ROOM ALLOCATION SYSTEM MANUAL

COMP3003 Dissertation Project

ID: 14323283

1 Introduction

1.1 Summary

The Room Allocation system is a Unity tool that allows a developer to create a finite space, constructed of Vector3 points, by which rooms can be allocated in such a way that infinite travel is possible and encouraged.

1.2 Prerequisites and Dependencies

To be able to use the sample rooms provided, ***ProBuilder*** must be imported into the project **before** importing the Room Allocation package. The rooms will still work without it, but will not display their walls, ceilings, or floors as they are ***ProBuilder*** objects. The doors themselves will display as they are not. Whilst ***Probuilder*** is not required (rooms can be created as you see fit), it is a recommended starting point. All other requirements, such as materials, should be provided in the Room Allocation package. The package also provides some tests, which are not required for the system to function. If you decide to import them, the relevant test frameworks may need to be added to your project.

The tool also provides ***RoomAllocationOVRIntegration.cs*** to help integrate Oculus virtual reality (OVR) into the system. However, it is recommended that this is only imported if you have the Oculus Integration asset, in addition to the relevant VR packages and system references for the OVR API to be detected. Though, ***RoomAllocationOVRIntegration.cs*** when imported will be completely commented out so it can be safely imported even if you do not intend to use OVR.

1.3 Importing into Unity

The tool is provided in the form of a custom package. After all prerequisites and dependencies have been assessed, simply import the custom package into the Assets folder of your project.

AllocationConstants.cs stores all relevant constant values that need to be accessed, such as tag names, path names or parent transform names. These can be changed as required. Note – the tool contains several Unity tags that can be used to tag specific objects for reference. However, most of these are not required for the system to function. One that is required is the TEST_AREA_TAG_NAME (~ line 34 of ***AllocationConstants.cs***), which will most likely not be defined when importing the package as it is only assigned at runtime; therefore, it is recommended to define this yourself to avoid errors relating to this issue. This is used to destroy rendered geometries when resetting the generation.

1.4 Compatibility

The entire development of this tool was done in Unity 2019.4.18f1, and so this is the only version that has been most rigorously tested. Compatibility with older or newer versions is not guaranteed, but there will likely be no issues providing that the appropriate functionality is present in those versions.

The only tested (and consequently provided) VR compatibility is with the OVR API as described. There are no dependencies in the main body of the system that requires this API, so it can be safely ignored if you do not intend to implement an OVR system.

2 Components

In this section, all relevant components of the system will be detailed and summarised. Note that the code itself contains summaries, and inspector values will be annotated with tooltips where necessary. It is suggested to look at these in addition to the documentation as they also provide details on any parameters and return values.

2.1 Room Archetypes

2.1.1 Summary

Room archetypes represent the foundational structure of each different type of room in the system, dictating their shapes and sizes. They are constructed of 3 main components:

- Doors - these are points for where the doors in the room can be placed.
- Corners - these dictate the boundaries for a room, and any allocation calculations will be done using the corners of the room, so it is important to place them such that they form the exact room shape you want.
- Spawns - these are used when generating the initial room. The spawns allow the system to place a room around the user at a specific position.

Archetypes also have an associated weighting. This will dictate the likelihood of the archetype being selected – setting a weighting to 0 will ensure that the archetype is never selected. The tool provides 17 sample archetypes, though a few of them have had their weights set to 0 as they were deemed to be impractical when designing rooms. These are the 1x1 Sided Rhombus, the 1x1 Triangle and the 0.5x1.5 Rectangle.

You can also specify a minimum and maximum door count. This provides some random variance to the rooms and prevents overloading doors. The system will generate a random number between these values and will enforce that the number of doors will be less than or equal to that value. This does **not** include the door that connects a room to the archetype that created it.

- For example, given a room with 4 doors:
    - Setting the minimum door count to 1 will enforce that at least 1 door will be allocated to if there are viable archetypes to be generated.
    - Setting the maximum door count to 3 will enforce that the maximum number of doors that can be active is 3 (meaning that all doors will be active in the room).
- Note that dead ends take priority over forced inactive doors – this means that a dead end does not count towards the door count.

