House Generator User Manual

Description

The Procedural Generation of a residential house, starting with a basic floor plan derived from a room-expansion algorithm, as described in [Jess Martin's paper "Procedural House Generation: A method for dynamically generating floor plans."](http://axon.cs.byu.edu/Dan/673/papers/martin.pdf) We focused on the possibility of the user being able to manipulate and specify the output of the procedural method, so that the final result is aesthetically pleasing but much simpler to create than starting from scratch. Additions include: removing walls, moving room-locations, and shrinking and scaling rooms, with further additions remaining on customizing the space with furniture, doors and windows, each of which will have modifiable positions.

Compiling and Running

Use the command “make all” in the Project Files directory.

Interaction

Camera

W, A, S, D : movement

Arrow Keys : direction

Numpad +, - : change incline

Numpad * : reset view

Numpad / : reset camera

The Rooms

Assume that at the start of the program, the top of the screen is North.

Martin's algorithm uses a method called 'expansion,' the process of which is visible in our program. At first, rooms are only a point. From there, they start expanding until they encounter other rooms, at which point they stop. If no rooms are encountered, the room will keep growing until its maximum area is reached, after which it stops by itself.

Expansion is always taking place, so that the slightest space between the rooms will cause the rooms to start expanding again, as much as they can. This is very useful in scaling the rooms, as making a room very small in a new space can cause it to 'snuggle' up to the edges of the space instead of needing to be manually scaled to that space. If this is not desired, it is possible to toggle off the expansion. We recommend this for smaller scaling and movement operations. Toggling off expansion shows the perimeter of the house; a kind of 'outer wall' to wrap the rooms inside.

Space Bar : toggle expansion and perimeter display

Holding and dragging a Node : move the room associated with it

Holding and dragging the top right corner of a room : scale the room up or down by its top right position

Holding and dragging the bottom left corner of a room : scale the room up or down by its bottom left position

A Node is one of the black squares, each of which is associated with a room. The rooms start 'expanding,' as described by Martin, from that point. Notice that the graph that shows which rooms ought to be connected to each other is visible as black lines connecting the Nodes. The white squares are the intersections of the graph's lines and the walls, which are the initial locations of doors. Because of this, if the Node's position is moved without moving its room, the location of the door on the room also changes.

Clicking on a Node selects it, and so doing selects its room.

I, J, K, L : delete the wall (of the selected room) associated with the button pressed. IJKL on a QWERTY Keyboard is synonymous with WASD, i.e. I = W = North, K = S = South, J = A = West, and L = D = East

Holding and Dragging a Node with Left CTRL held down : move the Node without moving its associated room

Right CTRL : toggles 3D walls (to better see which walls are deleted) (recall that Space Bar toggles the perimeter walls. This is true of 3D as well)

If 3D is activated, then it is furthermore possible to toggle the visibility of the roof and the ceiling.
F : toggles the roof and ceiling

UI

Esc : close the program
R : restart the room generation process

Other Notes

1. There are three colours to represent Martin's three classes of rooms: green represents Public rooms, such as living or dining rooms. Red represents Private rooms, such as bedrooms. Yellow represents Extra rooms, such as closets. Notice that Extra rooms do not stop Public rooms from expanding or vice-versa, as some Extra rooms can be thought of as being 'contained' by a Public room (think of a walk-in closet, for example). This also allows public rooms to naturally be in L-shaped, U-shaped or O-shaped varieties, giving the rooms more flavour than simply being adjacent rectangles (although this could be done by the user later, if they wanted).

2. Sometimes small gaps are left behind between rooms from the expansion method that cannot be filled in procedurally. The user may either move the other rooms around using our program's manipulative abilities, or simply consider them another room.
3. Private rooms, by their nature, cannot have their walls be deleted.

Bugs

1. The roof in general is a work-in-progress, which is why it lacks a meaningful texture at this point. However, at certain times the roof will find an “intersection” well outside its normal bounds, causing the roof to explode in size.
2. Moving a room a certain amount to the edge causes the program to crash. This may or may not have an easy fix, but it is nonetheless something to be cautious about.
3. Sometimes the algorithm only produces a single-roomed building. The reason is not known. A simple restart should fix it (pressing ‘R’).
4. The front door sometimes goes to private rooms. A simple restart should fix it (pressing ‘R’).
5. Sometimes our room-spreading algorithm fails, causing certain rooms to flip direction back to the direction of the parent. This causes rooms to sometimes spawn on top of each other, causing the rooms to expand in undefined behaviours (mainly expanding flat as a pancake).
6. Sometimes, for no discernible reason, restarting the procedural algorithm (pressing ‘R’) causes a segfault. This happens very rarely, and attempts to capture it have failed.