Sean Miles
Yuto Akutsu
CSS452

Grid System SDK
GTCS Game Engine

Overview
This is a grid system SDK for the GTCS Game Engine, allowing developers to create grids in their game with a specified width and height, and grid objects containing a GameObject that fill a certain width and height of cells in the grid. These grid objects have various functionalities including resizing, reassignment of GameObject, a custom Grid cell bounding box, and locking movement onto the grid which forces the objects to move from cell to cell.

Grids are a common feature in many game engines and one that the GTCS engine is lacking, therefore we decided to implement an API for a grid system.

Grid.js

`Grid(gridSizeX, gridSizeY, cellSizeX, cellSizeY)`

- `gridSizeX:` width of the grid in cells
- `gridSizeY:` height of the grid in cells
- `cellSizeX:` width of one cell in world coordinates
- `cellSizeY:` height of one cell in world coordinates
- Returns: `new GridObject`

Initializes a whole grid system with a grid size of `gridSizeX x gridSizeY` containing cells of size `cellSizeX x cellSizeY`. Also creates a transform for manipulation of the Grid, a 2D array to contain all GridObjects in the Grid, two arrays to contain the LineRenderables for drawing the array, the color to draw the Grid with, a count of the number of GridObjects, and a boolean to toggle drawing of the Grid.

`update()`

- Returns: N/A

Updates the lines of the grid (calls `_setGridLines()`).

`draw(aCamera)`

- `aCamera:` Camera to draw the Grid on
- Returns: N/A

Draws the lines of the grid (calls `_drawGrid()`), if enabled, and calls the draw function of all GridObjects in the Grid.

`getObjFromCell(cellX, cellY)`

- `cellX:` X position to check for an object
- `cellY:` Y position to check for an object
- Returns: `object`

Returns a reference of the object located at position (`cellX, cellY`) in the grid. If the object is a child, the function gets its parent and returns the parent. If there is no object, the function returns undefined.

`getWCFromCell(cellX, cellY)`

- `cellX:` X position to calculate world coordinates
- `cellY:` Y position to calculate world coordinates

- Returns: `vec2`

Returns the converted world coordinates of the position (`cellX, cellY`) in the grid. If the X or Y position is invalid, the function returns a vec2 containing max int values.

`addObj(obj)`

- `obj:` GridObject to add into the Grid
- Returns: N/A

Adds the GridObject (and its children, if it is a parent) to the Grid at its position, if the object is not undefined.

`removeObj(obj)`

- `obj:` GridObject to remove into the Grid
- Returns: N/A

Removes the GridObject (and its children, if it is a parent) from the Grid at its position, if the object is not undefined.

`getXform()`

- Returns: `Transform()`

Returns the Grid's Transform object.

`getNumObjects()`

- Returns: `number`

Returns the number of GridObjects in the Grid.

`getNumRows()`

- Returns: `number`

Returns the number of rows in the Grid.

`getNumCols()`

- Returns: `number`

Returns the number of columns in the Grid.

`getTotalWidth()`

- Returns: `number`

Returns the total width of the Grid in world coordinates.

`getTotalHeight()`

- Returns: `number`

Returns the total height of the Grid in world coordinates.

`getCellWidth()`

- Returns: `number`

Returns the width of a cell in the Grid in world coordinates.

`getCellHeight()`

- Returns: `number`

Returns the height of a cell in the Grid in world coordinates.

`setColor(color)`

- `color:` Color array of 4 values [R, G, B, A] to set
- Returns: N/A

Sets the color of the Grid.

`getColor()`

- Returns: `Array of 4 values`

Returns the color value of the Grid.

`setDraw(bool)`

- `bool:` Boolean to set
- Returns: N/A

Sets the boolean for enabling/disabling drawing of the Grid.

`_setGridLines()`

- Returns: N/A

Helper function for creating LineRenderables of the horizontal and vertical lines of the Grid based on the width and height of the Grid.

`_drawGrid()`

- Returns: N/A

Helper function for drawing the LineRenderables in the Grid.

GridObject.js

`GridObject(obj, grid, cellX, cellY, cellSizeX, cellSizeY, isLocked)`

- `obj:` GameObject to store in GridObject
- `grid:` parent Grid
- `cellX:` X position in the Grid
- `cellY:` Y position in the Grid
- `cellSizeX:` width of the GridObject in amount of cells in the Grid
- `cellSizeY:` height of the GridObject in amount of cells in the Grid
- `isLocked:` boolean of if object is gridlocked
- Returns: `new GridObject`

Initializes a GridObject with a GameObject, the parent grid, an x and y position, the width and height in the amount of cells this GridObject takes up, and if the object is gridlocked. For GridObjects with a size larger than 1x1, parent is initialized as undefined and the child array is made empty.

`draw(aCamera)`

- `aCamera:` Camera to draw the Grid on
- Returns: N/A

Calls the GameObject's draw function, if the GridObject is currently visible.

`getPos()`

- Returns: `vec2`

Returns the GridObject's position in the current grid as a vec2.

