load a specific configuration just enable the desired Grid2DConfig component or call “LoadConfiguration()” method of that script.

Programming Guide (API)

You can instantiate the prefab “Grids2D“ 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:

```
Using Grids2D;



void Start () {
    Grid2D grid = Grid2D.instance;
    ...
}
```

Complete list of public properties, methods and events

Cell related

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

grid.columnCount: number of columns for boxed and hexagonal topologies

grid.rowCount: number of rows for boxed and hexagonal topologies

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

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

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

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

grid.cellLastClicked: returns the last clicked Cell object.

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

grid.cellHighlightColor: color for the highlight of cells.

grid.cellBorderColor color for all cells borders.

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

grid.CellSetTag: assigns a cell a user-defined integer tag. Cell can be later retrieved very quickly with CellGetWithTag.

grid.CellGetWithTag: retrieves a cell previously tagged with an integer using the method CellSetTag.

grid.CellSetGroup: assigns a cell to a group.

grid.CellGetGroup: get the current group of a cell.

grid.CellGetIndex(Cell): returns the index of the provided Cell object in the Cells collection.

grid.CellGetIndex(row, column, clampToBorders): returns the index of the provided Cell by row and column. Use clampToBorders = true to limit row/column to edges of the grid or to allow wrap around edges otherwise.

grid.CellGetPosition(cellIndex): returns the center of a cell in the world space.

grid.CellGetVertexCount(cellIndex): returns the number of vértices of any cell.

grid.CellGetVertexPosition(cellIndex, vertexIndex): returns the world space position of any vertex of a cell.

grid.CellGetAtPosition(position): returns the cell that contains position (in local space coordinates).

grid.CellGetAtPosition(column, row): returns the cell located at given row and col (only boxed and hexagonal topologies).

grid.CellGetNeighbours(cellIndex): returns a list of neighbour cells to a specific cell.

grid.CellGetNeighbours(cellIndex, distance): returns a list of neighbour cells which are nearer than certain distance to a specific cell.

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

grid.CellToggle (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.CellToggle(cellIndex, visible, color, refreshGeometry, texture): color and texture one cell's surface with specified color. Use refreshGeometry to ignore cached surfaces (usually you won't do this). Texture can be scaled, panned and/or rotated adding optional parameters to this function call.

grid.CellHideRegionSurface(cellIndex): uncolorize/clears one cell's surface.

grid.CellFadeOut(cellIndex, color, duration): colorizes a cell and fades it out for duration in seconds.

grid.CellFlash(cellIndex, color, duration): colorizes a cell and flashes it out for duration in seconds.

grid.CellBlink(cellIndex, color, duration): colorizes a cell and blinks it out for duration in seconds.

grid.CellSetTerritory(cellIndex, territoryIndex): changes the territory of a cell and updates boundaries.

grid.CellGetColor(cellIndex): returns current cell's fill color.

grid.CellGetTexture(cellIndex): returns current cell's fill texture.

grid.CellSetTexture(cellIndex, texture): assigns a new texture to the cell.

grid.CellIsVisible: returns true if the cell is visible.

grid.CellISetVisible: makes the cell visible or invisible. Call Redraw() to update the grid afterwards.

Territory related

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

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

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

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

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

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

grid.territoryHighlightColor: color for the highlight of territories.

grid.territoryFrontiersColor: color for all territory frontiers.

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

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

grid.colorizedTerritoriesAlpha: transparency level for colorized territories.

grid.allowTerritoriesInsideTerritories: this option will allow that a territory surrounds completely another territory. When enabled, the internal territory still can be selected, colorized, textured, …

grid.TerritoryGetIndex(territory): returns the index of the territory object in the Territories list.

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

grid.TerritoryToggle(cellIndex, visible, color, refreshGeometry, texture): color and texture one territory's surface with specified color. Use refreshGeometry to ignore cached surfaces (usually you won't do this). Texture can be scaled, panned and/or rotated adding optional parameters to this function call.

grid.TerritoryHide (territoryIndex): uncolorize/clears one territory's surface.

grid.TerritoryGetNeighbours(territoryIndex): returns a list of neighbour territories to a specific territory.

grid.TerritoryGetAtPosition(position): returns the territory that contains position (in local space coordinates).

grid.TerritoryFadeOut(cellIndex, color, duration): colorizes a territory and fades it out for duration in seconds.

grid.TerritoryFlash(cellIndex, color, duration): colorizes a territory and flashes it out for duration in seconds.

grid.TerritoryBlink(cellIndex, color, duration): colorizes a territory and blinks it out for duration in seconds.

grid.CreateTerritories(texture, neutralColor): automatically generates territories based on a color texture. Use neutral color to ignore areas.

grid.TerritoryIsVisible(territoryIndex): returns true if territory is visible.

grid.TerritorySetVisible(territoryIndex): makes a territory visible or invisible.

User interaction

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

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

grid.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

grid.seed: seed constant for the random generator.

grid.gridTopology: type of cells (Irregular, Box, Hexagonal, …)

grid.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).

grid.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.

grid.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.

grid.voronoiSites: allows you to set the centers of the Voronoi cells.

Path Finding / LoS related

grid.CellSetCanCross: specifies if this cell blocks any path.

grid.CellSetCrossCost: specifies the crossing cost for entering a cell.

grid.CellSetSideCrossCost: specifies the crossing cost for a given side of the cell.

grid.FindPath(cellStartIndex, cellEndIndex, maxCost): returns a list of cell indices that form the path with an optional maximum search cost.

grid.CellGetNeighbours(cellIndex, distance): returns a list of neighbour cells which are nearer than certain distance to a specific cell.

grid.CellGetLineOfSight(startIndex, endIndex, ref cellindices, ref worldPosition, groupMask, lineResolution, exhaustiveCheck): returns true if LoS is available between cell at startIndex and cell at endIndex (also returns additional useful info).

Other useful methods

grid.instance: reference to the actual Grids 2D System component on the scene.

grid.Redraw: forces a redraw of the grid mesh.

grid.DrawLine: draws a line that connect two vertices of a cell or two cells by their centers.

grid.MoveTo: moves a given game object to a destination cell with a custom duration and optional separation from grid.

Events

grid.OnCellEnter: triggered when pointer enters a cell.
grid.OnCellExit: triggered when pointer exits a cell.
grid.OnCellClick: triggered when user clicks or taps a cell (visible or not).
grid.OnCellMouseUp: triggered when releasing the mouse button on a cell (visible or not).
grid.OnCellHighlight: triggered before highlighting cell occurs, allows cancel highlighting.
grid.OnTerritoryEnter: triggered when pointer enters a territory.
grid.OnTerritoryExit: triggered when pointer exits a territory.
grid.OnTerritoryClick: triggered when user clicks or taps a territory.
grid.OnTerritoryHighlight: triggered before highlighting territory occurs, allows cancel highlighting.
grid.OnPathFindingCrossCell: triggered when path finding algorithm estimates cost to this cell. You can use this event to return extra cost and customize path results.

Handling cells and territories

Please, refer to demo scripts included in demo scenes for example of API usage, like merging and toggling cells visibility, …