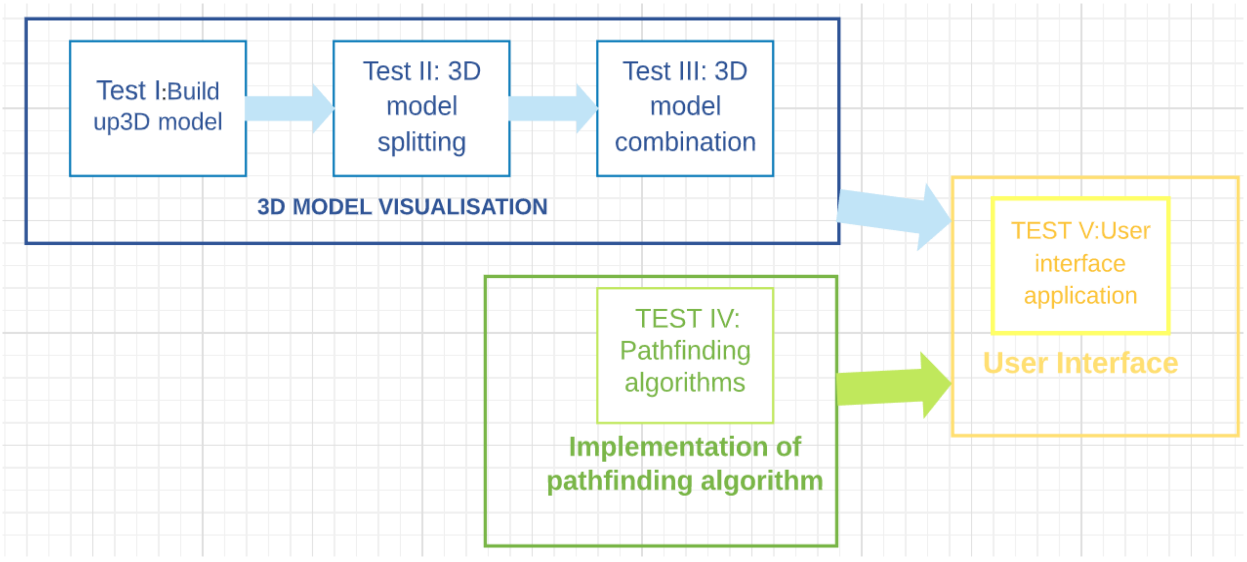
1. Overview

1.1 Introduction

This Acceptance test document mainly focuses on the repeatability and reproducibility of our dataset. The goal of this project was to determine whether 3D overviews of complex buildings like Gaia with indoor navigational features would be of any additional benefit for first responders. It is important to ensure that the process is reproducible by any other user.

We designed several tests for both 3D model visualisation (Test I ~ Test III), path finding algorithms (Test IV) and user interface(Test V). The test consists of a description of what capabilities are needed, the requirements which are being verified by the test such as software and extensions, the test prediction which addresses concerns surrounding the state of the software prior to executing this test and the test steps which is a specific instructions to follow while performing the test.

1.2 Data lineage


Test I:Build up3D model
Test II: 3D model splitting
Test III: 3D model combination
3D MODEL VISUALISATION
TEST V:User interface application
User Interface
TEST IV: Pathfinding algorithms
Implementation of pathfinding algorithm

1.3 Limitation

The limitation for 3D model visualisation is the requirement of floorplan, this step is time consuming. In order to make pathfinding work, grids need to be filled within whole building model. Not only is filling grids time-consuming work but also running pathfinding algorithms requires lots of computational time. It is because the data itself is very heavy to implement on normal desktop. Another difficult occurred in the publishing of user interface. Ideally, the final product is published on WebGL, so that everybody has access to try it. However there where issues with the sdk mapbox that didn’t work when published.

2. Test I: Build up 3D model

2.1 Description and requirements

2.1.1 Description

This test is designed to validate the processing of the raw data, in this case the floorplan to build the indoor 3D model. The Gaia building situated within the Wageningen university campus was set as the test study area. The floorplan was given by Coordinator Property Information. The main idea is using a 2D floorplan to extrude a 3D indoor model.