2.1.2 Creating Room Archetypes

Room archetypes are straightforward to construct, and the necessary prefabs are all provided in ***Resources/Prefabs/Room Archetype Construction***. However, there are specific rules to be followed when doing so:

- Start with the provided Template Archetype prefab, adding it to the active scene.
- Room Archetype objects have 2 children – the ***Room Points*** and ***Room Contents*** objects. When constructing archetypes, it is important to add all relevant objects to the ***Room Points*** child object – the ***Room Contents*** object is used to store the instantiated rooms themselves, and so do not need to be added manually.
- When placing doors, they should be facing forward in the direction leading to another room – that is, their z rotation should be facing away from the centre of the archetype. See the sample archetype doors for more information.
- Structure corners appropriately and exactly. Corners should also be organised in a clockwise fashion, to ensure that the rendering of the archetype works as intended. This is not required, but an incorrect organization can make them rendered incorrectly.
- Spawn points should always be placed within the archetype corner boundaries.

Room archetype points also have gizmos assigned to them, where you can change their colour. These can appear both in the scene editor and at runtime by enabling the gizmos for both the scene and the game. They also have an attached Line Renderer component which draws lines between each of the corners in the order they appear.

2.2 Rooms

2.2.1 Summary

Room prefabs are completely customizable, where their success as a room will be heavily dictated by how the developer designs them. There are sample rooms provided for each of the archetypes (except the ones highlighted as being disabled), though these are not expected to be used for projects and are intended to provide an example of how the rooms could be designed, and how the system works.

Rooms also have the capability of being unique – checking the unique box for a room prefab will ensure that the room is only ever generated once. However, if no rooms of a particular archetype are found the system will render the archetype using a line renderer. Therefore, it is important to ensure that all archetypes with a weighting above 0 have at least 1 room that is not unique to avoid this.

The sample rooms were created using ***ProBuilder*** and make use of several free assets as detailed in the acknowledgements section.

2.2.2 Creating Rooms

Room structure is dictated by the archetype it is assigned to. The general formulation of designing and creating rooms for the tool is as follows:

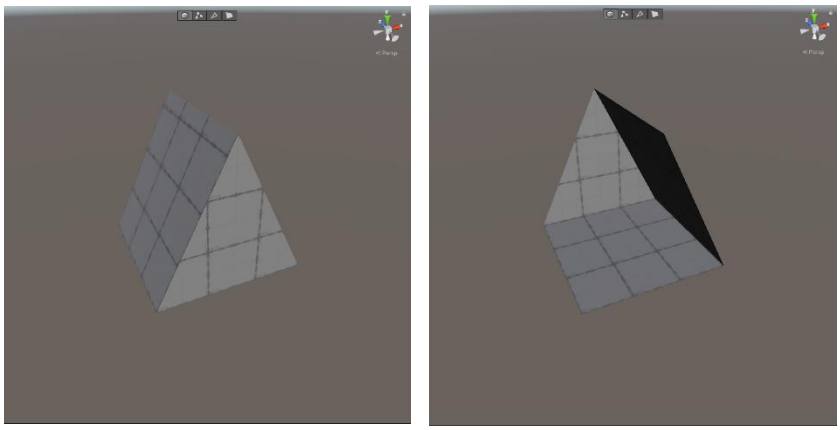
- Add an Empty Room prefab from the Room Construction folder.
- Assign the desired room archetype prefab to the room in the inspector.
- Align the room with the displayed archetype – this is important as if the room is not aligned with the archetype the generation will not work correctly.
- Create a room structure that fits exactly into the specified archetype.
- For each door in the archetype, add a DoorPoint prefab and assign the relevant inspector values:
    - Create an unmasked door object – this will be displayed whenever a door is active.
    - Create a masked door object – this is what is displayed if, for example, a door is a dead end. This should cover the gap and make the gap appear to be part of the walls of the room.
    - Place the DoorPoint on an appropriate Door object as defined in the assigned archetype:
        - For best results, the DoorPoint should be placed such that the Door object is at its bottom centre position:
            - When linking Doors to DoorPoints, the system will simply find the closest DoorPoint to the Door's position. Therefore, these must be placed as close as possible.
        - There should be a gap in the room object wall that fits the door prefab at this position.
    - There are some DoorPoint prefabs provided that can be used as a reference, and these are used throughout the sample rooms.

