Project Journal

5th November 2016

Plans:

1. ~~Research peer-reviewed literature on tile-based game creation, management games, and AI in gaming.~~
2. ~~Begin initial Unity project set up, and create initial basic scripts such as WorldController.~~

List of documents and journals viewed:

1. Tile-Based Game Design – Springer Link

http://link.springer.com/chapter/10.1007/978-1-4302-2740-3\_8

Journal Details:

1. Tile-Based Game Design – Springer Link

Advantages of tile-based games –

Array Storage: Creation of new levels is simple due to every level simply being stored as an array.

Collision Detection: Collisions will only happen when the two objects colliding are next to each other, so only neighbouring tiles need to be checked for collisions.

Simplified AI: In a tile-based world, it is very easy to see for the AI to check what is in the surrounding tiles, and so the AI decision making can be dramatically reduced due to it only being able to move to maybe 4 or 8 tiles.

Efficient use of graphics: Since every object is made up of a fixed number of tiles, making a graphic to fit those tiles is very easy, compared to a normal world where you have non-standard sizes and decimal numbers.

Making Tiles –

All the tiles in a game are the same size, and they all share a standard pixel amount too. 64 by 64 is popular, as well as other multiplies of 2 since computers can handle these number more efficiently.

Tile sheets are used to allow one large image to replace lots of smaller images for multiply things by putting them all onto one image. Due to all the tiles, and thus the sprites, being the same size, you can easily assign different sprites to different coordinates on the tile sheet.

Similarly to the tile sheet, the game world can be easily split into coordinates, making it easy to match the location needed for the sprite to be, and where it is on the tile sheet.

The tile model –

The tile model is a class that represents every tile, this can be done because every tile has the same base characteristics such as location, and type etc.

Putting the map into the game –

Creating a world full of tiles is easy since every row and column needs to be filled with the same tile. A for loop is good for this since it can go along each row and then up each column and create the world of tiles, it needs to know how tall and wide the world will be first though. Once all the tiles are in the world, and their location is stored as an array, it is easy to manipulate individual tiles based on the position.

Adding Game Characters –

The character occupies a single tiles, or a few tiles, just like the walls and floors do. So they also have a position and therefore can be manipulated easily.

Layering Maps –

Foreground images and background images are different, for example, the walls and floor is different from the character since they cannot move. The background images get loaded first, then the foreground map is checked and if there is supposed to be a character in a tile, the foreground image is then rendered on top of the background image. If the foreground image has transparency, then some of the background should be visible also, which is good.

Coding

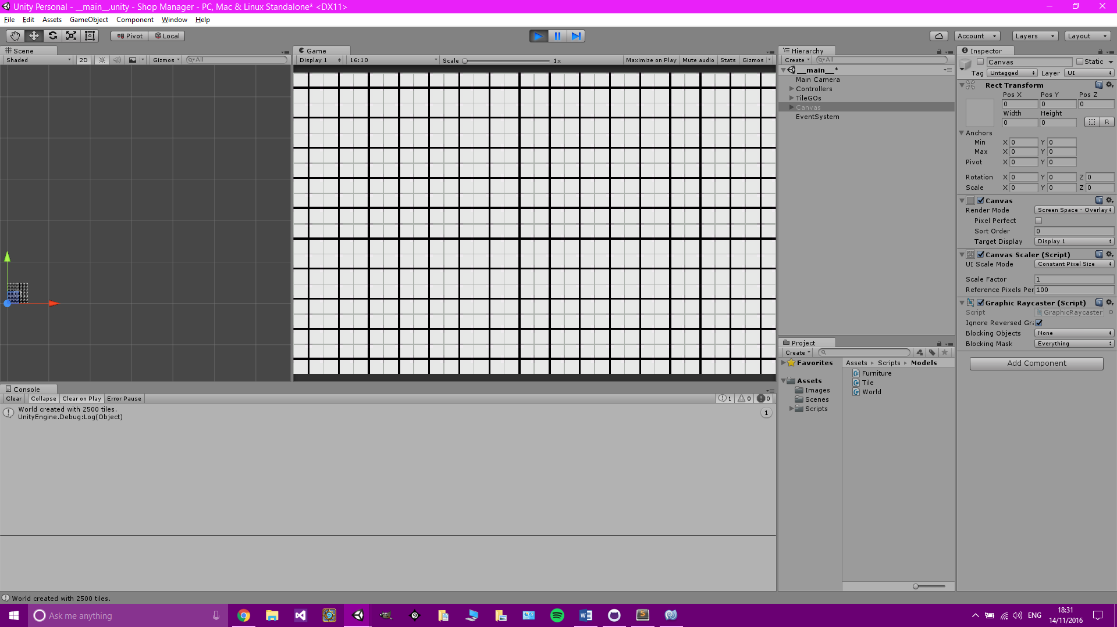
No visible changes on the game screen yet, all background processing.