2.1.2 Requirements

- Software: Blender version 2.8.1 and 2.8.2
- Input Data: The input data is a series of floorplans with dwg extension:
    - 2315-0-basisplattegrond.dwg (Groundfloor floorplan)
    - 2315-1-basisplattegrond.dwg (First floor floorplan)
    - 2315-2-basisplattegrond.dwg (Second floor floorplan)
    - 2315-3-basisplattegrond.dwg (Third floor floorplan)
    - 2315-4-basisplattegrond.dwg (Fourth floor floorplan)
    - 2315-K-basisplattegrond.dwg (Basement floorplan)

Considering the time limitation, we only chose ground floor and first floor as input data for building the model.

2.2 Test predictions

2.2.1 Previous test

No previous tests need to be run to perform this test

2.2.2 Additional software

File extensions convertor such as: DXF convertor and Convertio. They are free access online

2.2.3 data storage

The output data should be saved with blend extension and was committed on GitHub as component data in Blender folder for later process.

2.3 Test steps

1. Convert files to a proper extension:

   Blender is not compatible with dwg files, so to import floorplan, it's necessary to convert the input data into a file type that is recognized by Blender: .dae; .abc; .bvh; .svg; .ply; .stl; .fbx; .glb/gltf; .obj; .x3d/.wrt. In our case we choose svg file as import data.

   Firstly, the dwg file needs to be converted to a to dxf file by using DXF convertor. Next, this dxf file needs to be converted to an svg file by using Convertio. As mentioned before these tools can be found easily online and they are free to use.

2. Import svg file to Blender.:
    - Press delete to get rid of the default cube
    - Open file --> Import --> import Scalable Vector Graphics (.svg) --> select the input svg file (Tips: Remember to turn on 'Import/export svg format' in add-on list in Preference)
3. Build up 3D model by using 2D floorplan in Blender:
    - Select imported floorplan
    - Press S to scale if necessary
    - Use box selection (the default one) to select the whole floorplan
    - Press Ctrl + J to join
    - Select one object in the floorplan for instance: a wall, it will turn to yellow
    - Use box selection again to select whole floorplan, all the rest parts of floorplan will turn to orange colour except the selected wall which will stay in yellow

    - Press F3 and an operator search window will pop up
    - Tap ‘convert to’ as key words and press Enter
    - Choose ‘Mesh from Curve/ Meta/ Surf/ Text’
    - Press Ctrl + J again to join them, all floorplan will turn to yellow
    - Tab into edit model, it will show all lines that can be extrude
    - Press A once or twice to select them
    - Press E + Z to extrude model with desired axis
4. Save 3D model:
    - Open file --> Save as --> Blender file (. blend)
    - Give name as ‘0_Gaia_groundfloor. blend; 1_Gaia_firstfloor. Blend'

2.4 Expect result

The result of this test should be an 3D floor model, that has the same shape as the floorplan and every object that is contained in the floorplan should located as precise as in the floorplan.

3. Test II: 3D model splitting

3.1 Description and requirements

3.1.1 Description

This test is designed to improve the 3D model for implementing path finding algorithm later on. The main idea is splitting walls, columns and stairs into different pieces. In our case, firstly, we split walls from the 3D model and save them as individual object. Remember to give an explicit name that can easily indicate their location such as: ‘0_Right side wall of main hall’ ‘1_middle wall of left corridor’. Here 0 and 1 is used to refer ground floor and first floor. Then, columns and stairs will be split as well. All of them should be exported individually as object file and will be reassembled in Unity. This step will help implementing pathfinding algorithms in the 3D model later. In this way, the pathfinding can be applied with and without showing walls, doors etc.

3.1.2 Requirements

- Software: Blender version 2.8.1 and 2.8.2
- Input Data: The input data is 3D model of Groundfloor and first floor with .blend extension:
    - 0_Gaia_groundfloor. blend
    - 1_Gaia_firstfloor. blend

3.2 Test predictions

3.2.1 Previous test

This test requires test 1 to have been run as it is using 3D models as input dataset.

3.2.2 Additional software

No additional software is required for this test

3.2.3 data storage

The output data should be saved with obj extension and was committed on GitHub as component data in Blender folder for later process in Unity.