See the provided sample rooms to see how this works in practice and use them as a reference when designing rooms.

Additionally, you can observe the generation log provided in the documentation folder to see the generation distribution. It is a good idea to design multiple rooms for rooms that appear most often, to avoid constant repetition. This distribution will change dependent on the size and shape of the area being allocated to, as well as the associated weighting.

All sample rooms provided in the tool were created with ProBuilder, which is a package that can be added to any Unity project through the package manager and allows for more advanced geometric manipulations that can be used to make more complex areas and levels. It also provides several features that are particularly useful, as detailed below.

ProBuilder allows you to flip the normals of a particular object. When an object is created, it appears solid from the outside. Flipping the normals will reverse this process; the object will now look like a room, where any observations of the shape from outside it will allow a developer to see through the wall. This is particularly useful for creating rooms quickly, as well as being able to see the inside when observing the system at runtime. ProBuilder is the only package that the system requires to function as desired, and so it is recommended that any project using the tool has ProBuilder imported. Though, this is only cosmetic as it only affects the rendering of the sample rooms and nothing else.



2.3 System Manager

2.3.1 Summary

The System Manager prefab, found at the root of the Prefabs folder, is the only object you need to manually add to the scene. It contains all necessary components, except for the VR Integration scripts that you should add manually if you are using the tool for VR purposes. Simply drag the prefab onto the scene, and you are ready to start using the tool.

The later sections will detail each script that should be attached to the system manager, what it does, what inspector values do and so on.

2.3.2 Logger.cs

This script will create a .csv file in ***Room Allocation/Documentation/GenerationLogs***, and it will generate the folder if it is not found. If logging is enabled, it will then log each room archetype generation alongside numerous other details such as the room size or the tree layer. The latest generation log will be provided in this folder in the package.

- If you do not want logging to take place, untick the Logging Active checkbox in the inspector.
- These logs are for development and debugging purposes only and should be disabled in builds.
- Logging is automatically disabled in VR and should be disabled if this functionality is not provided by the integration scripts.

2.3.3 AllocationManager.cs

This script handles the initial setup of the system. For example: setting up the area, ensuring the camera is in the correct initial position and calling sanity check functions. You can define the target FPS of the application in the inspector.

2.3.4 TestHelper.cs

This script provides a form of user simulation, allowing you to iteratively and quickly test generations for differently sized rectangular or square rooms. You can define many properties, all of which are explained using tooltips in the inspector.

Whilst there are sanity checks for this input, it is recommended that the input makes sense for the test system to work as expected. During development, there were issues encountered when making the test cycle time too low – if there are any issues regarding the camera being stuck, this is probably the case. Issues encountered are highlighted by the checks in the script's sanity check function.

The test system is not required for the system to function, but it is a useful tool to visualise how the system works for a given area or set of areas.

2.3.5 RoomAllocator.cs

This is the most important script for the system, as it is responsible for organising the generation itself. It performs the main generation loop, which checks for viable doors near to the position of the camera and generates archetypes and rooms as appropriate.

- You can change the generation loop rate, which will increase the rate at which the loop is invoked. Though it is recommended to keep the default value, changing this value might improve the performance or reliability of the system in the context of your project and system.
- You can also force archetype rendering. This will essentially block room prefabs from being rendered/assigned, and will use the attached Line Renderer on the archetypes to render instead:
    - This is useful for knowing if the corners of the archetypes are structured properly.
    - This is also useful for viewing the main structure of the generation without worrying about rooms getting in the way.

2.3.6 GeometryManager.cs

This script handles functionality relating to geometry. This can range from creating the area to checking that a room can be placed at a particular point.