List of created scripts:

1. World Controller
2. World
3. Tile

Purposes of created scripts:

1. World Controller – This will contain all other elements of the game. It is where tiles and characters get created and destroyed, and their variables get stored such as position. It will be a singleton due to it being the main controller for the game.
2. World - This is a class not derived from monobehaviour. It will be controlled by the World Controller. It will contain all the elements of the current game such as tiles, characters, furniture etc. It currently knows information about all the tiles and the height and width of the world.
3. Tile – This is a class not derived from monobehaviour. The world is filled with tiles and they are sorted into a 2D array. It will contain only functions that affect itself, such as what character or furniture is on it, or what its neighbours are. It also contains information such as its position in the World and its movement cost, which is dependent upon what is in the tile such as characters or furniture.



7th November 2016

Plans

1. ~~Carry on coding and creating the basics for the world.~~
2. ~~Add mouse interactions such as moving/scrolling around the screen.~~
3. Add basics for furniture placement.

Coding

Tile game objects now visible in game. They have a basic sprite.

Camera movement has been implemented. Moving the camera around and zooming in and out works.

Did not get chance to implement basics for furniture placement, will begin with that next session.

Created Scripts

1. Mouse Controller

Purposes of created scripts

1. Mouse Controller – In charge of all mouse movements, clicks, and drags. Interacts with the camera for camera movements. Will eventually deal with furniture previews once they are implemented.

9th November 2016

Plans

1. ~~Add basics for furniture placement.~~
2. ~~Implement furniture~~
3. ~~Add walls to list of furniture~~

Coding

Can now add lots of different furniture with different base types, and sizes.

Created Scripts

1. Furniture
2. Furniture Sprite Controller

Purposes of created scripts

1. Furniture – This is a model which does not inherit from monobehaviour. It is the template for all furniture in the game, including walls and doors.
2. Furniture Sprite Controller – This is a controller in charge of all the sprites used for the furniture. If sprites get changed, or added during gameplay, this class sets all the correct settings for the game objects.

12th November 2016

Plans

1. ~~Finish furniture implementation~~
2. ~~Add UI for furniture placement~~
3. ~~Test furniture placement~~

Coding

Finished adding UI to Unity. Adding more furniture into the game is now very easy.

