Level Design

* Things to consider
  + Build castle pieces into a 3D grid system
  + Need to generate colliders/meshes according to room design
  + Will need a quick and dirty method in determining wall section(Is this a corner, which side?, etc.)
  + ~~Some floors will need to be “corner floors”~~
    - ~~Have wall AND floor at this location~~
  + Walls appear on the “outside” of room
    - w, f, f, f, f, w
    - w, f, f, f, f, w
    - NOT
    - wf, f, f, f, fw
    - wf, f, f, f, fw
  + Walls will appear as part of the floor
    - Floor tile will determine whether it
* Level Builder Singleton
  + Main object that generates and assigns Tiles and TileBlueprints
  + 3D grid memory of tiles: Type TileData(?)
  + Contains several lists of blueprints
    - Floor
    - Wall
    - Ceiling
    - Door
    - Pillar
    - These are the ones that could possibly appear in the map
  + Builds Levels via coroutine
* Tile
  + MonoBehaviour class attached to actual world view gameobject
    - Gives access to:
      * Mesh Filter
      * Mesh Renderer
      * Collider
* TileData
  + Contains x, y, z coordinates(correspond to indices in 3D map)
  + TileType: what kind of tile can be found here
  + Index of appropriate Blueprint
  + String ID of room they are a part of
* TileBlueprint
  + Contains
    - Mesh
    - Materials
    - Collider Dimensions
* Level Building Techniques
  + How will a castle be built?
    - Rooms?
    - Hallways?
  + Consider dynamic sizes of rooms/hallways
  + Will these be predefined?
* Draft 1:
  + Predefined room shapes(not sizes)
    - Basic rectangular
    - L-shaped hallway
    - 3-way intersection hallway
    - 4-way intersection hallway
    - Rectangular with Arch roof
  + These must account for changes in height(3D grid)
    - Floor Type: Stairs. Immediately convert to stair prefab if changes in elevation are detected
    - Separate list for stairs in Level Builder?
  + Multiple Pass System
    - Step 1: Establish rooms
      * Starting Room “might” always be the same
      * Current Room:
        + Has X number of entrances
      * For each entrance:
        + Determine usable area for the connected room
        + Within max size of a given room
        + Generate next room

Define dimensions

Generate flooring

Generate entrances

Ensure one entrance is connected to the current room

Add room to queue

* + - * After all entrances are connected, move to next room in queue
      * Each tile involved should have String ID of their room
    - Step 2: Build Walls
      * After all rooms have their flooring and are connected via hallways
      * Surround rooms and hallways with walls
      * Walls only appear on the outer most layer of rooms
    - Step 3: Build ceilings
      * Add ceilings to the top layers of rooms
    - Step 4: Decorate!
      * // TODO
* Are Modular room sizes out of scope?
* Should we have predefined rooms with predefined sizes?
* Edit of 1/13/19: Still going to attempt dynamically sized rooms. Need to consider how finer details of room structure will be built
* How to determine wall(types)
  + Walls should only have up to 2 neighbors of the same room.
  + 3 scenarios regarding walls
    - Wall piece is in between two walls
      * Look for non-wall neighbor
      * Build wall facing that neighbor on opposite side
    - Wall piece is in a corner
      * Build wall(s) facing each neighbor on opposite side
    - Wall piece is next to doorway
      * Should be similar to two walls
  + Check priority as follows:
    - Check if corner
    - Check if doorway
    - Check if middle
* Chaining rooms/Building rooms
  + Establish room corners at random!
  + Check if the room you are attempting to establish is overlapping with anything!
  + If it is!
    - Check if there is space to move it slightly
    - If there is!
      * Move it slightly
    - Otherwise
      * Shrink it!