- If the OVR integration script is attached:
    - The Boolean value *isVR* is set to true, which forces the system to only use the VR geometry as obtained by the script.
- Without VR:
    - AreaX and AreaZ represent how long the area is in X, and how long the area is in Z respectively.
    - The number of geometry points represents how many points will define the geometry (how many points are tested).
    - The area origin is where your area will start from – this is recommended to start from 0,0,0.
    - Enabling $renderGeometryPoints$ will render a sphere for each point in the geometry, which provides a reference to what the geometry looks like.
    - There are currently 3 shapes available to choose from:
        - Rect:
            - Creates a rectangle.
            - Uses AreaX as its width, and AreaZ as its height.
        - Square:
            - Creates a square.
            - Uses AreaX as its side length.
        - Triangle:
            - Creates a triangle.
            - Uses AreaX as its base width, AreaZ for the triangle height.
            - $TrianglePointOffset$ is how far the top point of the triangle is from the origin in the x-axis.

2.3.7 CameraManager.cs

This script handles most functions relating to the camera – it should be noted that the "camera" is the object that is responsible for the *movement* of the camera, and that acts as the positional reference for the player. This could be the camera itself, or it could a parent object. The camera in this context is used by the system to calculate the distance between doors and open the closest door if it is within the open distance.

- Start Pos is the position in the geometry that the camera starts in. Currently, there are 3 possible values:
    - Corner:
        - This will place the camera in the corner of the geometry area.
        - The Corner Offset is how far this position is from the minimum point of the area:
            - For example, setting the Corner Offset to 0.5 in a 1x1 Square area will place the camera at its centre.
        - This is recommended for use only in Square or Rect geometry shapes.
    - Centre:
        - This will place the camera at the centroid position of the geometry area, by calculating the averages of each corner position.
    - Camera position:
        - This will start the generation from the current position of the camera.
        - This is not recommended for use unless you have defined the camera in a position that will be able to generate rooms.
        - This is set automatically when using the VR integration script.
- Door Open Distance is the distance from a particular door that the camera must be before the door will open automatically:
    - The system will always choose the closest door if doors are both within the minimum distance so that only one door is open at a time.

2.3.8 PathManager.cs

This script handles the management of the path, which uses a custom tree structure. The details of this cannot be altered through the inspector. It also handles the replacement of archetypes. The details of the tree structure will not be explored in detail here; for this, it is recommended to look through the summaries and/or comments when necessary.

2.3.9 RoomManager.cs

This script handles the management of rooms, providing functionality to link room archetypes to rooms and instantiating room prefabs appropriately. For further details, reference the code and summaries.

2.3.10 InitialAllocator.cs

This script handles the generation of the initial room, which is slightly more complex than the general generation. It enforces that a certain number of dead ends cannot be made within the initial generation, to enforce conditions that enable the generation to proceed appropriately. For further details, reference the code and summaries.

2.3.11 InputManager.cs

This script handles the management of inputs and maps specific keys to specific functions.

- Reset Generation Input allows you to specify a key code, which when pressed will restart the generation from its initial starting point as defined by the Geometry Manager.
- Replace Archetype Input allows you to specify a key code, which when pressed will call the relevant function in the PathManager. This will use a raycast to see if the camera is looking at a *closed* door, and will replace the archetype it is attached to, and all subsequent archetypes attached to that. This also makes the current room the user is at the root of the allocation tree. It will then perform the generation again from the generation loop.

2.3.12 RoomAllocationOVRIntegration.cs

This script provides an example integration using OVR for the system. It is not attached to the SystemManager, but it should be if OVR integration is needed.

- The Reset Generation Input and Replace Archetype Input fields allow you to specify an OVRInput value to call the relevant functions as described above.
    - This interfaces with the InputManager functions as described above.