3.3 Test steps

1. Import 3D model: directly double click on the saved 3D model in Test 1
2. Splitting wall and columns:
    - Press W change to circle selection

    - Tab into edit model
    - Highlight the object you want to split in scene collection for example: 0_Gaia_groundfloor
    - Choose 'Face select' option on the top left of scene
    - Use circle selection cursor and click on the object you want to selection such as wall, columns stairs.
    - Press 'Shift' to add more selection, press' Ctrl' to deselect objects
    - Press 'P' and choose 'Select'
    - New object will appear in right Scene collection list
    - Rename the new object, assign it a number if necessary
    - Repeat above steps until all desired objects are split from the model
    - Remember to save the project frequently!
3. Export all objects individually:
    - Highlight the object you want to save in scene collection for example: '0_Right side wall of main hall.01'
    - Open file --> Export --> wavefront (.obj)
    - Give it a proper name as mentioned before
    - Repeat above process until all objects are exported individually

3.4 Expect result

The result of this test should be a floor model which is the same as test I, but the only difference is the objects such as walls, doors, stairs are split from floor frame.

4. Test III: 3D model combination

4.1 Description and requirements

4.1.1 Description

This test is designed to combine 2 floor models together in unity. The main idea is combining first floor and ground floor, so that the output will be a complete building and show path finding.

4.1.2 Requirements

- Software: Unity version 2019 3.14f1
- Input Data: The input data is a series of 3D objects of ground floor and first floor with .obj extension:
    - 0_Leftwing big columns_01
    - 0_Leftwing big columns_02
    - 0_Leftwing small columns_01
    - 0_Leftwing wall_01
    - 0_Leftwing wall_02
    - 0_Main hall big columns_01
    - 0_Main hall small columns_01
    - 0_Main hall side wall_01
    - 0_Main hall wall_01
    - 0_Middle hall big columns_01
    - 0_Middle hall big columns_02
    - 0_Middle hall small columns_01
    - 0_Middle hall wall_01
    - 0_Rightwing big columns_01
    - 0_Rightwing small columns_01
    - 0_Rightwing wall_01
    - 0_left spiral stairs wall_01
    - 0_left spiral stairs
    - 0_leftwing straight stairs
    - 0_middle straight stairs wall_01
    - 0_middle straight stairs
    - 0_middle spiral stairs
    - 0_rightwing straight stairs
    - 0_right spiral stairs wall_01
    - 0_right spiral stairs

- 1_Leftwing big columns_01
- 1_Leftwing big columns_02
- 1_Leftwing small columns_01
- 1_Leftwing wall_01
- 1_Leftwing wall_02
- 1_Main hall big columns_01
- 1_Main hall small columns_01
- 1_Main hall side wall_01
- 1_Main hall wall_01
- 1_Middle hall big columns_01
- 1_Middle hall big columns_02
- 1_Middle hall small columns_01
- 1_Middle hall wall_01
- 1_Rightwing big columns_01
- 1_Rightwing small columns_01
- 1_Rightwing wall_01
- 1_left spiral stairs wall_01
- 1_left spiral stairs
- 1_leftwing straight stairs
- 1_middle straight stairs wall_01
- 1_middle straight stairs
- 1_middle spiral stairs
- 1_rightwing straight stairs
- 1_right spiral stairs wall_01
- 1_right spiral stairs

4.2 Test predictions

4.2.1 Previous test

This test requires test I and test II to have been run as it is using 3D objects as input dataset.

4.2.2 Additional software

No additional software is required for this test

4.2.3 data storage

The output data should be saved with. unity extension and was committed on GitHub as project data in Unity folder.

4.3 Test steps

- Import all objects split from ground floor one by one into unity
- Create new game objects for each floor
- Import the different parts into the right game objects
- Set their location as 0,0,0
- Set the scale to the same parameters
- Move the first floor on top of the second floor
- Manually rescale and move wherever necessary

4.4 Expect result

The result of this test should be a complete 3D building model that contains several floors.

5. TEST IV: Pathfinding Implementation

5.1 Description and requirements

5.1.1 Description:

This test is designed to find the best path in 3D building model. The main idea is to fill the ground floor and first floor with cubes to generate a floor grid then implement path finding algorithm on it. The outcome shows a coloured path on the grid which means the shortest path between player and exit. The player, which is represented by a capsule, can horizontally move by pressing W,A,S,D keys and vertically move by pressing CTRL and SPACE keys, and the path will dynamically recalculate after the player’s movement.

