Tile-Based Map Editor

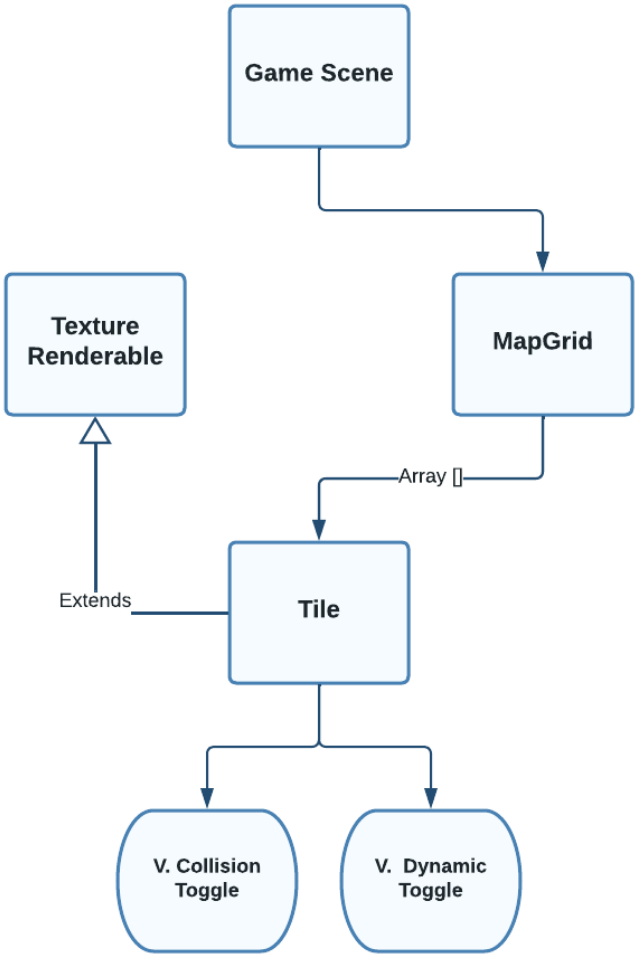
Ani Rao, Mikhail Ermolenko, Trevor Nguyen

Module Description:

Our API supports creating a map based editor scene.

The map supports storing textures in each coordinate as a tile or an object, and allows the user to manipulate the map through adding, removing, or moving objects or tiles. Our module utilizes 2 new classes: MapGrid and Tile. MapGrid allows the user to initialize and create a data structure to store the map, while tile extends texture renderable to allow manipulation of tiles.

The current game engine doesn't support an easy way to create a map. While the user can create texture renderables to create a map grid, the user doesn't have a way to manipulate specific texture objects, or have each object interact with each other without extensive coding. Our module allows ease of use on defining a space to manipulate, storing and placing objects, and interactions between objects in the map. Below describes a diagram of our current API.



API Documentation

Tile Class (extends TextureRenderable):

Tiles are TextureRenderable objects that can be made to contain several properties. They are the core to how the MapGrid is constructed.

Data Fields:

- Collision Toggle (bool)
- Dynamics Toggle (bool)
- Main Texture Image
- BG Texture 1
- BG Texture 2

Methods:

- draw(camera)
    - Draws the tile within the canvas
- update()
    - Event Listener
- setCollisionsMode(bool mode)
    - Sets collisions boundaries for the tile
- getCollisionsMode()
    - Returns Collision Toggle
- getDynamicMode()
    - Returns Dynamics Toggle
- getTexture()
    - Returns Main Texture Image
- toggleEvent()
    - Sets Event Toggle to its opposite
- setDynamicMode(bool mode)
    - Sets Dynamic Toggle to mode
- setFirstTextureObject(tex)
    - Set BG Texture 1
- setSecondTextureObject(tex)
    - Set BG Texture 2
- getFirstTextureObject()
    - Returns BG Texture 1
- getSecondTextureObject
    - Returns BG Texture 2

MapGrid Class:

constructor(width, height)

- Creates a map with resolution width , height

Adding functions:

- createObject(objectPic, xPos, yPos)

    - Creates an object on the map on the tile position xPos, yPos.
- createTilePicturesForGrid()
    - Creates tiles background pictures.
- AddTile(xPos, yPos, tile)
    - Adds individual tiles into the map
- RemoveTile(xPos, yPos)
    - Removes tile from coord

Getters:

- getTile(xPos, yPos)
    - Returns a tile on the position of xPos, yPos (map coordinates)
- getCenterOfTile(xPos, yPos)
    - Returns the center position of a tile in xPos, yPos. (world cord.)