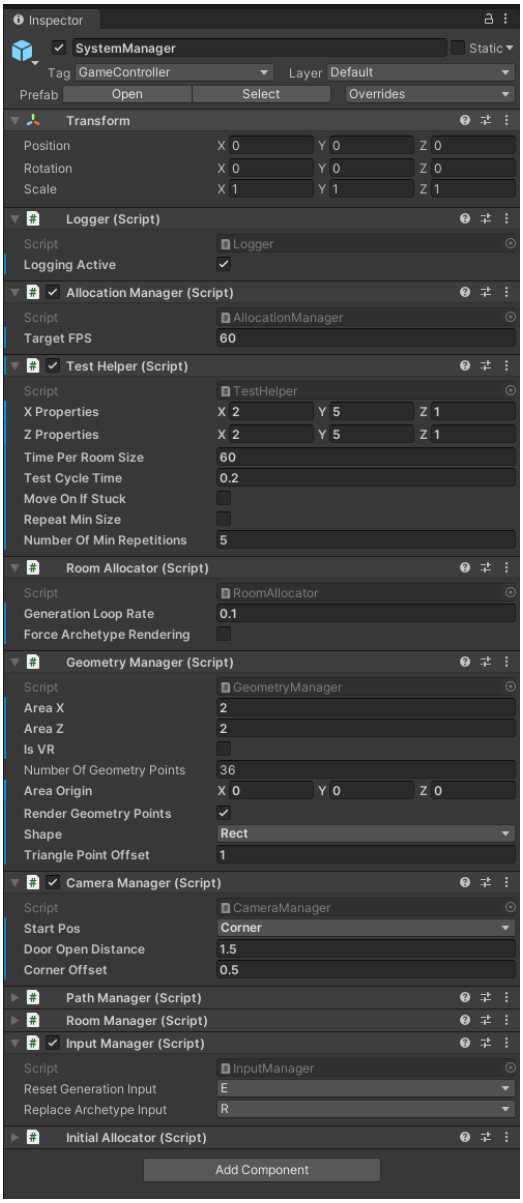
There are no dependencies on OVR in the main body of the system. VR integration works by setting the *isVR* Boolean to true in the GeometryManager, and then obtaining and assigning the Geometry value using the OVR API. The system will use this geometry if *isVR* is true, otherwise, it will create the area as specified by the inspector values.

3 Example Instructions

This section will provide a brief example scenario of using the tool, using the provided ***AlgorithmTestScene*** Unity scene which can be found in the Scenes folder of the package. This will be done using the default settings of each script as provided in the System Manager object, but with ***TestHelper.cs*** disabled. It will also use the provided sample camera prefab.