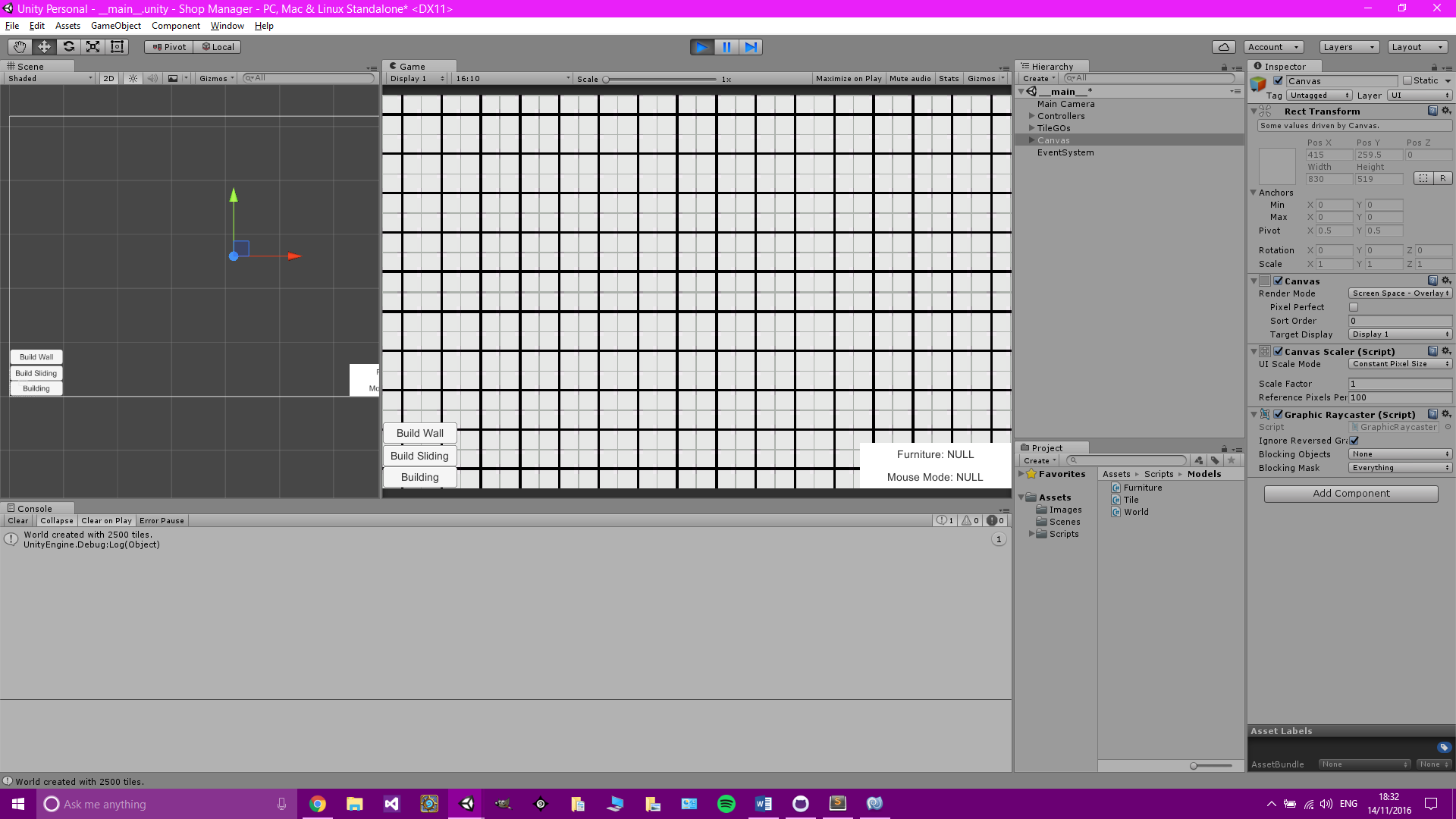
All that that is required is to add the prototype to the world class, and its parameters. Then add a button and the furniture will be created in the world when the button is clicked and a location is set.