5.1.2 Requirements:

- Software: Unity version 2019 3.14f1; Visual Studio with C# for Unity environment
- Input Data: 3D building model

5.2 Test predictions

5.2.1 Previous test:

This test requires Test III to have been run as it is using 3D building model as input data.

5.2.2 Additional software:

No additional software is required for this test

5.2.3 data storage:

The output grid should be saved with. prefab and .prefab.meta extensions and was committed on GitHub as project data in PathFindingCode folder.

The algorithm should be saved with .cs extensions and was also committed on Github as project data in BFS folder.

5.3 Test steps:

5.3.1 Grids filling:

- Import the 3D building model
- Fill the floors and stairs by copy pasting cubes, make sure to avoid walls and columns and save them in separate objects
- Merge those objects into one grid
- If a slight demo version is needed, clip the grid to fit the computer capacity.

5.3.2: Pathfinding algorithm implementation:

Code comment link: https://github.com/yttehs123/ACTIndoorCode

Download the hole zip file of the project data and unzip it. In PathFindingCode folder there are all the scripts. Open them with Visual Studio with C# for Unity environment. Each method in those scripts has its own comment about what is it doing.

In Unity, drag the MovementScript.cs, PlayerWayPoint.cs,PlayerGridSnap.cs, SearchGridSize.cs, PathMode.cs into a capsule, which presents a player, and drag PathFinder.cs into the grid. Drag the end cube to the End Waypoint in Path Finder in the grid panel.

5.4 Expect result

The output is a grid containing pathfinding algorithm. The path will be generated between the capsule and the end cube. The capsule can be moved by pressing WASD, Ctrl and Space in the play mode in Unity and the new path will be generated after that.

6. TEST V: User interface Implementation

6.1 Description and requirements

6.1.1 Description:

This test is designed to showcase all the previous mentioned result in user friendly way. There are three unity scenes provided. The introduction scene, the city view scene and the final building scene. The last scene needs to be adjusted depending on the building.

6.1.2 Requirements:

- Software: Unity version 2019 3.14f1; Visual Studio with C# for Unity environment
- Input Data: 3D building model; pathfinding algorithm

6.2 Test predictions

6.2.1 Previous test:

Before starting this test, all other test needed to be run.

6.2.2 Additional software:

The buttons used in the user interface are created with adobe illustrator.

6.2.3 data storage:

The output data should be saved and committed on GitHub for potential further adjustments. You can publish on

6.3 Test steps

1. Start unity
    - Download webGL if you don’t have it
    - Open the project in unity
2. Import components
    - Load scene 03_BuildingScene
    - Remove the example building and replace it with the new 3d model
    - Remove the example grid and replace and with the pathfinding algorithm
    - Make sure that the grid is on right place inside of the building
3. Add interaction
    - Open all UI buttons and fill empty inputs
    -
    - Click on “play" to see if everything works
4. Publish results
    - File --> Build settings --> PC, mac, linux Standalone
        1. Keep default settings
        2. Choose a folder to publish too
    - File --> Build settings --> WebGl
        1. Keep default settings

ii. Choose a folder to publish too
iii. If you don't have one create an account on: https://simmer.io/
iv. Upload files inside of the folder to simmer and it is online

6.4 Expect result

The output should be a working UI prototype that showcase the 3d model and pathfinding algorithm.

7. Time log

7.1 Test I time log

For extruding 3D model test, it requires a very short time. If users are familiar with Blender and relevant tools, it only takes 10 minutes from import 2D floorplan to get 3D model

7.2 Test II time log

The splitting step is most time consuming one in visualisation section. Depends on the size of model and how many objects need to be split, time log also varies. For our case we spent 3 days to split all objects from 3D model.

7.3 Test III time log

For model combination, it took three hours to put ground floor and first floor together. As all objects have been split and saved individually, they also need to be imported one by one to Unity and reassembled like puzzles to form a complete floor model. In addition, for each object need to be given a location as 0,0,0 so that they can be relocated to the origin.

7.4 Test IV time log

This test is time consuming when trying to fill and merge the grid and write the script. If only do a demo version, which only consists of part of the whole grid, it will consume way less time.

7.5 Test V time log

This test is time consuming, because it contains all previous mentioned components. Depending on the size of the building there can be a lot of components.