Setters:

- setTile(tile, width, height)
    - Set tiles parameters (tiles background picture, tiles width, tile height)
- setGridColor(color)
    - Set color for the background tile pictures.
- forceSetGridColor(color)
    - Immediately sets the color of the tiles in the grid.
- setTileColor(color)
    - Sets individual tiles color.
- setDynamicModeOfTile(mode, xPos, yPos)
    - Sets the dynamic mode of the tile. The tile mode can be static or dynamic. If it is dynamic, the object can be moved.
- setTileDoor(tilePic, closedPic = null, openPic = null, xPos, yPos)
    - Creates a door. tilePic is a background picture. ClosedPic is a closed door picture. The xPos and yPos grid coordinates of the door.
- setCollisionForObject(index)
    - Set collision mode of the object. The object can have collisions or not.
- setGridPos(xPos, yPos)
    - Set the grid offset in the world. The xPos and yPos are in the world coordinates.

Manipulate:

- moveObjectToSpecTile(index, xPosChange, yPosChange)
    - Moves objects to specific positions in map coordinates. The index is the index of the object.
- moveObjectPicture(index, xPos, yPos)
    - Moves object picture. The xPos and yPos are in map coordinates.
- Async movePictureSmoothly(index, newPos)
    - Make picture movement from one tail to another smooth. The newPos is an arr of xPos and yPos are in map coordinates.

Utility:

- findObjectIndexBasedOnPos(xPos, yPos)
    - Return object on the grind at xPos, yPos position.
- CheckObjectPosition(index, xPos, yPos)
    - Returns true if the index object is in the position specified.
- printGrid()
    - Prints all of the tiles out.
- tilesUpdate()
    - Run all tiles update functions.
- Update()
    - MapGrid update
- draw(camera)
    - Draw the map.
- Async sleep(seconds)
    - Send function to sleep for a number of seconds.

Tutorial

Getting Started

Before you get started, ensure that you have textures that can be added to the map.

To start creating a map, you first need to define a MapGrid object. This object takes 2 parameters: A width, and a height. Next you want to set the MapGrid's position to the position of the map on the canvas. This will be a separate function "setGridPos".

```
this.mGrid = new engine.MapGrid(8,8);
this.mGrid.setGridPos(27,16);
```

To draw onto the map, simply call draw as if it was any other renderable.

Adding Tiles and Objects

Tiles are non-interactables that exist to serve as the "floor" of the map. There are a couple of ways to place them. First being createTilePictureForGrid(). To get started with this function, you will need to set a default tile, that being setTile(). setTile takes 3 parameters: the texture of the tile, the width, and height. This also scales the whole map.

```
this.mGrid.setTile(this.mDefaultTilePic, 8, 8);
this.mGrid.createTilePicturesForGrid();
this.mGrid.setGridColor([.1, .1, .1, .8]);
```

Note that you can also change the whole grid color, using the same array of 4 to set it. Alpha value at 1 will completely make the texture a pure specified color, and is not advised.

To set individual tiles, you use add tile, which takes an x coord, a y coord, and a texture parameter. This places a tile at a specified location.

With objects, it's the same as setting one tile. You use createObject() to create an object on the map. It takes a texture parameter, an x coord, a y coord, and an optional color array. This places an object at a specified location.

```
this.mGrid.createObject(this.mDefaultTilePic, 3, 4, [0, 0, .8, .8]);
this.mGrid.createObject(this.mDefaultTilePic, 4, 4, [0, 0, .8, .8]);
this.mGrid.createObject(this.mDefaultTilePic, 3, 3, [0, 0, .8, .8]);
this.mGrid.createObject(this.mDefaultTilePic, 4, 3, [0, 0, .8, .8]);
```

Demos

https://trevorngu.github.io/452-Final-Project/

https://mikhail10920.github.io/CSS452FinalProject/

Conclusion

Our main goal when creating this API is to allow users an easy way to create and edit maps to play with. We believe that our current API has succeeded in that goal. While there are other ideas we wanted to implement, we wanted to fulfill our main goal before working on other functionalities.

Strengths:

- Ease of use. Doesn't require too many functions to create a map
- Easy to understand. Uses coordinates based on the map size to place tiles.

Weaknesses:

- Lots of function calls when creating complex maps. Not easy to create complex designed maps since there's a ton of function calls to place each tile.
- Defining collision requires 2 functions rather than one.

Improvements:

- Adding an easier way to define collision and tile at the same time.
- Reduce the amount of function calls by creating a function that defines an area rather than a specific point
- Allow scaling with texture size to the map.