`setPos(cellX, cellY)`

- `cellX:` X position in the Grid
- `cellY:` Y position in the Grid
- Returns: boolean if movement occured

Checks for valid cell values, and that every cell in the new position to be occupied is unoccupied, for GridObjects of any size. Then sets the GridObject's position in the current grid as (`cellX, cellY`) and the converted WC position of the object.

`getSize()`

- Returns: `vec2`

Returns the GridObject's size in the current grid as a vec2.

`setSize(cellSizeX, cellSizeY)`

- `cellSizeX:` width of the GridObject in amount of cells in the Grid
- `cellSizeY:` height of the GridObject in amount of cells in the Grid
- Returns: N/A

Checks if the GridObject is able to resize (calls gridMovement) and sets the size in the current grid as `cellSizeX x cellSizeY`. If unable to resize, size is not set.

`isLocked()`

- Returns: `boolean`

Returns if the current GridObject is gridlocked or not.

`lockObject()`

- Returns: N/A

Sets the GridObject's movement to be gridlocked on the current Grid, limiting movement to cells.

`unlockObject()`

- Returns: N/A

Sets the GridObject's movement to be un-gridlocked on the current Grid, freeing movement but still storing the closest grid position for the object.

`getParent()`

- Returns: `GridObject`

Returns the parent of the current GridObject, or undefined if there is no parent. Used for GridObjects with a size larger than 1x1.

`getChildren()`

- Returns: `Array`

Returns the child array of the current GridObject. Used for GridObjects with a size larger than 1x1.

`getBBox()`

- Returns: `BoundingBox`

Creates and returns a new bounding box of the current GridObject.

`setGameObject()`

- Returns: N/A

Reassigns the GameObject in the current GridObject, if valid.

`getGameObject()`

- Returns: `GameObject`

Returns the GameObject stored within the current GridObject.

`getClosestCell()`

- Returns: `vec2`

Returns a vec2 of the closest Grid cell to the current GridObject. Used for un-gridlocked movement.

`addChildren()`

- Returns: N/A

Adds children (dummy GridObjects that point to the origin/parent GridObject) to the current GridObject's child array and the parent Grid. Used for GridObjects with a size larger than 1x1.

`removeChildren()`

- Returns: N/A

Removes children from the current GridObject's child array and the parent Grid. Used for GridObjects with a size larger than 1x1.

Tutorials

Tutorial 1: Creating a Grid:

- Creates a Grid with:
  - 5 as the width of the Grid in amount of cells
  - 5 as the height of the Grid in amount of cells
  - 25 as the cell width of the Grid in world coordinates
  - 25 as the cell width of the Grid in world coordinates

```
this.mGrid = new Grid(5, 5, 25, 25);
```

Tutorial 2: Creating a GridObject:

- Creates a GridObject with:
  - Hero as the GameObject
  - Grid as the parent grid
  - (0, 0) as the position
  - (1, 1) as the size
  - true - GridObject is locked to grid

```
this.mHero = new Hero(this.kMinionSprite, 35, 50);
this.mHero = new GridObject(this.mHero, this.mGrid, 0, 0, 1, 1, true);

this.mGrid.addObj(this.mHero); // adds Hero to the Grid
```

Tutorial 3: Getting and Setting a GridObject's Position:

- Get a GridObject's position
- Set a GridObject to a new position
- Get a GridObject from a cell location

```
this.mHero.getPos()                 // returns 1, 1
this.mHero.setPos(3, 3)             // returns true, slot unoccupied
this.mHero.getPos()                 // returns 3, 3
this.mGrid.getObjFromCell(3, 3)     // returns reference of obj
this.mGrid.getObjFromCell(1, 1)     // returns undefined
```

Tutorial 4: Getting and Setting a GridObject's Size:

- Get a GridObject's size
- Set a GridObject to a new size
- Get a GridObject from a cell location

```
this.mHero.getSize()                // returns 1, 1
this.mHero.setSize(4, 4)       // returns true, slots unoccupied
this.mHero.getSize()                 // returns 4, 4
this.mGrid.getObjFromCell(3, 1)      // returns reference of obj
this.mGrid.getObjFromCell(0, 3)      // returns reference of obj
this.mGrid.getObjFromCell(0, 0)      // returns reference of obj
this.mGrid.getObjFromCell(3, 3)      // returns reference of obj
this.mGrid.getObjFromCell(4, 4)      // returns undefined
```

Tutorial 5: Locking and Unlocking a GridObject's Movement:

- Lock a GridObject's movement to the Grid
    - Cannot modify GameObject's transform when locked
    - GridObject setPos() must be used instead
- Unlock a GridObject's movement to the Grid
    - Can modify GameObject's transform
    - Grid position still stored, using getClosestCell()
    - Can move outside the Grid

```
this.mHero.unlockObject();           // unlocks Hero from Grid
this.mHero.isLocked()                // returns false
this.mHero.getXform()                // returns Hero's transform
// GameObject can move freely

this.mHero.lockObject()              // locks Hero to Grid
this.mHero.isLocked()                // returns true
this.mHero.getXform()                // returns false, cannot modify
// GameObject cannot be accessed, use GridObject function setPos()
```