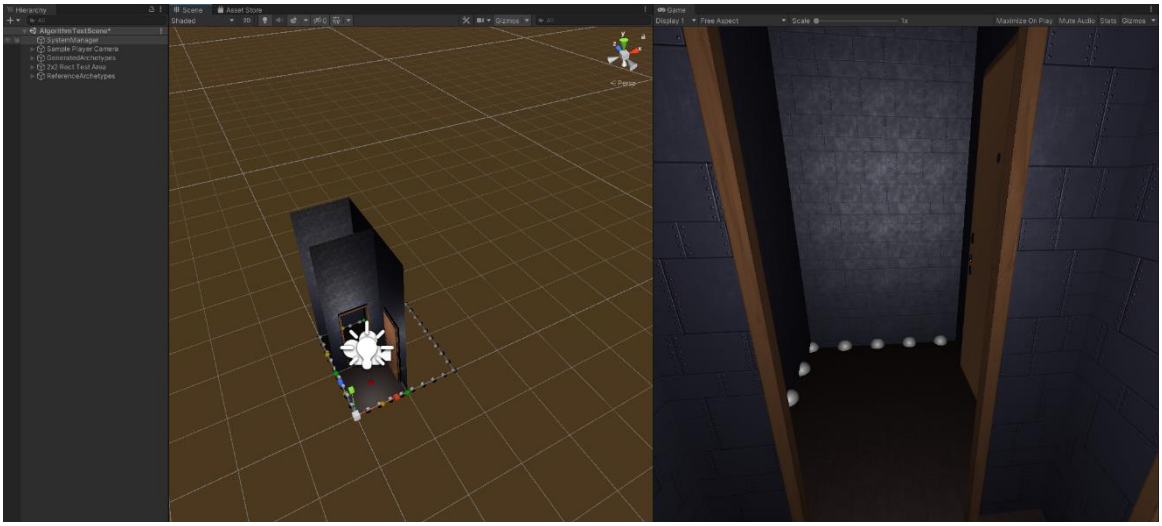
3.1 Evaluating the System Manager

- The initial scene is empty, with only the System Manager and the camera being added to the scene; both of their positions are set to 0,0,0.
- The screenshot to the right shows the initial state of the System Manager object:
    - Logging is enabled, meaning that each archetype generation will be logged.
- Keeping the Test Helper enabled would cause the system to test areas of sizes 2-5 in both X and Z:
    - After testing for a particular value of X or Z, the corresponding value will increase by 1 as indicated by the Z value of the X and Z properties.
    - Each area will be tested for 60 seconds.
    - The test will loop every 0.2s with the current setting:
        - Note that the generation loop rate is 0.1; through development, it was found that the test cycle time should be greater than the generation loop rate by 0.05s to obtain reliable testing.
- As the Test Helper will be disabled, the Geometry Manager will create a 2x2 Rect area, and this will render each point in the geometry.
- The camera will start in the corner of this geometry – an offset of 0.5 will ensure it fits in the centre of 1 of the 4 squares in the grid as dictated by the area.



3.2 Starting the System

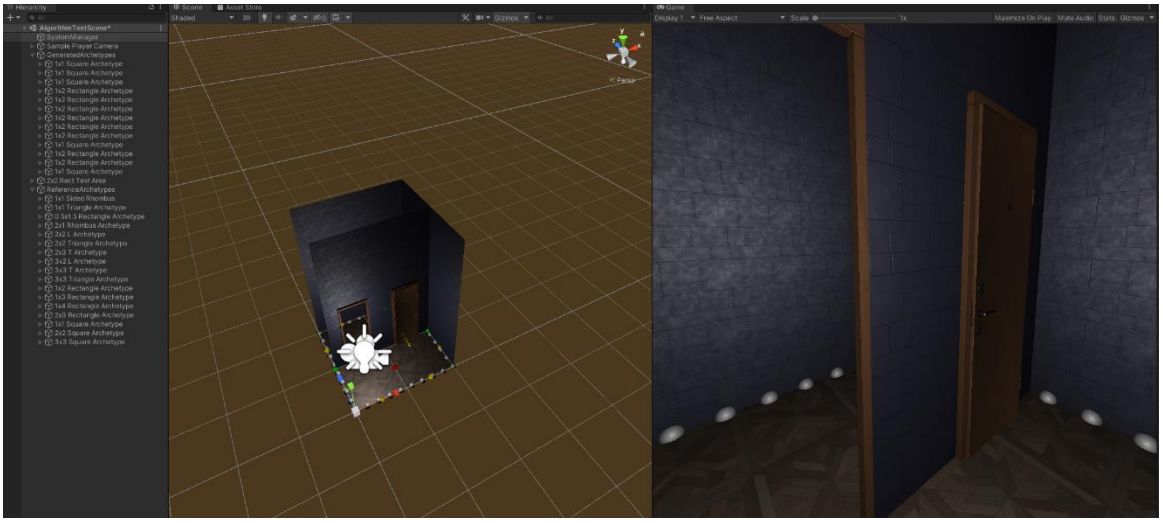
All that is needed is to press the Play button in the Unity editor, upon doing so nothing will happen. To initialise the system, you must press the Reset Generation Input key – this defaults to E. The result should be something like the following:



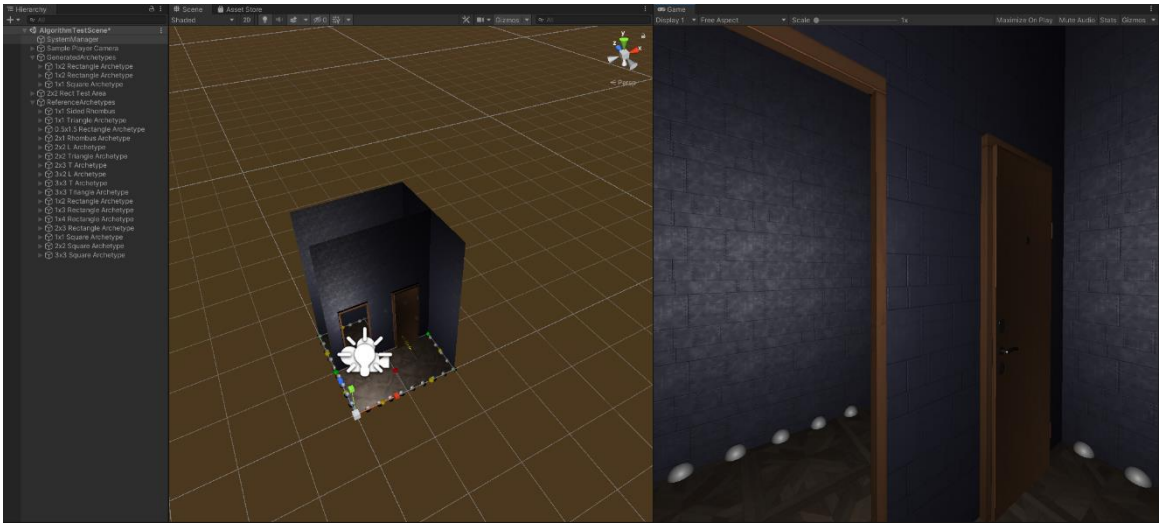
There are a couple of things to note here:

- The system has created the 2x2 area and stored all of the generated points in the 2x2 Rect Test Area object in the Hierarchy.
- The system generates reference archetypes which are used to more efficiently test archetypes to see if they fit in a particular space:
    - These are stored as children of the ReferenceArchetypes object.
- Any other archetypes generated (the ones being interacted with by the user) will be stored as children of the GeneratedArchetypes object.

From this point, you should be able to move the camera (if you are using the provided prefab). None of the sample room walls will block the camera – they will allow you to "walk" through them. Moving the camera will open doors, and consequently generate further rooms. This was done to respect VR implementations.



After moving for some time, several more archetypes and rooms have been generated. Note that the previous archetypes are not disabled – they are simply not rendered. As they are simplistic objects, this did not seem to hurt the performance much if at all. The room objects themselves **are** disabled in the hierarchy when they are not currently rendered. If performance becomes an issue, the Replace Archetype functionality can be used to remove a particular archetype at its associated (closed) door; doing this will delete that archetype and any archetypes that follow on from its path.



After using this function, most of the archetypes have been removed from the scene. This functionality not only provides self-controllable performance management but also allows the users to regenerate a path they may not like. It is important to consider this functionality when developing rooms and experiences.

4 Acknowledgements

4.1 Asset Store

- The door models used in the sample rooms were created as a free asset by Andrey Ferar, and as of 02/04/2021 they can be found using the following link:
    - https://assetstore.unity.com/packages/3d/props/interior/door-free-pack-aferar-148411
- The metal textures used in the sample rooms for the walls and ceilings were created as a free asset by Nobiax/Yughues, and as of 02/04/2021 they can be found using the following link:
    - https://assetstore.unity.com/packages/2d/textures-materials/metals/yughues-free-metal-materials-12949
- The wood textures used in the sample rooms for the floors were created as a free asset by FrOzBi, and as of 02/04/2021 they can be found using the following link:
    - https://assetstore.unity.com/packages/2d/textures-materials/wood/wood-pattern-material-170794
- The Free Fly Camera created as a free asset by Sergey Stafeyev was used to create the Sample Player Camera prefab, which is a camera limited to only moving in the x or z axes. As of 02/04/2021, it can be found using the following link:
    - https://assetstore.unity.com/packages/tools/camera/free-fly-camera-140739