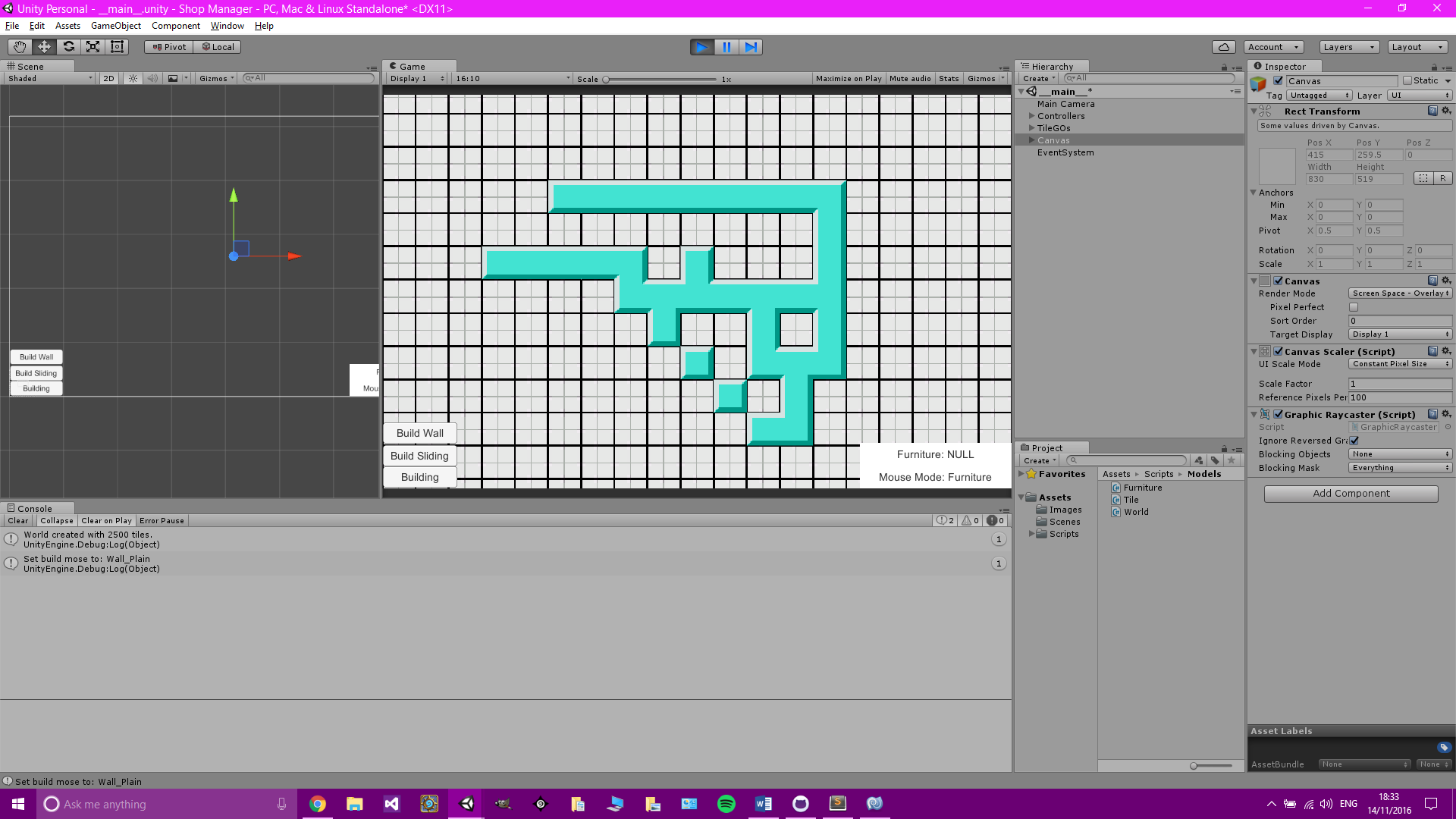
Created Scripts

1. 5 ‘Helper’ scripts that are used to easily change UI menus.

Purposes of created scripts

1. 5 scripts were made to allow a button to be active in the editor so at the click of a button the menu will be re-sized for when new furniture buttons are added.





13th November 2016

Plans

1. ~~Add additional furniture such as doors.~~

Coding

Doors implemented. If a piece of furniture has the ‘Door’ base type, it can only be placed between two pieces of furniture with the ‘Wall’ base type, and will rotate if required. Additional furniture will be added when appropriate, for example, to test or demonstrate another system such as pathfinding.

Created Scripts

1. Furniture Actions

Purposes of created scripts

1. Furniture Actions – This is a static script which is used to allow furniture to have some kind of parameters attached to them which update at certain points. For example, the furniture door needs to be open before it can be walked through, and it cannot go from closed to open instantly, it needs some time in between where it is opening. This is used in conjunction with the furniture update function.